ДЛЯ ПРОФЕССИОНАЛОВ

HTML5



руководство по разработке
в примерах сообщества Stack Overflow

ДЛЯ ПРОФЕССИОНАЛОВ

HTML5

в примерах сообщества Stack Overflow

Издательство АСТ
г. Москва

УДК 004.43
ББК 32.973.2
Э34

Последнюю версию книги «HTML5 Notes for Professionals»
на английском языке
можно скачать по адресу: https://goalkicker.com/HTML5Book.
Вы можете делиться этим PDF-документом со всеми,
кому он может быть полезен, бесплатно.
Книга создана на основе документации Stack Overflow
(https://archive.org/details/documentation-dump.7z), содержание которой
написано талантливыми участниками сообщества Stack Overflow.
Текстовые материалы распространяются на условиях лицензии
Creative Commons BY-SA.
Список авторов, внесших вклад в создание различных глав книги,
приведен в конце в разделе «Благодарности».
Изображения могут являться объектами авторского права
соответствующих владельцев, если не указано иное.

HTML5 для профессионалов. Руководство по разработке в при-
Э34 мерах от сообщества Stack Overflow. — Москва : Издательство АСТ, 2025. — 160 с. : ил. — (Программирование. Оффер на миллион).

ISBN 978-5-17-178629-8.

Перед вами компактное, но вместе с тем весьма насыщенное руководство по HTML — основополагающему языку разметки, лежащему в основе каждой веб-страницы. Здесь собрана только самая важная и актуальная информация для быстрого и эффективного освоения HTML, от простейших структур до расширенных возможностей современной веб-разметки.

Вы узнаете, как создавать и структурировать контент с помощью заголовков, абзацев, списков, таблиц и мультимедийных элементов. Кроме того, рассматриваются принципы правильного использования ссылок, форм и управляющих элементов, организации навигации и семантической верстки. Особое внимание уделено атрибутам, работе с классами, идентификаторами, кастомными данными, а также взаимодействию HTML с CSS и JavaScript.

Книга охватывает современные возможности HTML5: адаптивные изображения, встроенные аудио- и видеоплееры, элемент <canvas> и SVG-графику, работу с ARIA-атрибутами для доступности, использование метаинформации и глобальных атрибутов. Приведены практические советы по оптимальной разметке, поддержке кросс-браузерности и улучшению UX.

Это издание станет надежной опорой как для начинающих программистов, делающих первые шаги в веб-разработке, так и для опытных специалистов, желающих систематизировать и расширить свои знания. Лаконичные объяснения, четкая структура и множество практических примеров делают эту книгу незаменимым инструментом в освоении и реальном применении HTML.

УДК 004.43
ББК 32.973.2

ISBN 978-5-17-178629-8

Перевод на русский язык: ООО «Интеджер».
Издание на русском языке: ООО «Издательство АСТ».

Содержание

Глава 1. Начало работы с HTML **9**
1.1. Hello World 9
Глава 2. Доктайпы (Doctypes) **12**
2.1. Добавление доктайпа 12
2.2. Доктайп HTML5 12
Глава 3. Заголовки **14**
3.1. Использование заголовков 14
Глава 4. Абзацы **16**
4.1. Абзацы в HTML 16
Глава 5. Форматирование текста **17**
5.1. Выделение 17
5.2. Жирный, курсив и подчеркивание 18
5.3. Аббревиатура 19
5.4. Вставленный, удаленный или зачеркнутый текст 19
5.5. Верхний и нижний индекс 19
Глава 6. Якоря и гиперссылки **20**
6.1. Ссылка на другой сайт 21
6.2. Ссылка на якорь 21
6.3. Ссылка на страницу того же сайта 22
6.4. Ссылка, которая набирает номер 23
6.5. Открытие ссылки в новой вкладке/окне 23
6.6. Ссылка, которая выполняет JavaScript 23
6.7. Ссылка, которая запускает почтовый клиент 24
Глава 7. Списки **26**
7.1. Упорядоченный список 26
7.2. Неупорядоченный список 28
7.3. Вложенные списки 28
7.4. Список описаний (description list) 29
Глава 8. Таблицы **31**
8.1. Простая таблица 31
8.2. Объединение столбцов или строк 31
8.3. Группы столбцов 33
8.4. Таблица с элементами thead, tbody, tfoot и caption 34
8.5. Область заголовка 35
Глава 9. Комментарии **38**
9.1. Создание комментариев 38
9.2. Комментирование пробелов между строчными элементами 39

Глава 10. Классы и идентификаторы **40**
10.1. Назначение элемента класса 40
10.2. Назначение элемента идентификатора 41
10.3. Допустимые значения 42
10.4. Проблемы, связанные с дублированием идентификаторов 45
Глава 11. Атрибуты данных (Data Attributes) **46**
11.1. Поддержка в старых браузерах 46
11.2. Использование атрибутов данных (Data Attribute Use) 46
Глава 12. Связывание ресурсов **47**
12.1. JavaScript 47
12.2. Внешняя таблица стилей CSS 48
12.3. Favicon 49
12.4. Альтернативные таблицы стилей 49
12.5. Подсказки для ресурсов: dns-prefetch, prefetch, prerender 50
12.6. Атрибут media в теге <link> 51
12.7. Предыдущая и следующая страницы 51
12.8. Веб-фид (Web Feed) 51
Глава 13. Включение JavaScript-кода в HTML **52**
13.1. Обработка отключенного JavaScript 52
13.2. Ссылка на внешний файл JavaScript 53
13.3. Прямое включение JavaScript-кода 53
13.4. Включение JavaScript-файла с асинхронным выполнением 53
Глава 14. Использование HTML с CSS **54**
14.1. Использование внешних таблиц стилей 54
14.2. Внутренние таблицы стилей 54
14.3. Встроенные стили 55
14.4. Несколько таблиц стилей 55
Глава 15. Изображения **57**
15.1. Создание изображения 57
15.2. Выбор альтернативного текста 58
15.3. Адаптивное изображение с использованием атрибута srcset 59
15.4. Адаптивное изображение с использованием элемента <picture> 60
Глава 16. Карты изображений (Image Maps) **61**
16.1. Введение в карты изображений 62
Глава 17. Элементы управления вводом **63**
17.1. Текстовое поле 64
17.2. Флажки и переключатели 64
17.3. Валидация ввода 67
17.4. Цвет 69
17.5. Пароль 70
17.6. Файл 70
17.7. Кнопка 71
17.8. Отправка формы 73
17.9. Сброс формы 73

17.10. Скрытое поле (Hidden) ... 73
17.11. Телефон ... 73
17.12. Электронная почта ... 74
17.13. Число ... 74
17.14. Диапазон ... 74
17.15. Поиск ... 75
17.16. Изображение ... 75
17.17. Неделя ... 75
17.18. URL ... 75
17.19. Локальные дата и время ... 75
17.20. Месяц ... 76
17.21. Время ... 76
17.22. Глобальные дата и время ... 76
17.23. Дата ... 77

Глава 18. Формы ... 78
18.1. Отправка данных ... 78
18.2. Атрибут target в теге form ... 79
18.3. Загрузка файлов ... 80
18.4. Группировка нескольких полей ввода ... 80

Глава 19. Элемент Div ... 82
19.1. Основное использование ... 82
19.2. Вложенность ... 83

Глава 20. Секционные элементы ... 84
20.1. Элемент <nav> ... 84
20.2. Элемент <article> ... 85
20.3. Элемент <main> ... 87
20.4. Элемент <header> ... 89
20.5. Элемент <footer> ... 89
20.6. Элемент <section> ... 90

Глава 21. Навигационные панели ... 91
21.1. Базовая навигационная панель ... 91
21.2. Навигационная панель в HTML5 ... 91

Глава 22. Элемент <label> ... 92
22.1. Об элементе <label> ... 92
22.2. Базовое использование ... 93

Глава 23. Элемент Output (Вывод) ... 94
23.1. Использование атрибутов For и Form в элементе Output ... 94
23.2. Элемент Output с атрибутами ... 95

Глава 24. Пустые элементы (Void Elements) ... 96
24.1. Пустые элементы ... 96

Глава 25. Медиаэлементы ... 98
25.1. Аудио ... 98
25.2. Видео ... 98
25.3. Использование элементов <video> и <audio> для отображения аудио/видео контента ... 99
25.4. Видео в качестве заголовка или фона ... 100

Глава 26. Элемент Progress **101**
26.1. Прогресс 101
26.2. Изменение цвета полосы прогресса 101
26.3. Запасной вариант с использованием HTML 102
Глава 27. Элементы управления выпадающими меню **103**
27.1. Элемент <select> 103
27.2. Опции 104
27.3. Группы опций 105
27.4. Элемент <datalist> 105
Глава 28. Элемент <embed> **107**
28.1. Основное использование 107
28.2. Определение MIME-типа 107
Глава 29. IFrames (Встроенные фреймы) **108**
29.1. Основы встроенного фрейма 108
29.2. Песочница (Sandboxing) 108
29.3. Установка размера фрейма 109
29.4. Использование атрибута srcdoc 109
29.5. Использование якорей с IFrames 110
Глава 30. Языки контента **111**
30.1. Основной язык документа 111
30.2. Язык элемента 111
30.3. Элементы с несколькими языками 111
30.4. Региональные URL 112
30.5. Обработка атрибутов с разными языками 112
Глава 31. SVG **113**
31.1. Встроенный SVG 113
31.2. Встраивание внешних SVG-файлов в HTML 114
31.3. Встраивание SVG с помощью CSS 114
Глава 32. Canvas (Холст) **116**
32.1. Базовый пример 116
32.2. Рисование двух прямоугольников на <canvas> 117
Глава 33. Метаинформация **119**
33.1. Информация о странице 119
33.2. Кодировка символов 120
33.3. Роботы (robots) 120
33.4. Социальные сети 121
33.5. Управление мобильной версткой 123
33.6. Автоматическое обновление 124
33.7. Распознавание телефонных номеров 125
33.8. Автоматическое перенаправление 125
33.9. Веб-приложение 125
Глава 34. Разметка компьютерного кода **126**
34.1. Блок с использованием <pre> и <code> 126
34.2. Встроенный код с использованием <code> 126

Глава 35. Разметка цитат ... **127**
35.1. Встроенные цитаты с использованием <q> ... 127
35.2. Блочные цитаты с использованием <blockquote> ... 127
Глава 36. Tabindex ... **130**
36.1. Добавление элемента в порядок табуляции ... 130
36.2. Удаление элемента из порядка табуляции ... 130
36.3. Определение пользовательского порядка табуляции (не рекомендуется) ... 130
Глава 37. Глобальные атрибуты ... **132**
37.1. Атрибут contenteditable ... 132
Глава 38. Кеш HTML5 ... **134**
38.1. Базовый пример кеша HTML5 ... 134
Глава 39. Атрибуты событий HTML ... **135**
39.1. События форм HTML ... 135
39.2. События клавиатуры ... 135
Глава 40. Символьные сущности (Character Entities) ... **136**
40.1. Символьные сущности в HTML ... 136
40.2. Распространенные специальные символы ... 136
Глава 41. ARIA ... **138**
41.1. role="presentation" ... 138
41.2. role="alert" ... 138
41.3. role="alertdialog" ... 138
41.4. role="application" ... 139
41.5. role="article" ... 139
41.6. role="banner" ... 140
41.7. role="button" ... 140
41.8. role="cell" ... 140
41.9. role="checkbox" ... 140
41.10. role="columnheader" ... 141
41.11. role="combobox" ... 141
41.12. role="complementary" ... 141
41.13. role="contentinfo" ... 141
41.14. role="definition" ... 142
41.15. role="dialog" ... 142
41.16. role="directory" ... 142
41.17. role="document" ... 142
41.18. role="form" ... 143
41.19. role="grid" ... 144
41.20. role="gridcell" ... 144
41.21. role="group" ... 144
41.22. role="heading" ... 145
41.23. role="img" ... 145
41.24. role="link" ... 145
41.25. role="list" ... 145
41.26. role="listbox" ... 145

41.27. role="listitem" 146
41.28. role="log" 146
41.29. role="main" 146
41.30. role="marquee" 146
41.31. role="math" 147
41.32. role="menu" 147
41.33. role="menubar" 147
41.34. role="menuitem" 147
41.35. role="menuitemcheckbox" 148
41.36. role="menuitemradio" 148
41.37. role="navigation" 148
41.38. role="note" 148
41.39. role="option" 149
41.40. role="progressbar" 149
41.41. role="radio" 149
41.42. role="region" 149
41.43. role="radiogroup" 149
41.44. role="row" 150
41.45. role="rowgroup" 150
41.46. role="rowheader" 150
41.47. role="scrollbar" 151
41.48. role="search" 151
41.49. role="searchbox" 151
41.50. role="separator" 152
41.51. role="slider" 152
41.52. role="spinbutton" 152
41.53. role="status" 152
41.54. role="switch" 152
41.55. role="tab" 153
41.56. role="table" 153
41.57. role="tablist" 153
41.58. role="tabpanel" 153
41.59. role="textbox" 154
41.60. role="timer" 154
41.61. role="toolbar" 154
41.62. role="tooltip" 154
41.63. role="tree" 155
41.64. role="treegrid" 155
41.65. role="treeitem" 155
Благодарности 157

Глава 1.
Начало работы с HTML

Версия	Дата выпуска
1.0	01-01-1994
2.0	24-11-1995
3.2	14-01-1997
4.0	24-04-1998
4.01	24-12-1999
5	29-10-2014
5.1	01-11-2016
5.2	14-12-2017
5.3	24-01-2018

1.1. Hello World

Введение

HTML (Hypertext Markup Language, язык гипертекстовой разметки) использует систему разметки, состоящую из элементов, которые представляют определенное содержимое. *Разметка* означает следующее: с помощью HTML вы объявляете, что именно будет показано зрителю, а не как это будет показано. Визуальное представление определяется с помощью каскадных таблиц стилей (CSS) и реализуется браузерами. Некоторые элементы, такие как `<font>`, которые все еще существуют, полностью устарели и не должны использоваться авторами [1].

HTML иногда называют языком программирования, но он не содержит логики, поэтому является языком разметки. HTML-теги обеспечивают семантическое значение и машинную читаемость содержимого на странице.

Элемент обычно состоит из открывающего тега (`<element_name>`), закрывающего тега (`</element_name>`), которые содержат имя элемента, окруженное угловыми скобками, и содержимого между ними:

```
<element_name>...content...</element_name>
```

Существуют некоторые HTML-элементы, которые не имеют закрывающего тега или содержимого. Они называются пустыми элементами (void elements). К ним относятся `<img>`, `<meta>`, `<link>` и `<input>`.

Имена элементов можно рассматривать как описательные ключевые слова для содержимого, которое они содержат, например, `video` (видео), `audio` (аудио), `table` (таблица), `footer` (футер).

HTML-страница может состоять из сотен элементов, которые затем читаются веб-браузером, интерпретируются и преобразуются в читаемое или озвученное содержимое на экране.

Для этого документа важно отметить разницу между элементами и тегами:

- Элементы: `video`, `audio`, `table`, `footer`
- Теги: `<video>`, `<audio>`, `<table>`, `<footer>`, `</html>`, `</body>`

Понимание элементов

Давайте разберем, что представляет собой тег.

Тег `<p>` представляет собой обычный абзац (paragraph).

Элементы чаще всего имеют открывающий и закрывающий теги. Открывающий тег содержит имя элемента в угловых скобках (`<p>`). Закрывающий тег идентичен открывающему, но с добавлением косой черты (`/`) между открывающей угловой скобкой и именем элемента (`</p>`).

Содержимое может быть помещено между этими двумя тегами:

```
<p>This is a simple paragraph.</p>.
```

Создание простой страницы

Следующий пример HTML создает простую веб-страницу «`Hello World`».

HTML-файлы можно создавать с помощью любого текстового редактора. Файлы должны быть сохранены с расширением `.html` или `.htm`[2], чтобы они распознавались как HTML-файлы.

После создания этот файл можно открыть в любом веб-браузере.

```
<!DOCTYPE html>
<html lang="en"> <!-- Язык документа – английский -->
<head>
    <meta charset="UTF-8"> <!-- Кодировка символов – UTF-8 -->
    <title>Hello!</title> <!-- Заголовок страницы -->
</head>
<body>
    <h1>Hello World!</h1> <!-- Заголовок первого уровня -->
    <p>This is a simple paragraph.</p> <!-- Простой абзац -->
</body>
</html>
```

Разбор простой страницы

Ниже поясняются теги, использованные в примере.

Тег	Значение
`<!DOCTYPE>`	Определяет версию HTML, используемую в документе. В данном случае это HTML5.
`<html>`	Открывает страницу. Никакая разметка не должна идти после закрывающего тега (`</html>`). Атрибут `lang` объявляет основной язык страницы с использованием кодов ISO (например, en для английского).
`<head>`	Открывает раздел `head`, который не отображается в основном окне браузера, но содержит информацию о HTML-документе, называемую метаданными. Он также может содержать импорт внешних таблиц стилей и скриптов. Закрывающий тег — `</head>`.
`<meta>`	Предоставляет браузеру метаданные о документе. Атрибут `charset` объявляет кодировку символов. Современные HTML-документы всегда должны использовать UTF-8, даже если это не является обязательным требованием. В HTML тег `<meta>` не требует закрывающего тега.
`<title>`	Заголовок страницы. Текст, написанный между этим открывающим и закрывающим тегом (`</title>`), будет отображаться на вкладке страницы или в заголовке браузера.
`<body>`	Открывает часть документа, отображаемую пользователям, то есть все видимое или озвученное содержимое страницы. Никакое содержимое не должно добавляться после закрывающего тега `</body>`.
`<h1>`	Заголовок первого уровня для страницы.
`<p>`	Представляет собой обычный абзац текста.

1. HTML5, 11.2 Несоответствующие функции (https://html.spec.whatwg.org/multipage/obsolete.html#non-conforming-features).



2. .htm унаследован от ограничения DOS на трехсимвольные расширения файлов. (https://en.wikipedia.org/wiki/DOS).



Глава 2.
Доктайпы (Doctypes)

Доктайпы (Doctypes) — сокращение от document type (тип документа) — помогают браузерам понять версию HTML, на которой написан документ, для лучшей интерпретации. Объявления доктайпов не являются HTML-тегами и должны располагаться в самом верху документа. В этой теме объясняются структура и объявление различных доктайпов в HTML.

2.1. Добавление доктайпа

Объявление `<!DOCTYPE>` всегда должно быть включено в начало HTML-документа, перед тегом `<html>`.

Версия ≥ 5

См. раздел «Доктайп HTML5» ниже, чтобы подробнее узнать о доктайпе HTML5.

2.2. Доктайп HTML5

HTML5 не основан на SGML (Standard Generalized Markup Language, стандартный обобщенный язык разметки) и поэтому не требует ссылки на DTD (Document Type Definition, определение типа документа).

Объявление доктайпа HTML5:

```
<!DOCTYPE html>
```

Регистронезависимость

Согласно спецификации W3.org HTML5 DOCTYPE Доктайп должен состоять из следующих компонентов в указанном порядке:

> 1. Строка, которая является регистронезависимым совпадением строки «<!DOCTYPE».

Следовательно, следующие доктайпы также являются допустимыми:

```
<!doctype html>
<!dOCtyPe html>
<!DocTYpe html>
```

Спецификации W3.org HTML5 DOCTYPE (https://html.spec.whatwg.org/multipage/syntax.html#the-doctype).



В этой статье на Stack Overflow подробно обсуждается тема Uppercase or lowercase doctype? (https://stackoverflow.com/questions/7020961/uppercase-or-lowercase-doctype)



Глава 3.
Заголовки

HTML предоставляет не только обычные теги для абзацев, но и шесть отдельных тегов заголовков для обозначения заголовков различных размеров и толщины. Они нумеруются от заголовка 1 до заголовка 6, где заголовок 1 имеет самый крупный и жирный текст, а заголовок 6 — самый мелкий и тонкий, вплоть до уровня абзаца. В данной главе подробно описывается правильное использование этих тегов.

3.1. Использование заголовков

Заголовки могут использоваться для описания темы, которую они предваряют, и определяются с помощью тегов <h1>–<h6>. Заголовки поддерживают все глобальные атрибуты.

- <h1> определяет самый важный заголовок.
- <h6> определяет наименее важный заголовок.

Пример определения заголовков:

```
<h1>Heading 1</h1> <!-- Заголовок 1 -->
<h2>Heading 2</h2> <!-- Заголовок 2 -->
<h3>Heading 3</h3> <!-- Заголовок 3 -->
<h4>Heading 4</h4> <!-- Заголовок 4 -->
<h5>Heading 5</h5> <!-- Заголовок 5 -->
<h6>Heading 6</h6> <!-- Заголовок 6 -->
```

Правильная структура важна

Поисковые системы и другие пользовательские агенты обычно индексируют содержимое страницы на основе элементов заголовков, например, для создания оглавления, поэтому использование правильной структуры для заголовков имеет большое значение.

Как правило, статья должна содержать один элемент <h1> для основного заголовка, за которым следуют подзаголовки <h2> — при необходимости можно углубляться на уровень ниже. Если элементы <h1> используются на более высоком уровне, они не должны описывать содержимое более низкого уровня.

Пример документа (дополнительные отступы для иллюстрации иерархии):

```
<h1>Main title</h1> <!-- Основной заголовок -->
<p>Introduction</p> <!-- Введение -->

    <h2>Reasons</h2> <!-- Причины -->

        <h3>Reason 1</h3> <!-- Причина 1 -->
        <p>Paragraph</p> <!-- Абзац -->

        <h3>Reason 2</h3> <!-- Причина 2 -->
        <p>Paragraph</p> <!-- Абзац -->

    <h2>In conclusion</h2> <!-- В заключение -->
    <p>Paragraph</p> <!-- Абзац -->
```

Глава 4.
Абзацы

Элемент HTML	Описание
`<p>`	Определяет абзац
` `	Вставляет одинарный разрыв строки
`<pre>`	Определяет предварительно отформатированный текст

Абзацы являются самым базовым элементом HTML. В данной главе объясняется и демонстрируется использование элемента абзаца в HTML.

4.1. Абзацы в HTML

Элемент HTML `<p>` определяет абзац:

```
<p>This is a paragraph.</p> <!-- Это абзац. -->
<p>This is another paragraph.</p> <!-- Это еще один абзац. -->
```

Вы не можете быть уверены в том, как HTML будет отображаться.

Большие или маленькие экраны, а также измененные размеры окон дадут разные результаты.

В HTML вы не можете изменить вывод, добавляя дополнительные пробелы или строки в ваш код.

Браузер удалит все лишние пробелы и строки при отображении страницы:

```
<p>This is      another        paragraph, extra spaces    will be
removed     by browsers</p>
<!-- Это еще один абзац, лишние пробелы будут удалены браузерами.
-->
```

Глава 5.
Форматирование текста

Хотя большинство тегов HTML используются для создания элементов, HTML также предоставляет теги для форматирования текста, которые применяют определенные стили к частям текста. В этой главе приведены примеры форматирования текста в HTML, такие как выделение, жирный шрифт, подчеркивание, нижний индекс и зачеркнутый текст.

5.1. Выделение

Элемент <mark> является новым в HTML5 и используется для выделения текста в документе «в связи с его важностью в другом контексте». Ссылка на источник (https://www.w3.org/TR/2014/REC-html5-20141028/text-level-semantics.html#the-mark-element).



Наиболее распространенный пример — это результаты поиска, где пользователь ввел поисковый запрос, и результаты отображаются с выделением искомого запроса:

```
<p>Here is some content from an article that contains the
<mark>searched query</mark>
that we are looking for. Highlighting the text will make it easier
for the user to find what they are looking for.</p>
<!-- Вот содержимое статьи, содержащее <mark>искомый запрос</mark>,
который мы ищем. Выделение текста упрощает пользователю поиск
нужной информации. -->
```

Вывод:

Here is some content from an article that contains the searched query that we are looking for. Highlighting he text will make it easier for the user to find what they are looking for.

Стандартное форматирование — это черный текст на желтом фоне, но это можно изменить с помощью CSS.

5.2. Жирный, курсив и подчеркивание

Жирный текст

Чтобы выделить текст жирным, используйте теги `<strong>` или `<b>`:

```
<strong>Bold Text Here</strong>
<!-- Жирный текст здесь -->
```

или

```
<b>Bold Text Here</b>
```

В чем разница? В семантике. Тег `<strong>` используется для обозначения текста, который является фундаментально или семантически *важным* для окружающего текста, в то время как `<b>` не указывает на такую важность и просто представляет текст, который должен быть выделен жирным.

Если вы используете `<b>`, программа для преобразования текста в речь не будет произносить слово(а) иначе, чем остальные слова вокруг него — вы просто привлекаете внимание к ним, не добавляя дополнительной важности. Однако при использовании `<strong>` та же программа озвучит это слово (слова) с другим тоном, чтобы передать, что текст важен.

Курсив

Чтобы выделить текст курсивом, используйте теги `<em>` или `<i>`:

```
<em>Italicized Text Here</em>
<!-- Курсивный текст здесь -->
```

или

```
<i>Italicized Text Here</i>
```

В чем разница? В семантике. Тег `<em>` используется для обозначения текста, который должен иметь дополнительное ударение, в то время как `<i>` просто представляет текст, который должен быть выделен курсивом.

Например, если вы хотите подчеркнуть действие в предложении, можно выделить его курсивом с помощью `<em>`:

```
«Would you just submit the edit already?"
<!-- Не могли бы вы просто отправить правку уже? -->
```

Но если вы указываете название книги или газеты, которое обычно выделяется курсивом стилистически, следует использовать `<i>`:

```
“I was forced to read Romeo and Juliet in high school."
<!-- Меня заставили читать "Ромео и Джульетту" в школе. -->
```

Подчеркнутый текст

Хотя элемент <u> был устаревшим в HTML4, он был возвращен в HTML5 с измененной семантикой — для представления неартикулированных, не текстовых аннотаций. Например, его можно использовать для обозначения опечаток на странице или для китайских имен собственных:

```
<p>This paragraph contains some <u>mispelled</u> text.</p>
<!-- В этом абзаце есть опечатки. -->
```

5.3. Аббревиатура

Чтобы обозначить выражение как аббревиатуру, используйте тег <abbr>:

```
<p>I like to write <abbr title="Hypertext Markup Language">HTML</
abbr>!</p>
<!-- Мне нравится писать <abbr title="Hypertext Markup
Language">HTML</abbr>! -->
```

Если присутствует атрибут title, он используется для отображения полного описания аббревиатуры.

5.4. Вставленный, удаленный или зачеркнутый текст

Чтобы обозначить текст как вставленный, используйте тег <ins>:

```
<ins>New Text</ins>
<!-- Новый текст -->
```

Чтобы обозначить текст как удаленный, используйте тег <del>:

```
<del>Deleted Text</del>
<!-- Удаленный текст -->
```

Чтобы зачеркнуть текст, используйте тег <s>:

```
<s>Struck-through text here</s>
<!-- Зачеркнутый текст здесь -->
```

5.5. Верхний и нижний индекс

Чтобы сместить текст вверх или вниз, используйте теги <sup> и <sub>.

Для создания верхнего индекса:

```
<sup>superscript here</sup>
<!-- Верхний индекс здесь -->
```

Для создания нижнего индекса:

```
<sub>subscript here</sub>
<!-- Нижний индекс здесь -->
```

Глава 6.
Якоря и гиперссылки

Параметр	Описание
`href`	Указывает адрес назначения. Это может быть абсолютный или относительный URL, либо имя якоря. Абсолютный URL — это полный адрес веб-сайта, например http://example.com/. Относительный URL указывает на другой каталог и/или документ внутри того же сайта, например, /about-us/ указывает на каталог «about-us» внутри корневого каталога (/). При указании на другой каталог без явного указания документа веб-серверы обычно возвращают документ index.html внутри этого каталога.
`hreflang`	Указывает язык ресурса, на который ссылается атрибут `href` (который должен присутствовать вместе с этим атрибутом). Используйте значения языков из BCP 47 для HTML5 и RFC 1766 для HTML4.
`rel`	Указывает отношение между текущим документом и связанным документом. Для HTML5 значения должны быть определены в спецификации или зарегистрированы в вики Microformats.
`target`	Указывает, где открыть ссылку, например, в новой вкладке или окне. Возможные значения: `_blank`, `_self`, `_parent`, `_top` и `framename` (устаревший). Принудительное использование такого поведения не рекомендуется, так как это нарушает контроль пользователя над сайтом.
`title`	Указывает дополнительную информацию о ссылке. Эта информация чаще всего отображается в виде всплывающей подсказки при наведении курсора на ссылку. Этот атрибут не ограничивается ссылками, его можно использовать практически во всех HTML-тегах.
`download`	Указывает, что цель будет загружена при нажатии на гиперссылку. Значение атрибута будет именем загружаемого файла. Нет ограничений на допустимые значения, и браузер автоматически определит правильное расширение файла и добавит его к имени файла (например, .img, .pdf и т. д.). Если значение опущено, используется оригинальное имя файла.

Якорные теги (`<a>`) обычно применяются для ссылок на отдельные веб-страницы, но они также могут использоваться для ссылок на разные места в одном документе, часто в оглавлениях, или даже для запуска внешних

приложений. В этой главе объясняются реализация и применение HTML-якорей в различных ролях.

6.1. Ссылка на другой сайт

Это базовое использование элемента `<a>` (якорный элемент):

```
<a href="http://example.com/">Link to example.com</a> <!-- Ссылка
на example.com -->
```

Он создает гиперссылку на URL `http://example.com/`, указанный в атрибуте `href` (гипертекстовая ссылка), с текстом ссылки «`Link to example.com`». Это будет выглядеть следующим образом:

> Link to example.com

Чтобы обозначить, что эта ссылка ведет на внешний сайт, можно использовать тип ссылки `external`:

```
<a href="http://example.com/" rel="external">example site</a> <!--
Пример сайта -->
```

Вы можете ссылаться на сайт, использующий протокол, отличный от HTTP. Например, для ссылки на FTP-сайт можно сделать следующее:

```
<a href="ftp://example.com/">This could be a link to a FTP site</
a> <!-- Это может быть ссылка на FTP-сайт -->
```

В данном случае разница заключается в том, что этот якорный тег запрашивает у браузера пользователя подключение к `example.com` с использованием протокола передачи файлов (FTP) вместо протокола передачи гипертекста (HTTP).

6.2. Ссылка на якорь

Якоря могут использоваться для перехода к определенным тегам на HTML-странице. Тег `<a>` может указывать на любой элемент, у которого есть атрибут `id`. Чтобы узнать больше об идентификаторах, изучите документацию о классах и идентификаторах. Якоря чаще всего используются для перехода к подразделу страницы и используются вместе с тегами заголовков.

Предположим, вы создали страницу (`page1.html`) на множество тем:

```
<h2>First topic</h2> <!-- Первая тема -->
<p>Content about the first topic</p> <!-- Содержание первой темы -->
```

```
<h2>Second topic</h2> <!-- Вторая тема -->
<p>Content about the second topic</p> <!-- Содержание второй темы -->
```

После создания нескольких разделов вы можете создать оглавление в верхней части страницы с быстрыми ссылками (или закладками) на конкретные разделы.

Если вы присвоили атрибут `id` вашим темам, вы можете ссылаться на них:

```
<h2 id="Topic1">First topic</h2> <!-- Первая тема -->
<p>Content about the first topic</p> <!-- Содержание первой темы -->
<h2 id="Topic2">Second topic</h2> <!-- Вторая тема -->
<p>Content about the second topic</p> <!-- Содержание второй темы -->
```

Теперь вы можете использовать якоря в вашем оглавлении:

```
<h1>Table of Contents</h1> <!-- Оглавление -->
<a href='#Topic1'>Click to jump to the First Topic</a> <!-- Нажмите,
чтобы перейти к первой теме -->
<a href=›#Topic2›>Click to jump to the Second Topic</a> <!--
Нажмите, чтобы перейти ко второй теме -->
```

Эти якоря также привязаны к веб-странице, на которой они находятся (`page1.html`). Таким образом, вы можете ссылаться на другую страницу сайта, указывая страницу и имя якоря:

```
Remember, you can always <a href="page1.html#Topic1">look back in
the First Topic</a> for  supporting information.
<!-- Помните, вы всегда можете <a href="page1.html#Topic1">вернуть-
ся к первой теме</a> для получения дополнительной информации. -->
```

6.3. Ссылка на страницу того же сайта

Вы можете использовать относительный путь для ссылок на страницы того же сайта:

```
<a href="/example">Text Here</a> <!-- Текст здесь -->
```

Приведенный выше пример перейдет к файлу example в корневом каталоге (`/`) сервера.

Если эта ссылка находится на `http://example.com`, следующие две ссылки приведут пользователя в одно и то же место:

```
<a href="/page">Text Here</a> <!-- Текст здесь -->
<a href="http://example.com/page">Text Here</a> <!-- Текст здесь -->
```

Обе ссылки перейдут к файлу page в корневом каталоге `example.com`.

6.4. Ссылка, которая набирает номер

Если значение атрибута `href` начинается с `tel:`, ваше устройство наберет номер при нажатии на ссылку. Это работает на мобильных устройствах или на компьютерах / планшетах с программным обеспечением, таким как Skype или FaceTime, которое может совершать звонки:

```
<a href="tel:11234567890">Call us</a> <!-- Позвоните нам -->
```

Большинство устройств и программ запросят у пользователя подтверждение номера, который они собираются набрать.

6.5. Открытие ссылки в новой вкладке / окне

```
<a href="example.com" target="_blank">Text Here</a> <!-- Текст
здесь -->
```

Атрибут `target` указывает, где открыть ссылку. Установив его значение в `_blank`, вы указываете браузеру открыть ее в новой вкладке или окне (в зависимости от предпочтений пользователя).

> ПРЕДУПРЕЖДЕНИЕ ОБ УЯЗВИМОСТИ БЕЗОПАСНОСТИ!
>
> Использование `target="_blank"` дает открываемому сайту частичный доступ к объекту `window.opener` через JavaScript, что позволяет этой странице затем изменять `window.opener.location` вашей страницы и потенциально перенаправлять пользователей на сайты с вредоносным ПО или фишингом.
>
> При использовании этого для страниц, которые вы не контролируете, добавьте `rel="noopener"` к вашей ссылке, чтобы предотвратить отправку объекта `window.opener` с запросом.
>
> В настоящее время Firefox не поддерживает `noopener`, поэтому для максимального эффекта вам нужно использовать `rel="noopener noreferrer"`.

6.6. Ссылка, которая выполняет JavaScript

Просто используйте протокол `javascript:`, чтобы выполнить текст как JavaScript, вместо открытия его как обычной ссылки:

```
<a href="javascript:myFunction();">Run Code</a> <!-- Выполнить код
-->
```

Вы также можете достичь того же результата, используя атрибут onclick:

```
<a href="#" onclick="myFunction(); return false;">Run Code</a> <!-- Выполнить код -->
```

Здесь return false; необходим для предотвращения прокрутки страницы вверх при нажатии на ссылку #. Убедитесь, что весь код, который вы хотите выполнить, находится перед ним, так как возврат остановит выполнение дальнейшего кода.

Также стоит отметить, что вы можете добавить восклицательный знак ! после хештега, чтобы предотвратить прокрутку страницы вверх. Это работает, потому что любой недопустимый слаг (slug) заставит ссылку не прокручивать страницу, так как она не сможет найти элемент, на который ссылается (элемент с id="!"). Вы также можете использовать любой недопустимый слаг (например #scrollsNowhere) для достижения того же эффекта. В этом случае return false; не требуется:

```
<a href="#!" onclick="myFunction();">Run Code</a> <!-- Выполнить код -->
```

> Стоит ли использовать это?
>
> Ответ почти всегда — нет. Выполнение JavaScript встроенным способом, как в этом примере, является плохой практикой. Рассмотрите использование чистых JavaScript-решений, которые ищут элемент на странице и привязывают к нему функцию.
>
> Также подумайте, является ли этот элемент действительно кнопкой, а не ссылкой. Если это так, используйте <button>.

6.7. Ссылка, которая запускает почтовый клиент

Базовое использование

Если значение атрибута href начинается с mailto:, при нажатии на ссылку попытается открыться почтовый клиент:

```
<a href="mailto:example@example.com">Send email</a> <!-- Отправить письмо -->
```

Это установит адрес example@example.com в качестве получателя нового письма.

Cc и Bcc

Вы также можете добавить адреса для получателей в копии (cc) или скрытой копии (bcc), используя следующий синтаксис:

```
<a  href="mailto:example@example.com?cc=john@example.com&bcc=jane@
example.com">Send email</a>
<!-- Отправить письмо -->
```

Тема и текст письма

Вы можете заполнить тему и текст нового письма:

```
<a  href="mailto:example@example.com?subject=Example+subject&body=
Message+text">Send email</a>
<!-- Отправить письмо -->
```

Эти значения должны быть закодированы в URL.

Нажатие на ссылку с `mailto:` запустит почтовый клиент по умолчанию, указанный вашей операционной системой, или предложит выбрать клиент. Не все параметры, указанные после адреса получателя, поддерживаются во всех почтовых клиентах.

Глава 7.
Списки

HTML предлагает три способа создания списков: упорядоченные списки, неупорядоченные списки и списки описаний. Упорядоченные списки используют порядковые номера для указания порядка элементов, неупорядоченные используют определенные символы, такие как маркеры, для перечисления элементов без указания порядка, а списки описаний используют отступы для перечисления элементов вместе с их дочерними элементами. В этой главе объясняются реализация и комбинирование этих списков в HTML-разметке.

7.1. Упорядоченный список

Упорядоченный список можно создать с помощью тега `<ol>`, а каждый элемент списка — с помощью тега `<li>`, как показано в примере ниже:

```
<ol>
    <li>Item</li>
    <li>Another Item</li>
    <li>Yet Another Item</li>
</ol>
```

Это создаст нумерованный список (по умолчанию):

1. Item
2. Another Item
3. Yet Another Item

Ручное изменение номеров

Есть несколько способов изменить номера, отображаемые в элементах упорядоченного списка. Первый способ — задать начальный номер с помощью атрибута `start`. Список начнется с указанного числа и будет продолжать увеличиваться на единицу, как обычно:

```
<ol start="3">
    <li>Item</li>
    <li>Some Other Item</li>
    <li>Yet Another Item</li>
</ol>
```

Это создаст нумерованный список (по умолчанию):

4. Item
5. Some Other Item
6. Yet Another Item

Вы также можете явно задать определенный номер для конкретного элемента списка. Последующие элементы списка будут продолжать увеличиваться на единицу, начиная с этого значения и игнорируя текущее значение родительского списка:

```
<li value="7"></li>
```

Стоит отметить, что использование атрибута `value` непосредственно на элементе списка позволяет переопределить существующую нумерацию упорядоченного списка, начав ее с меньшего значения. Например, если родительский список уже достиг значения 7, а встретился элемент со значением 4, то этот элемент будет отображаться как 4, и нумерация продолжится с этого значения.

```
<ol start="5">
    <li>Item</li>
    <li>Some Other Item</li>
    <li value="4">A Reset Item</li>
    <li>Another Item</li>
    <li>Yet Another Item</li>
</ol>
```

Пример выше создаст список с нумерацией: 5, 6, 4, 5, 6 — начиная снова с числа, меньшего, чем предыдущее, и дублируя номер 6 в списке.

Примечание: атрибуты `start` и `value` принимают только числа — даже если упорядоченный список настроен на отображение римских цифр или букв.

Версия ≥ 5

Вы можете изменить порядок нумерации на обратный, добавив атрибут `reversed` в элемент `<ol>`:

```
<ol reversed>
    <li>Item</li>
    <li>Some Other Item</li>
    <li value="4">A Reset Item</li>
    <li>Another Item</li>
    <li>Yet Another Item</li>
</ol>
```

Обратная нумерация полезна, если вы постоянно добавляете элементы в список, например, новые эпизоды подкастов или презентации, и хотите, чтобы самые свежие элементы отображались первыми.

Изменение типа нумерации

Вы можете легко изменить тип нумерации, отображаемый в маркерах элементов списка, с помощью атрибута `type`:

```
<ol type="1|a|A|i|I">
```

Тип	Описание	Примеры
1	Значение по умолчанию — десятичные числа	1, 2, 3, 4
a	Алфавитный порядок (строчные буквы)	a, b, c, d
A	Алфавитный порядок (заглавные буквы)	A, B, C, D
i	Римские цифры (строчные)	i, ii, iii, iv
I	Римские цифры (заглавные)	I, II, III, IV

Используйте `<ol>` для отображения списка элементов, где порядок элементов важен и должен быть подчеркнут. Если изменение порядка элементов не делает список некорректным, следует использовать `<ul>`.

7.2. Неупорядоченный список

Неупорядоченный список можно создать с помощью тега `<ul>`, а каждый элемент списка — с помощью тега `<li>`, как показано в примере ниже:

```
<ul>
    <li>Item</li>
    <li>Another Item</li>
    <li>Yet Another Item</li>
</ul>
```

Это создаст маркированный список (по умолчанию):

- Item
- Another Item
- Yet Another Item

Используйте `<ul>` для отображения списка элементов, где порядок элементов неважен. Если изменение порядка элементов делает список некорректным, следует использовать `<ol>`.

7.3. Вложенные списки

Вы можете вкладывать списки для представления подэлементов элемента списка:

```
<ul>
    <li>item 1</li>
```

```
    <li>item 2
        <ul>
            <li>sub-item 2.1</li>
            <li>sub-item 2.2</li>
        </ul>
    </li>
    <li>item 3</li>
</ul>
```

- item 1
- item 2
    - sub-item 2.1
    - sub-item 2.2
- item 3

Вложенный список должен быть дочерним элементом элемента `<li>`.

Вы также можете вкладывать списки разных типов:

```
<ol>
    <li>Hello, list!</li>
    <li>
        <ul>
            <li>Hello, nested list!</li>
        </ul>
    </li>
</ol>
```

7.4. Список описаний (description list)

Список описаний (description list) (или *список определений* (definition list), как он назывался до HTML5) можно создать с помощью элемента `<dl>`. Он состоит из групп «имя-значение», где имя задается в элементе `<dt>`, а значение — в элементе `<dd>`:

```
<dl>
    <dt>name 1</dt>
    <dd>value for 1</dd>
    <dt>name 2</dt>
    <dd>value for 2</dd>
</dl>
```

Группа «имя-значение» может содержать несколько имен и/или несколько значений (которые представляют альтернативы):

```
<dl>
    <dt>name 1</dt>
    <dt>name 2</dt>
    <dd>value for 1 and 2</dd>
    <dt>name 3</dt>
    <dd>value for 3</dd>
    <dd>value for 3</dd>
</dl>
```

Глава 8.
Таблицы

Элемент HTML <table> позволяет авторам веб-страниц отображать табличные данные (такие как текст, изображения, ссылки, другие таблицы и т. д.) в виде двумерной таблицы с строками и столбцами ячеек.

8.1. Простая таблица

```
<table>
    <tr>
        <th>Heading 1/Column 1</th> <!-- Заголовок 1/Столбец 1 -->
        <th>Heading 2/Column 2</th> <!-- Заголовок 2/Столбец 2 -->
    </tr>
    <tr>
       <td>Row 1 Data Column 1</td> <!-- Данные строки 1, столбец 1 -->
       <td>Row 1 Data Column 2</td> <!-- Данные строки 1, столбец 2 -->
    </tr>
    <tr>
       <td>Row 2 Data Column 1</td> <!-- Данные строки 2, столбец 1 -->
       <td>Row 2 Data Column 2</td> <!-- Данные строки 2, столбец 2 -->
    </tr>
</table>
```

Эта таблица состоит из трех строк (<tr>): одна строка с заголовками (<th>) и две строки с данными (<td>). Элементы <th> используются для заголовков таблицы, а элементы <td> — для данных. Внутрь <td> и <th> можно поместить любой контент.

Heading 1/Column 1	Heading 2/Column 2
Row 1 Data Column 1	Row 1 Data Column 2
Row 2 Data Column 1	Row 2 Data Column 2

8.2. Объединение столбцов или строк

Ячейки таблицы могут объединять несколько столбцов или строк с помощью атрибутов colspan и rowspan. Эти атрибуты могут применяться к элементам <th> и <td>:

```
<table>
    <tr>
        <td>row 1 col 1</td> <!-- строка 1, столбец 1 -->
        <td>row 1 col 2</td> <!-- строка 1, столбец 2 -->
        <td>row 1 col 3</td> <!-- строка 1, столбец 3 -->
    </tr>
    <tr>
       <td colspan="3">This second row spans all three columns</td>
        <!-- Эта строка объединяет все три столбца -->
    </tr>
    <tr>
        <td rowspan="2">This cell spans two rows</td>
<!-- Эта ячейка объединяет две строки -->
        <td>row 3 col 2</td> <!-- строка 3, столбец 2 -->
        <td>row 3 col 3</td> <!-- строка 3, столбец 3 -->
    </tr>
    <tr>
        <td>row 4 col 2</td> <!-- строка 4, столбец 2 -->
        <td>row 4 col 3</td> <!-- строка 4, столбец 3 -->
    </tr>
</table>
```

Результат:

<table>
<tr><td>row 1 col 1</td><td>row 1 col 2</td><td>row 1 col 3</td></tr>
<tr><td colspan="3">This second row spans all three columns</td></tr>
<tr><td rowspan="2">This cell spans two rows</td><td>row 3 col 2</td><td>row 3 col 3</td></tr>
<tr><td>row 4 col 2</td><td>row 4 col 3</td></tr>
</table>

Обратите внимание, что не следует проектировать таблицы, в которых строки и столбцы перекрываются, так как это недопустимо в HTML, и результат может отображаться по-разному в различных браузерах.

`rowspan` — неотрицательное целое число, указывающее количество строк, которые объединяет ячейка. Значение по умолчанию — 1. Значение 0 означает, что ячейка будет расширяться от текущей строки до последней строки таблицы (`<thead>`, `<tbody>` или `<tfoot>`).

`colspan` — неотрицательное целое число, указывающее количество столбцов, которые объединяет текущая ячейка. Значение по умолчанию — 1. Значение 0 означает, что ячейка будет расширяться от текущего столбца до последнего столбца группы столбцов `<colgroup>`, в которой определена ячейка.

8.3. Группы столбцов

Иногда может потребоваться применить стили к столбцу или группе столбцов. Либо, для семантических целей, может потребоваться сгруппировать столбцы вместе. Для этого используются элементы `<colgroup>` и `<col>`.

Необязательный тег `<colgroup>` позволяет группировать столбцы. Элементы `<colgroup>` должны быть дочерними элементами `<table>` и должны располагаться после любых элементов `<caption>` и перед любым содержимым таблицы (например, `<tr>`, `<thead>`, `<tbody>` и т. д.);

```
<table>
    <colgroup span="2"></colgroup>
    <colgroup span="2"></colgroup>
    ...
</table>
```

Необязательный тег `<col>` позволяет ссылаться на отдельные столбцы или диапазон столбцов без применения логической группировки. Элементы `<col>` необязательны, но если они присутствуют, они должны находиться внутри элемента `<colgroup>`:

```
<table>
    <colgroup>
        <col id="MySpecialColumn" />
        <col />
    </colgroup>
    <colgroup>
        <col class="CoolColumn" />
        <col class="NeatColumn" span="2" />
    </colgroup>
    ...
</table>
```

К элементам `<colgroup>` и `<col>` можно применять следующие CSS-стили:

- `border`
- `background`
- `width`
- `visibility`
- `display` (например, `display: none`)
    - `display: none;` фактически удалит столбцы из отображения, и таблица будет отображаться так, как если бы этих ячеек не существовало.

Для получения дополнительной информации см. HTML5 Tabular data (https://html.spec.whatwg.org/multipage/tables.html).



8.4. Таблица с элементами thead, tbody, tfoot и caption

HTML также предоставляет таблицы с элементами `<thead>`, `<tbody>`, `<tfoot>` и `<caption>`. Эти дополнительные элементы полезны для добавления семантической ценности таблицам и для применения отдельных CSS-стилей.

При печати таблицы, которая не помещается на одну страницу, большинство браузеров повторяют содержимое `<thead>` на каждой странице.

Существует определенный порядок, который должен соблюдаться, и следует учитывать, что не каждый элемент располагается так, как можно было бы ожидать. Следующий пример демонстрирует, как должны располагаться эти четыре элемента:

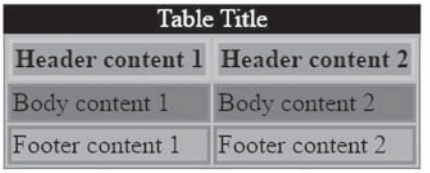
```
<table>
    <caption>Table Title</caption> <!-- Заголовок таблицы является
первым дочерним элементом таблицы -->
   <thead> <!--=======================| thead следует после caption
|-->
        <tr>
            <th>Header content 1</th>
            <th>Header content 2</th>
        </tr>
    </thead>
    <tbody> <!--=======================| tbody следует после thead
|-->
        <tr>
            <td>Body content 1</td>
            <td>Body content 2</td>
        </tr>
    </tbody>
     <tfoot><!--| tfoot может быть размещен до или после tbody,
но не внутри группы tbody. |-->
             <!--| Независимо от расположения tfoot в разметке
он отображается внизу. |-->
        <tr>
            <td>Footer content 1</td>
            <td>Footer content 2</td>
```

```
        </tr>
    </tfoot>
</table>
```

Результаты следующего примера демонстрируются дважды — первая таблица не имеет никаких стилей, вторая таблица имеет несколько примененных CSS-свойств: `background-color`, `color` и `border*`. Стили предоставлены в качестве визуального руководства и не являются важным аспектом рассматриваемой темы.

Table Title

Header content 1	Header content 2
Body content 1	Body content 2
Footer content 1	Footer content 2



Элемент	Стили
`<caption>`	Желтый текст на черном фоне.
`<thead>`	Жирный текст на фиолетовом фоне.
`<tbody>`	Текст на синем фоне.
`<tfoot>`	Текст на зеленом фоне.
`<th>`	Оранжевые границы.
`<td>`	Красные границы.

8.5. Область заголовка

Элементы `<th>` часто используются для указания заголовков строк и столбцов таблицы, например:

```
<table>
    <thead>
        <tr>
            <td></td>
            <th>Column Heading 1</th> <!-- Заголовок столбца 1 -->
```

```
            <th>Column Heading 2</th> <!-- Заголовок столбца 2 -->
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Row Heading 1</th> <!-- Заголовок строки 1 -->
            <td></td>
            <td></td>
        </tr>
        <tr>
            <th>Row Heading 2</th> <!-- Заголовок строки 2 -->
            <td></td>
            <td></td>
        </tr>
    </tbody>
</table>
```

Это можно улучшить для доступности с помощью атрибута scope. Пример выше будет изменен следующим образом:

```
<table>
    <thead>
        <tr>
            <td></td>
             <th scope="col">Column Heading 1</th> <!-- Заголовок
столбца 1 -->
             <th scope="col">Column Heading 2</th> <!-- Заголовок
столбца 2 -->
        </tr>
    </thead>
    <tbody>
        <tr>
           <th scope="row">Row Heading 1</th> <!-- Заголовок стро-
ки 1 -->
            <td></td>
            <td></td>
        </tr>
        <tr>
           <th scope="row">Row Heading 1</th> <!-- Заголовок стро-
ки 1 -->
            <td></td>
            <td></td>
        </tr>
```

```
    </tbody>
</table>
```

scope — перечисляемый атрибут, и это означает, что он может иметь значение из определенного набора возможных значений. Этот набор включает:

- col
- row
- colgroup
- rowgroup

Ссылки:

- MDN: th element (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/th#attr-scope).



- W3C: H63 (https://www.w3.org/TR/WCAG20-TECHS/H63.html).



Глава 9.
Комментарии

Как и в других языках программирования и разметки, комментарии в HTML предоставляют другим разработчикам информацию, связанную с разработкой, без влияния на пользовательский интерфейс. Однако, в отличие от других языков, комментарии в HTML могут использоваться для указания элементов, предназначенных только для Internet Explorer. В этой главе объясняется, как писать комментарии в HTML, и рассматривается их функциональное применение.

9.1. Создание комментариев

Комментарии в HTML могут использоваться для оставления заметок себе или другим разработчикам о конкретной части кода. Они начинаются с `<!--` и заканчиваются `-->`, например:

```
<!-- Я HTML-комментарий! -->
```

Они могут быть встроены в другие элементы:

```
<h1>This part will be displayed <!-- while this will not be displayed
-->.</h1>
<!-- Эта часть будет отображена, а эта — нет. -->
```

Комментарии также могут занимать несколько строк, чтобы предоставить больше информации:

```
<!-- Это многострочный HTML-комментарий.
     Все, что находится здесь, не будет отображено браузером.
     Вы можете "закомментировать" целые секции HTML-кода.
-->
```

Однако они не могут находиться внутри другого HTML-тега, например:

```
<h1 <!-- testAttribute="something" -->>This will not work</h1>
<!-- Это не сработает -->
```

Это создает невалидный HTML, так как весь блок `<h1 <!-- testAttribute="something" -->` будет считаться одним открывающим

тегом h1 с недопустимой информацией внутри, за которым следует ничего не делающая закрывающая скобка > .

Для совместимости с инструментами, которые пытаются парсить HTML как XML или SGML, тело комментария не должно содержать два дефиса подряд --.

9.2. Комментирование пробелов между строчными элементами

Строчные элементы, такие как span или a, могут включать до одного пробела перед и после них в документе. Чтобы избежать слишком длинных строк в разметке (которые трудно читать) и непреднамеренных пробелов (которые влияют на форматирование), пробелы можно закомментировать:

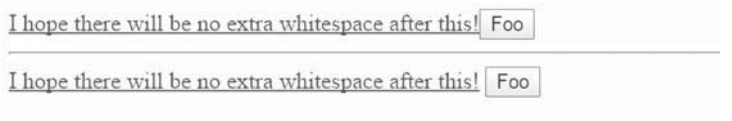
```
<!-- Используйте HTML-комментарий, чтобы убрать символ новой стро-
ки ниже: -->
<a href="#">I hope there will be no extra whitespace after this!</
a><!--
--><button>Foo</button>
```

Попробуйте убрать комментарий между строчными элементами, и между ними появится пробел. Иногда добавление пробела является желаемым.

Пример кода:

```
<!-- Используйте HTML-комментарий, чтобы убрать символ новой стро-
ки ниже: -->
<a href="#">I hope there will be no extra whitespace after this!</
a><!--
--><button>Foo</button>
<hr>
<!-- Without it, you can notice a small formatting difference: -->
<!-- Без комментария можно заметить небольшую разницу в форматиро-
вании: -->
<a href="#">I hope there will be no extra whitespace after this!</
a>
<button>Foo</button>
```

Результат:



Глава 10.
Классы и идентификаторы

Параметр	Описание
`class`	указывает класс элемента (не уникальный).
`id`	указывает идентификатор элемента (уникальный в пределах одного контекста).

Классы и идентификаторы упрощают обращение к HTML-элементам из скриптов и таблиц стилей. Атрибут `class` может использоваться на одном или нескольких тегах и применяется в CSS для стилизации. Однако идентификаторы (`id`) предназначены для ссылки на один элемент, что означает, что один и тот же идентификатор не должен использоваться дважды. Идентификаторы обычно применяются в JavaScript и внутренних ссылках документа, а их использование в CSS не рекомендуется. В этой главе приведены полезные объяснения и примеры правильного использования атрибутов `class` и `id` в HTML.

10.1. Назначение элемента класса

Классы являются идентификаторами для элементов, которым они назначены. Используйте атрибут `class`, чтобы назначить класс элементу:

```
<div class="example-class"></div>
```

Чтобы назначить элементу несколько классов, разделите их имена пробелами:

```
<div class="class1 class2"></div>
```

Использование классов в CSS

Классы можно использовать для стилизации определенных элементов без изменения всех элементов этого типа. Например, эти два элемента `span` могут иметь совершенно разное оформление:

```
<span></span>
<span class="special"></span>
```

Классы с одинаковыми именами могут быть назначены любому количеству элементов на странице, и все они получат стили, связанные с этим классом. Это всегда будет верно, если вы не укажете элемент в CSS.

Например, у нас есть два элемента, оба с классом `highlight`:

```
<div class="highlight">Lorem ipsum</div>
<span class="highlight">Lorem ipsum</span>
```

Если наш CSS выглядит следующим образом, то зеленый цвет будет применен к тексту внутри обоих элементов:

```
.highlight { color: green; }
```

Однако если мы хотим стилизовать только `div` с классом `highlight`, то можем добавить специфичность следующим образом:

```
div.highlight { color: green; }
```

Тем не менее при стилизации с помощью CSS рекомендуется использовать только классы (например, `.highlight`), а не элементы с классами (например `div.highlight`).

Как и с любыми другими селекторами, классы могут быть вложенными:

```
.main .highlight { color: red; } /* Комбинатор потомков */
.footer > .highlight { color: blue; } /* Дочерний комбинатор */
```

Вы также можете объединять селекторы классов, чтобы выбрать элементы, имеющие комбинацию нескольких классов. Например, если у нас есть такой HTML:

```
<div class="special left menu">This text will be pink</div>
<!—Этот текст будет розовым-->
```

...и мы хотим окрасить этот текст в розовый цвет, мы можем сделать следующее в CSS:

```
.special.left.menu { color: pink; }
```

10.2. Назначение элемента идентификатора

Атрибут `id` элемента является уникальным идентификатором в пределах всего документа. Его цель — однозначно идентифицировать элемент при создании ссылок (с использованием якоря), написании скриптов или стилизации (с помощью CSS):

```
<div id="example-id"></div>
```

Не следует использовать два элемента с одинаковым идентификатором в одном документе, даже если атрибуты назначены разным типам элементов. Например, следующий код некорректен:

```
<div id="example-id"></div>
<span id="example-id"></span>
```

Браузеры постараются отобразить этот код, но при стилизации с помощью CSS или добавлении функциональности с использованием JavaScript могут возникнуть непредвиденные проблемы.

Для ссылки на элементы по их идентификатору в CSS используйте префикс #:

```
#example-id { color: green; }
```

Чтобы перейти к элементу с определенным идентификатором на странице, добавьте # с именем элемента в URL:

```
http://example.com/about#example-id
```

Эта функция поддерживается большинством браузеров и не требует дополнительного JavaScript или CSS для работы.

10.3. Допустимые значения

Для идентификатора

Версия ≥ 5

Единственные ограничения для значения `id`:

1. Оно должно быть уникальным в документе.
2. Оно не должно содержать пробелов.
3. Оно должно содержать хотя бы один символ.

Таким образом, значение может состоять из цифр, одной цифры, знаков препинания, специальных символов и т. д. Главное — без пробелов.

Примеры допустимых значений:

```
<div id="container"> ... </div>
<div id="999"> ... </div>
<div id="#%LV-||"> ... </div>
<div id="____V"> ... </div>
<div id="⌘¬"> ... </div>
<div id="♥"> ... </div>
<div id="{}"> ... </div>
<div id="©"> ... </div>
<div id="♤₩¤☆€~¥"> ... </div>
```

Пример недопустимого значения:

```
<div id=" «> ... </div>
```

Также недопустимо использовать одинаковые идентификаторы в одном документе:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

Версия ≤ 4.01

Значение `id` должно начинаться с буквы, за которой могут следовать только:

- буквы (`A-Z/a-z`);
- цифры (`0–9`);
- дефисы («`–`»);
- подчеркивания («`_`»);
- двоеточия («`:`»);
- точки («`.`»).

Из примеров выше для HTML5 только один является допустимым:

```
<div id="container"> ... </div>
```

Также допустимы:

```
<div id="sampletext"> ... </div>
<div id="sample-text"> ... </div>
<div id="sample_text"> ... </div>
<div id="sample:text"> ... </div>
<div id="sample.text"> ... </div>
```

Если значение не начинается с буквы (в верхнем или нижнем регистре), оно недопустимо.

Для класса

Правила для классов практически такие же, как и для `id`. Разница в том, что значения классов не должны быть уникальными в документе.

Например, если это недопустимо для одного документа:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

...то это вполне допустимо:

```
<div class="results"> ... </div>
<div class="results"> ... </div>
```

Важное замечание: как значения ID и Class обрабатываются вне HTML

Помните, что правила и примеры выше применимы в контексте HTML.

Использование цифр, знаков препинания или специальных символов в значении `id` или `class` может вызвать проблемы в других контекстах, таких как CSS, JavaScript и регулярные выражения.

Например, хотя следующий `id` допустим в HTML5:

```
<div id="9lions"> ... </div>
```

...он недопустим в CSS:

> 4.1.3 Символы и регистр
>
> В CSS идентификаторы (включая имена элементов, классы и идентификаторы в селекторах) могут содержать только символы [a-zA-Z0-9] и символы ISO 10646 U+00A0 и выше, а также дефис (-) и подчеркивание (_); они не могут начинаться с цифры, двух дефисов или дефиса, за которым следует цифра.
>
> Ссылка на источник (https://www.w3.org/TR/CSS2/syndata.html#characters).



В большинстве случаев вы можете экранировать символы в контекстах, где они имеют ограничения или специальное значение.

Ссылки на W3C

- 3.2.5.1 Атрибут id (https://html.spec.whatwg.org/multipage/dom.html#the-id-attribute).



- 3.2.5.7 Атрибут class (https://html.spec.whatwg.org/multipage/dom.html#classes).



- 6.2 Базовые типы SGML (https://www.w3.org/TR/html4/types.html#type-id).



10.4. Проблемы, связанные с дублированием идентификаторов

Наличие более одного элемента с одинаковым идентификатором (ID) — это сложная для диагностики проблема. Обычно HTML-парсер все равно пытается отобразить страницу. Как правило, ошибка не возникает, но страница может начать работать некорректно.

В этом примере:

```
<div id="aDiv">a</div>
<div id="aDiv">b</div>
```

CSS-селекторы все еще работают:

```
#aDiv {
    color: red;
}
```

...но JavaScript не может обработать оба элемента:

```
var html = document.getElementById(“aDiv").innerHTML;
```

В этом случае переменная HTML содержит только содержимое первого элемента `div ("a")`.

Глава 11.
Атрибуты данных (Data Attributes)

Значение	Описание
`somevalue`	Указывает значение атрибута (в виде строки)

11.1. Поддержка в старых браузерах

Атрибуты данных (data attributes) были введены в HTML5, который поддерживается всеми современными браузерами. Однако старые браузеры, выпущенные до HTML5, не распознают атрибуты данных.

Тем не менее, согласно спецификациям HTML, атрибуты, которые не распознаются браузером, должны оставаться без изменений, и браузер просто проигнорирует их при отображении страницы.

Веб-разработчики использовали этот факт для создания нестандартных атрибутов, которые не являются частью спецификаций HTML. Например, атрибут `value` в следующей строке считается нестандартным, поскольку спецификации для тега `<img>` не включают атрибут `value`, и он не является глобальным атрибутом:

```
<img src="sample.jpg" value="test" />
```

Это означает, что хотя атрибуты данных не поддерживаются в старых браузерах, они все равно работают, и вы можете устанавливать и получать их с помощью тех же универсальных методов JavaScript `setAttribute` и `getAttribute`. Однако вы не сможете использовать новое свойство `dataset`, которое поддерживается только в современных браузерах.

11.2. Использование атрибутов данных (Data Attribute Use)

Атрибуты данных HTML5 `data-*` предоставляют удобный способ хранения данных в HTML-элементах. Хранимые данные можно читать или изменять с помощью JavaScript:

```
<div data-submitted="yes" class="user_profile">
    … some content …
</div>
```

Структура атрибута данных выглядит как `data-*`, то есть имя атрибута данных следует после части `data-`. Используя это имя, можно получить доступ к атрибуту. Данные в строковом формате (включая JSON) могут быть сохранены с использованием атрибута `data-*`.

Глава 12.
Связывание ресурсов

Атрибут	Описание
`charset`	Указывает кодировку символов связанного документа.
`crossorigin`	Определяет, как элемент обрабатывает запросы между источниками (cross-origin).
`href`	Указывает местоположение связанного документа.
`hreflang`	Указывает язык текста в связанном документе.
`media`	Указывает, на каком устройстве будет отображаться связанный документ, часто используется для выбора таблиц стилей в зависимости от устройства.
`rel`	Обязательный. Указывает отношение между текущим документом и связанным документом.
`rev`	Указывает отношение между связанным документом и текущим документом.
`sizes`	Указывает размер связанного ресурса. Используется только при `rel="icon"`.
`target`	Указывает, где должен быть загружен связанный документ.
`type`	Указывает тип медиа связанного документа.
`integrity`	Указывает хеш (sha256, sha384 или sha512) в формате base64 для проверки подлинности связанного ресурса.

Хотя многие скрипты, иконки и таблицы стилей могут быть написаны непосредственно в HTML-разметке, лучшей практикой и более эффективным подходом является включение этих ресурсов в отдельные файлы и их связывание с вашим документом. В этой главе рассматривается подключение внешних ресурсов, таких как таблицы стилей и скрипты, в HTML-документ.

12.1. JavaScript

Синхронная загрузка

```
<script src="path/to.js"></script>
```

Стандартной практикой является размещение тегов `<script>` непосредственно перед закрывающим тегом `</body>`. Загрузка скриптов в послед-

нюю очередь позволяет быстрее отображать визуальные элементы вашего сайта и предотвращает попытки JavaScript взаимодействовать с элементами, которые еще не загрузились.

Асинхронная загрузка

```
<script src="path/to.js" async></script>
```

Альтернативный подход, когда загружаемый код JavaScript не требуется для инициализации страницы, — это асинхронная загрузка, которая ускоряет загрузку страницы. Использование атрибута async позволяет браузеру загружать содержимое скрипта параллельно и, как только оно будет полностью загружено, прерывает парсинг HTML для выполнения JavaScript.

Отложенная загрузка

```
<script src="path/to.js" defer></script>
```

Отложенные скрипты похожи на асинхронные, с тем отличием, что их выполнение начнется только после полного парсинга HTML. Отложенные скрипты гарантированно загружаются в порядке их объявления, так же как и синхронные скрипты.

Элемент <noscript>

```
<noscript>JavaScript disabled</noscript>
```

Элемент <noscript> определяет контент, который будет отображаться, если у пользователя отключен JavaScript или если браузер не поддерживает использование скриптов. Тег <noscript> может быть размещен как в <head>, так и в <body>.

12.2. Внешняя таблица стилей CSS

```
<link rel="stylesheet" href="path/to.css" type="text/css">
```

Стандартной практикой является размещение тегов <link> для CSS внутри тега <head> в верхней части HTML-документа. Таким образом, CSS загружается первым и применяется к странице по мере ее загрузки, вместо того чтобы показывать неоформленный HTML до загрузки CSS. В HTML5 атрибут type не обязателен, так как HTML5 по умолчанию поддерживает CSS:

```
<link rel="stylesheet" href="path/to.css" type="text/css">
```

и

```
<link rel="stylesheet" href="path/to.css">
```

... делают одно и то же в HTML5.

Другой, хотя и менее распространенный подход, — использование директивы `@import` внутри CSS. Например:

```
<style type="text/css">
@import(«path/to.css")
</style>
```

или

```
<style>
@import(«path/to.css")
</style>
```

12.3. Favicon

```
<link rel="icon" type="image/png" href="/favicon.png">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
```

Используйте MIME-тип `image/png` для PNG-файлов и `image/x-icon` для иконок (`*.ico`). Другие варианты решения этого вопроса рассматриваются на Stack Overflow (https://stackoverflow.com/questions/1344122/favicon-png-vs-favicon-ico-why-should-i-use-png-instead-of-ico).



Файл с именем `favicon.ico` в корне вашего сайта обычно загружается и применяется автоматически, без необходимости использования тега `<link>`. Однако если этот файл изменяется, браузеры могут медленно и неохотно обновлять его кеш.

12.4. Альтернативные таблицы стилей

```
<link rel="alternate stylesheet" href="path/to/style.css"
title="yourTitle">
```

Некоторые браузеры позволяют применять альтернативные таблицы стилей, если они предоставлены. По умолчанию они не применяются, но обычно их можно изменить через настройки браузера:

> Firefox позволяет пользователю выбрать таблицу стилей через меню «Вид» > «Стиль страницы». Internet Explorer также поддерживает эту функцию (начиная с IE 8), доступную через «Вид» > «Стиль страницы» (по крайней мере, в IE 11). Chrome требует установки расширения для использования

> этой функции (по состоянию на версию 48). Веб-страница также может предоставлять собственный интерфейс для переключения стилей.
>
> Источник: https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/rel/alternate_stylesheet.



12.5. Подсказки для ресурсов: dns-prefetch, prefetch, prerender

Предварительное соединение (Preconnect)

Отношение `preconnect` похоже на `dns-prefetch`, так как оно также разрешает DNS. Однако оно также выполняет TCP-рукопожатие и, при необходимости, TLS-согласование. Это экспериментальная функция:

```
<link rel="preconnect" href="URL">
```

Предварительное разрешение DNS (DNS-Prefetch)

Информирует браузеры о необходимости разрешить DNS для URL, чтобы все ресурсы с этого URL загружались быстрее:

```
<link rel="dns-prefetch" href="URL">
```

Предварительная загрузка (Prefetch)

Информирует браузеры о необходимости предварительно загрузить ресурс, чтобы он мог быть загружен быстрее:

```
<link rel="prefetch" href="URL">
```

DNS-Prefetch разрешает только доменное имя, тогда как `prefetch` загружает/сохраняет указанные ресурсы.

Предварительный рендеринг (Prerender)

Информирует браузеры о необходимости загрузить и отрендерить URL в фоновом режиме, чтобы он мог быть мгновенно доставлен пользователю при переходе на этот URL. Это экспериментальная функция:

```
<link rel="prerender" href="URL">
```

12.6. Атрибут media в теге <link>

```
<link rel="stylesheet" href="test.css" media="print">
```

Атрибут `media` указывает, для какого типа медиа должна использоваться таблица стилей. Использование значения `print` означает, что эта таблица стилей будет применяться только для печати страниц.

Значение этого атрибута может быть любым из типов медиа (аналогично медиазапросам в CSS).

12.7. Предыдущая и следующая страницы

Когда страница является частью серии статей, можно использовать `prev` и `next` для указания страниц, которые идут до и после текущей:

```
<link rel="prev" href="http://stackoverflow.com/documentation/
java/topics">
<link rel="next" href="http://stackoverflow.com/documentation/
css/topics">
```

12.8. Веб-фид (Web Feed)

Используйте атрибут `rel="alternate"` для обеспечения возможности обнаружения ваших Atom/RSS-фидов.

```
<link rel="alternate" type="application/atom+xml" href="http://
example.com/feed.xml" />
<link rel="alternate" type="application/rss+xml" href="http://
example.com/feed.xml" />
```

Глава 13.
Включение JavaScript-кода в HTML

Атрибут	Описание
`src`	Указывает путь к файлу JavaScript. Это может быть относительный или абсолютный URL.
`type`	Указывает MIME-тип. Этот атрибут обязателен в HTML4, но необязателен в HTML5.
`async`	Указывает, что скрипт должен выполняться асинхронно (только для внешних скриптов). Этот атрибут не требует значения (кроме XHTML).
`defer`	Указывает, что скрипт должен выполняться после завершения разбора страницы (только для внешних скриптов). Этот атрибут не требует значения (кроме XHTML).
`charset`	Указывает кодировку символов, используемую во внешнем файле скрипта, например UTF-8.
`crossorigin`	Определяет, как элемент обрабатывает запросы между источниками (cross-origin requests).
`nonce`	Криптографический одноразовый код (nonce), используемый для проверок политики безопасности контента (CSP3).

13.1. Обработка отключенного JavaScript

Возможно, что клиентский браузер не поддерживает JavaScript или выполнение JavaScript отключено, возможно, по соображениям безопасности. Чтобы сообщить пользователям, что на странице должен выполняться скрипт, можно использовать тег `<noscript>`. Содержимое `<noscript>` отображается, если JavaScript отключен для текущей страницы:

```
<script>
    document.write(«Hello, world!");
</script>
<noscript>This browser does not support Javascript.</noscript>
<!-- Этот браузер не поддерживает JavaScript.-->
```

13.2. Ссылка на внешний файл JavaScript

```
<script src="example.js"></script>
```

Атрибут `src` работает так же, как атрибут `href` в ссылках: можно указать абсолютный или относительный URL. В примере выше указан файл, находящийся в той же директории, что и HTML-документ. Обычно этот тег добавляется внутри тегов `<head>` в верхней части HTML-документа.

13.3. Прямое включение JavaScript-кода

Вместо ссылки на внешний файл можно включить JavaScript-код непосредственно в HTML:

```
<script>
    // JavaScript-код
</script>
```

13.4. Включение JavaScript-файла с асинхронным выполнением

```
<script type="text/javascript" src="URL" async></script>
```

Глава 14.
Использование HTML с CSS

CSS предоставляет стили для HTML-элементов на странице. Встроенное стилирование (inline styling) предполагает использование атрибута `style` в тегах, однако этот метод крайне не рекомендуется. Внутренние таблицы стилей (internal stylesheets) используют тег `<style>` и применяются для объявления правил для определенных частей страницы. Внешние таблицы стилей (external stylesheets) могут быть подключены с помощью тега `<link>`, который загружает внешний файл CSS и применяет его правила к документу. В этой главе рассматриваются все три метода подключения.

14.1. Использование внешних таблиц стилей

Используйте атрибут `link` в разделе `head` документа:

```
<head>
    <link rel="stylesheet" type="text/css" href="stylesheet.css">
</head>
```

Также можно использовать таблицы стилей, предоставляемые веб-сайтами через сеть доставки контента (CDN, Content Delivery Network), например Bootstrap:

```
<head>
    <link rel="stylesheet"
          href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYi
iSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
```

Как правило, поддержку CDN для фреймворка можно найти на его официальном сайте.

14.2. Внутренние таблицы стилей

Вы также можете включать CSS-элементы внутри документа с помощью тега `<style>`:

```
<head>
    <style type="text/css">
        body {
            background-color: gray;
        }
    </style>
</head>
```

В программе можно использовать несколько внутренних таблиц стилей:

```
<head>
    <style type="text/css">
        body {
            background-color: gray;
        }
    </style>
    <style type="text/css">
        p {
            background-color: blue;
        }
    </style>
</head>
```

14.3. Встроенные стили

Вы можете стилизовать конкретный элемент с помощью атрибута `style`:

```
<span style="color: red">This text will appear in red.</span>
<!-- Этот текст будет отображаться красным. -->
```

Примечание: старайтесь избегать этого метода — цель CSS заключается в разделении содержимого и представления.

14.4. Несколько таблиц стилей

Можно загружать несколько таблиц стилей:

```
<head>
    <link rel="stylesheet" type="text/css" href="general.css">
    <link rel="stylesheet" type="text/css" href="specific.css">
</head>
```

Обратите внимание, что более поздние файлы и объявления будут переопределять более ранние. Например, если `general.css` содержит:

```
body {
    background-color: red;
}
```

а `specific.css` содержит:

```
body {
    background-color: blue;
}
```

...то при использовании обоих файлов фон документа будет синим.

Глава 15.
Изображения

Параметр	Описание
`src`	Указывает URL изображения
`srcset`	Изображения для использования в различных ситуациях (например, на экранах с высоким разрешением, маленьких мониторах и т. д.)
`sizes`	Размеры изображения между точками останова (breakpoints)
`crossorigin`	Указывает, как элемент обрабатывает запросы с кросс-доменными источниками
`usemap`	Имя карты изображения, которую следует использовать
`ismap`	Указывает, является ли изображение серверной картой изображения
`alt`	Альтернативный текст, который должен отображаться, если изображение не может быть загружено
`width`	Указывает ширину изображения (необязательно)
`height`	Указывает высоту изображения (необязательно)

15.1. Создание изображения

Чтобы добавить изображение на страницу, используйте тег `<img>`.

Теги изображений (`<img>`) не имеют закрывающего тега. Два основных атрибута, которые вы указываете для тега `<img>`, — это `src` (источник изображения) и `alt` (альтернативный текст, описывающий изображение):

```
<img src="images/hello.png" alt="Hello World">
```

Вы также можете использовать изображения из веб-URL:

```
<img src="https://i.stack.imgur.com/ALgZi.jpg?s=48&g=1"
alt="StackOverflow user Caleb Kleveter">
```

Примечание: изображения технически не вставляются в HTML-страницу, а связываются с ней. Тег <img> создает пространство для отображения ссылки на изображение.

Также можно встраивать изображения напрямую в страницу с использованием `base64`:

```
<img src="..." alt="Hello World">
```

Совет: чтобы связать изображение с другим документом, просто вложите тег `<img>` внутрь тегов `<a>`.

15.2. Выбор альтернативного текста

Альтернативный текст (alt-text) используется программами чтения с экрана для пользователей с нарушениями зрения, а также поисковыми системами. Поэтому важно писать качественный альтернативный текст для ваших изображений.

Текст должен выглядеть корректно, даже если заменить изображение его атрибутом `alt`. Например:

```
<!-- Неправильно -->
<img src="anonymous.png" alt="Anonymous user avatar"/> An
anonymous user wrote:
<blockquote>Lorem ipsum dolor sed.</blockquote>
<a href="https://google.com/"><img src="edit.png" alt="Edit
icon"/></a> /
<a href="https://google.com/"><img src="delete.png" alt="Delete
icon"/></a>
```

Без изображений это будет выглядеть так:

> Anonymous user avatar An anonymous user wrote:
>
> > Lorem ipsum dolor sed.
>
> Edit icon / Delete icon

Чтобы исправить это:

- Удалите альтернативный текст для аватара. Это изображение добавляет информацию для зрячих пользователей (легко узнаваемый значок, показывающий, что пользователь анонимный), но эта информация уже доступна в тексте.
- Удалите слово «icon» из альтернативного текста для значков. Знание о том, что это значок, не помогает передать его фактическое назначение.

```
<!-- Правильно -->
<img src="anonymous.png" alt=""/> An anonymous user wrote:
<blockquote>Lorem ipsum dolor sed.</blockquote>
<a href="https://google.com/"><img src="edit.png" alt="Edit"/></
a> /
<a href="https://google.com/"><img src="delete.png"
alt="Delete"/></a>
```

Результат:

> An anonymous user wrote:
>
> Lorem ipsum dolor sed.
>
> Edit / Delete

Примечание: существует семантическая разница между включением пустого атрибута `alt` и его полным отсутствием. Пустой атрибут `alt` указывает, что изображение не является ключевой частью контента (как в данном случае — это просто дополнительное изображение, которое не обязательно для понимания остального). Однако отсутствие атрибута `alt` указывает, что изображение является ключевой частью контента, и для него просто нет текстового эквивалента.

15.3. Адаптивное изображение с использованием атрибута srcset

Использование **`srcset`** ***с*** **`sizes`**

```
<img sizes="(min-width: 1200px) 580px,
    (min-width: 640px) 48vw,
    98vw"
  srcset="img/hello-300.jpg 300w,
    img/hello-600.jpg 600w,
    img/hello-900.jpg 900w,
    img/hello-1200.jpg 1200w"
  src="img/hello-900.jpg" alt="hello">
```

`sizes` работают как медиазапросы, описывая, сколько места изображение занимает во `viewport`.

- Если `viewport` больше `1200px`, изображение будет ровно `580px` (например, наш контент центрирован в контейнере шириной `1200px`. Изображение занимает половину минус отступы).
- Если `viewport` между `640px` и `1200px`, изображение занимает `48%` `viewport` (например, изображение масштабируется вместе со страницей и занимает половину ширины `viewport` минус отступы).
- Если `viewport` любого другого размера, в нашем случае меньше `640px`, изображение занимает `98%` `viewport` (например, изображение масштабируется вместе со страницей и занимает всю ширину `viewport` минус отступы). Условие медиазапроса должно быть опущено для последнего элемента.

`srcset` просто сообщает браузеру, какие изображения у нас есть и какие у них размеры.

- `img/hello-300.jpg` имеет ширину `300px`,
- `img/hello-600.jpg` имеет ширину `600px`,
- `img/hello-900.jpg` имеет ширину `900px`,
- `img/hello-1200.jpg` имеет ширину `1200px`.

***Использование* `srcset` *без* `sizes`**

```
<img src="img/hello-300.jpg" alt="hello"
  srcset="img/hello-300.jpg 1x,
    img/hello-600.jpg 2x,
    img/hello-1200.jpg 3x">
```

`srcset` предоставляет список доступных изображений с указанием плотности пикселей (device-pixel ratio).

- Если плотность пикселей равна 1, используйте `img/hello-300.jpg`.
- Если плотность пикселей равна 2, используйте `img/hello-600.jpg`.
- Если плотность пикселей равна 3, используйте `img/hello-1200.jpg`.

`src` — это всегда обязательный источник изображения. В случае использования с `srcset` `src` будет служить резервным изображением, если браузер не поддерживает `srcset`.

15.4. Адаптивное изображение с использованием элемента <picture>

Код

```
<picture>
    <source media="(min-width: 600px)" srcset="large_image.jpg">
    <source media="(min-width: 450px)" srcset="small_image.jpg">
    <img src="default_image.jpg" style="width:auto;">
</picture>
```

Использование

Для отображения разных изображений в зависимости от ширины экрана необходимо включить все изображения с использованием тега `<source>` внутри тега `<picture>`, как показано в примере выше.

Результат

- На экранах с шириной `>600px` отображается `large_image.jpg`.
- На экранах с шириной `>450px` отображается `small_image.jpg`.
- На экранах с другой шириной отображается `default_image.jpg`.

Глава 16.
Карты изображений (Image Maps)

Теги / Атрибуты	Значения
`<img>`	Ниже приведены атрибуты, специфичные для карт изображений, которые используются с тегом `<img>`. Обычные атрибуты `<img>` также применяются.
`usemap`	Имя карты с добавленным символом решетки. Например, для карты с именем `name="map"` изображение должно иметь атрибут `usemap="#map"`.
`<map>`	
`name`	Имя карты для ее идентификации. Используется вместе с атрибутом `usemap` изображения.
`<area>`	Ниже приведены атрибуты, специфичные для тега `<area>`. Если указан атрибут `href`, делающий `<area>` ссылкой, то `<area>` также поддерживает все атрибуты тега `<a>`, за исключением `ping`.
`alt`	Альтернативный текст, который отображается, если изображения не поддерживаются. Этот атрибут необходим только в том случае, если на `<area>` также установлен атрибут `href`.
`coords`	Координаты, определяющие область для выбора. Если `shape="polygon"`, этот атрибут должен быть установлен в виде списка пар `"x, y"`, разделенных запятыми (например, `shape="polygon" coords="x1, y1, x2, y2, x3, y3, ..."`). Если `shape="rectangle"`, этот атрибут должен быть установлен в виде `left`, `top`, `right`, `bottom`. Если `shape="circle"`, этот атрибут должен быть установлен в виде `centerX`, `centerY`, `radius`.
`href`	URL-адрес гиперссылки, если он указан. Если он опущен, то `<area>` не будет представлять собой гиперссылку.
`shape`	Форма области (`<area>`). Может быть установлена как `default` для выбора всего изображения (атрибут `coords` не требуется), `circle` или `circ` для круга, `rectangle` или `rect` для прямоугольника, и `polygon` или `poly` для многоугольной области, заданной угловыми точками.

16.1. Введение в карты изображений

Описание

Карта изображения (image map) — это изображение с кликабельными областями, которые обычно действуют как гиперссылки.

Изображение определяется тегом `<img>`, а карта — тегом `<map>` с тегами `<area>`, обозначающими каждую кликабельную область. Используйте атрибуты `usemap` и `name`, чтобы связать изображение и карту.

Базовый пример

Для создания карты изображения, в которой каждая из фигур на изображении ниже будет кликабельной:



...код будет следующим:

```
<img src="http://jaced.com/blogpix/2007/trisquarecircle/002.gif"
usemap="#shapes">
<map name="shapes">
    <area shape="polygon" coords="79,6,5,134,153,134">
    <area shape="rectangle" coords="177,6,306,134">
    <area shape="circle" coords="397,71,65">
</map>
```

Вы увидите, что браузер распознает области, когда курсор превращается в указатель.

Глава 17.
Элементы управления вводом

Параметр	Описание
`class`	Указывает класс элемента ввода.
`id`	Указывает идентификатор элемента ввода.
`type`	Определяет тип элемента управления вводом. Допустимые значения: `hidden`, `text`, `tel`, `url`, `email`, `password`, `date`, `time`, `number`, `range`, `color`, `checkbox`, `radio`, `file`, `submit`, `image`, `reset`, `button`. По умолчанию используется `text`, если тип не указан, указан неверно или не поддерживается браузером.
`name`	Указывает имя элемента ввода.
`disabled`	Булево значение, указывающее, что элемент ввода должен быть отключен. Отключенные элементы нельзя редактировать, они не отправляются при отправке формы и не могут получать фокус.
`checked`	Если значение атрибута `type` — `radio` или `checkbox`, наличие этого булева атрибута указывает, что элемент выбран по умолчанию; в противном случае он игнорируется.
`multiple`	HTML5: Указывает, что можно передать несколько файлов или значений (применяется только к элементам типа `file` и `email`).
`placeholder`	HTML5: Подсказка для пользователя, что можно ввести в поле. Текст подсказки не должен содержать символов перевода строки.
`autocomplete`	HTML5: Указывает, может ли значение элемента быть автоматически заполнено браузером.
`readonly`	Булево значение, указывающее, что элемент ввода нельзя редактировать. Только для чтения элементы все равно отправляются при отправке формы, но не могут получать фокус. HTML5: Этот атрибут игнорируется, если значение атрибута `type` установлено как `hidden`, `range`, `color`, `checkbox`, `radio`, `file` или `button`.
`required`	HTML5: Указывает, что значение должно быть указано или элемент должен быть отмечен для отправки формы.
`alt`	Альтернативный текст для изображений, если они не отображаются.
`autofocus`	Элемент `<input>` должен получить фокус при загрузке страницы.
`value`	Указывает значение элемента `<input>`.
`step`	Определяет допустимые числовые интервалы. Работает с типами ввода: `number`, `range`, `date`, `datetime-local`, `month`, `time`, `week`.

Элементы ввода — ключевой компонент интерактивных веб-систем. Это HTML-элементы, предназначенные для получения определенного типа данных от пользователей. Различные типы элементов ввода позволяют регулировать формат вводимых данных и обеспечивают безопасность при вводе паролей.

17.1. Текстовое поле

Самый базовый тип ввода и тип по умолчанию, если тип не указан. Этот тип определяет однострочное текстовое поле, в котором автоматически удаляются переносы строк. Все остальные символы могут быть введены. Элементы `<input>` используются внутри элемента `<form>` для объявления элементов управления, позволяющих пользователям вводить данные.

Синтаксис

```
<input type="text">
```

или (без указания типа, с использованием атрибута по умолчанию):

```
<input>
```

Ширина текстового поля по умолчанию составляет 20 символов. Это можно изменить, указав значение для атрибута `size`:

```
<input type="text" size="50">
```

Атрибут `size` отличается от установки ширины с помощью CSS. Использование ширины определяет конкретное значение (в пикселях, процентах от родительского элемента и т. д.), которое поле ввода должно занимать. Использование `size` рассчитывает ширину на основе используемого шрифта и ширины символов.

Примечание: использование атрибута `size` не ограничивает количество символов, которые можно ввести в поле, а только определяет его ширину. Для ограничения длины см. раздел «Валидация ввода».

Текстовое поле позволяет вводить только одну строку текста. Если вам нужно многострочное текстовое поле для большого объема текста, используйте элемент `<textarea>`.

17.2. Флажки и переключатели

Обзор

Флажки (checkboxes) и переключатели (radio buttons) создаются с помощью HTML-тега `<input>`, а их поведение определяется в спецификации HTML.

Самый простой флажок или переключатель — это элемент `<input>` с атрибутом `type`, установленным как `checkbox` или `radio` соответственно:

```
<input type="checkbox">
<input type="radio">
```

Одиночный флажок используется для двоичного выбора, например, для вопроса «да» или «нет». Флажки независимы, то есть пользователь может выбрать несколько вариантов в группе флажков. Другими словами, выбор одного флажка не снимает выбор с других флажков в группе.

Переключатели обычно используются в группах (если переключатель не сгруппирован с другим, вероятно, вы хотели использовать флажок). Группа переключателей идентифицируется использованием одного и того же атрибута `name` для всех переключателей в группе. Выбор переключателей взаимоисключающий, то есть пользователь может выбрать только один вариант из группы. Когда один переключатель выбран, любой другой переключатель с тем же именем, который был выбран ранее, становится невыбранным.

Пример:

```
<input type="radio" name="color" id="red" value="#F00">
<input type="radio" name="color" id="green" value="#0F0">
<input type="radio" name="color" id="blue" value="#00F">
```

При отображении переключатели выглядят как кружок (не выбран) или заполненный кружок (выбран). Флажки выглядят как квадрат (не выбран) или заполненный квадрат (выбран). В зависимости от браузера и операционной системы квадрат иногда имеет закругленные углы.

Атрибуты

Флажки и переключатели имеют несколько атрибутов для управления их поведением.

`value`

Как и у любого другого элемента ввода, атрибут `value` указывает строковое значение, связанное с элементом при отправке формы. Однако флажки и переключатели отличаются тем, что если значение опущено, по умолчанию отправляется значение `on`, а не пустое значение. Атрибут `value` не влияет на внешний вид элемента.

`checked`

Атрибут `checked` указывает начальное состояние флажка или переключателя. Это булев атрибут, который может быть опущен.

Каждый из следующих примеров эквивалентен и определяет выбранный переключатель:

```
<input checked>
<input checked="">
<input checked="checked">
<input checked="ChEcKeD">
```

Отсутствие атрибута `checked` — единственный допустимый синтаксис для невыбранного элемента:

```
<input type="radio">
<input type="checkbox">
```

При сбросе формы (`<form>`) флажки и переключатели возвращаются в состояние, определенное атрибутом `checked`.

Доступность

Метки

Чтобы предоставить контекст для элементов и показать пользователям, для чего нужен каждый элемент, всем им необходимо предоставить метку. Это можно сделать с помощью элемента `<label>`, который оборачивает элемент. Также это делает метку кликабельной, чтобы можно было выбрать соответствующий элемент.

Пример:

```
<label>
    <input type="radio" name="color" value="#F00">
    Красный
</label>
```

...или с использованием элемента `<label>` с атрибутом `for`, установленным в значение атрибута `id` элемента:

```
<input type="checkbox" name="color" value="#F00" id="red">
<label for="red">Красный</label>
```

Группы элементов

Поскольку каждый переключатель влияет на другие в группе, обычно предоставляется метка или контекст для всей группы переключателей.

Чтобы предоставить метку для всей группы, переключатели должны необходимо включить в элемент `<fieldset>` с элементом `<legend>` внутри него.

Пример:

```
<fieldset>
    <legend>Theme color:</legend>
```

```
    <!-- Цвет темы -->
    <p>
        <input type="radio" name="color" id="red" value="#F00">
        <label for="red">Красный</label>
    </p>
    <p>
        <input type="radio" name="color" id="green" value="#0F0">
        <label for="green">Зеленый</label>
    </p>
    <p>
        <input type="radio" name="color" id="blue" value="#00F">
        <label for="blue">Синий</label>
    </p>
</fieldset>
```

Флажки также могут быть сгруппированы аналогичным образом, с использованием `<fieldset>` и `<legend>` для идентификации группы связанных флажков. Однако помните, что флажки не должны иметь одно и то же имя, так как они не являются взаимоисключающими. В противном случае приведет к отправке нескольких значений для одного ключа, и не все серверные языки обрабатывают это одинаково (неопределенное поведение). Каждый флажок должен иметь уникальное имя или использовать квадратные скобки ([ ]), чтобы указать, что форма должна отправить массив значений для этого ключа. Выбор метода зависит от того, как вы планируете обрабатывать данные формы на стороне клиента или сервера. Также следует делать легенду короткой, так как некоторые комбинации браузеров и программ чтения с экрана читают легенду перед каждым полем ввода в `<fieldset>`.

17.3. Валидация ввода

Валидация ввода HTML выполняется браузером автоматически на основе специальных атрибутов элемента `<input>`. Она может частично или полностью заменить валидацию ввода на JavaScript. Однако такую валидацию можно обойти с помощью специально сформированных HTTP-запросов, поэтому она не заменяет валидацию на стороне сервера. Валидация происходит только при попытке отправить форму, поэтому все ограниченные элементы ввода должны находиться внутри формы, чтобы валидация сработала (если только вы не используете JavaScript). Учтите, что элементы ввода, которые отключены (disabled) или доступны только для чтения (readonly), не вызывают валидацию.

Некоторые новые типы ввода (например, `email`, `url`, `tel`, `date` и многие другие) автоматически проверяются и не требуют дополнительных ограничений для валидации.

Версия ≥ 5

Обязательное поле

Используйте атрибут `required`, чтобы указать, что поле должно быть заполнено для прохождения валидации:

```
<input required>
```

Минимальная / Максимальная длина

Используйте атрибуты `minlength` и `maxlength`, чтобы указать требования к длине. Большинство браузеров предотвращают ввод более `max` символов в поле, что не позволяет пользователю сделать запись недействительной еще до попытки отправки формы:

```
<input minlength="3">
<input maxlength="15">
<input minlength="3" maxlength="15">
```

Указание диапазона

Используйте атрибуты `min` и `max`, чтобы ограничить диапазон чисел, которые пользователь может ввести в поле типа `number` или `range`:

```
Marks: <input type="number" size="6" name="marks" min="0" max="100" />
<!-- Оценки -->
Subject Feedback: <input type="range" size="2" name="feedback"
min="1" max="5" />
<!--Отзыв по предмету -->
```

Версия ≥ 5

Соответствие шаблону

Для более точного контроля используйте атрибут `pattern`, чтобы указать регулярное выражение, которому должно соответствовать значение для прохождения валидации. Также можно указать атрибут `title`, который будет включен в сообщение об ошибке, если поле не пройдет валидацию:

```
<input pattern="\d*" title="Numbers only, please.">
<!–Только цифры, пожалуйста -->
```

Вот пример сообщения, которое отображается в Google Chrome версии 51 при попытке отправить форму с недопустимым значением в этом поле:



Не все браузеры отображают сообщение для недопустимых шаблонов, хотя большинство современных браузеров поддерживают эту функцию.

Проверьте последнюю информацию о поддержке на CanIUse и реализуйте соответственно (https://caniuse.com/input-pattern).



Версия ≥ 5

Прием определенных типов файлов

Для полей ввода типа `file` можно указать, что принимаются только определенные типы файлов, такие как видео, изображения, аудио, определенные расширения файлов или типы медиа. Например:

```
<input type="file" accept="image/*" title="Допускаются только изображения">
```

Несколько значений можно указать через запятую, например:

```
<input type="file" accept="image/*,.rar,application/zip">
```

Примечание: добавление атрибута `novalidate` к элементу `<form>` или атрибута `formnovalidate` к кнопке отправки предотвращает валидацию элементов формы. Например:

```
<form>
    <input type="text" name="name" required>
    <input type="email" name="email" required>
    <input pattern="\d*" name="number" required>
    <input type="submit" value="Publish “> <!-- Опубликовать-->
    <!-- форма будет проверена -->
    <input type="submit" value="Save" formnovalidate> <!--
Сохранить -->
    <!-- форма НЕ будет проверена -->
</form>
```

В этом примере форма содержит поля, обязательные для «публикации» черновика, но не обязательные для «сохранения» черновика.

17.4. Цвет

Версия ≥ 5

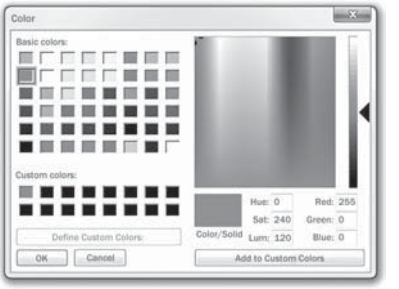
```
<input type="color" name="favcolor" value="#ff0000">
```

В поддерживающих браузерах элемент `<input>` с атрибутом `type`, установленным как `color`, создает кнопку, цвет которой соответствует значению

атрибута value (по умолчанию черный, если значение не указано или имеет недопустимый шестнадцатеричный формат).



При нажатии на эту кнопку открывается виджет выбора цвета операционной системы (на экране компьютера вы увидите цветное изображение), который позволяет пользователю выбрать цвет.



Для браузеров, которые не поддерживают этот тип ввода, используется обычное поле типа text.



17.5. Пароль

```
<input type="password" name="password">
```

Элемент <input> с атрибутом type, установленным как password, создает однострочное текстовое поле, похожее на поле типа text, за исключением того, что текст не отображается при вводе:

```
<input type="password" name="password" placeholder="Password">
<!-- Пароль -->
```

Текст-заполнитель (placeholder) отображается обычным текстом и автоматически заменяется, когда пользователь начинает ввод:



Примечание: некоторые браузеры и системы изменяют поведение поля пароля, чтобы отображать последний введенный символ в течение короткого времени, например:



17.6. Файл

```
<input type="file" name="fileSubmission">
```

Поля ввода файлов позволяют пользователям выбрать файл из локальной файловой системы для использования на текущей странице. Если они ис-

пользуются вместе с элементом <form>, они могут позволить пользователям загружать файлы на сервер.

Следующий пример позволяет пользователям выбрать файл из файловой системы и загрузить его на сервер с помощью скрипта upload_file.php:

```
<form action="upload_file.php" method="post" enctype="multipart/
form-data">
    Select file to upload:
    <input type="file" name="fileSubmission" id="fileSubmission">
    <input type="submit" value="Upload your file" name="submit">
</form>
```

Несколько файлов

Добавление атрибута multiple позволяет пользователю выбрать несколько файлов:

```
<input type="file" name="fileSubmission" id="fileSubmission" multiple>
```

Прием определенных типов файлов

Атрибут accept указывает типы файлов, которые пользователь может выбрать. Например, .png, .gif, .jpeg:

```
<input type="file" name="fileSubmission" accept="image/x-png,image/
gif,image/jpeg">
```

17.7. Кнопка

```
<input type="button" value="Button text">
```

Кнопки могут использоваться для выполнения действий на странице без отправки формы. Вы также можете использовать элемент <button>, если вам нужна кнопка, которую легче стилизовать или которая может содержать другие элементы:

```
<button type="button">Текст кнопки</button>
```

Кнопки обычно используются с событием onclick:

```
<input type="button" onclick="alert('Hello, world!')" value="Click Me">
```

или

```
<button type="button" onclick="alert('Hello, world!')">Click Me</
button>
```

Атрибуты

`[name]`

Имя кнопки, которое отправляется с данными формы.

`[type]`

Тип кнопки.

Возможные значения:

`submit`: кнопка отправляет данные формы на сервер. Это значение по умолчанию, если атрибут не указан или изменен на пустое или недопустимое значение.

`reset`: кнопка сбрасывает все элементы управления к их начальным значениям.

`button`: кнопка не имеет стандартного поведения. Она может иметь клиентские скрипты, связанные с событиями элемента, которые срабатывают при наступлении событий.

`menu`: кнопка открывает всплывающее меню, определенное через соответствующий элемент.

`[value]`

Начальное значение кнопки.

Версия ≥ 5

Дополнительные атрибуты для кнопок отправки

Атрибут	Описание
`form`	Указывает ID формы, к которой принадлежит кнопка. Если не указано, кнопка принадлежит ближайшей родительской форме (если она существует).
`formaction`	Указывает, куда отправлять данные формы при отправке с использованием этой кнопки.
`formenctype`	Указывает, как должны кодироваться данные формы при отправке на сервер с использованием этой кнопки. Может использоваться только с `formmethod="post"`.
`formmethod`	Указывает HTTP-метод (`POST` или `GET`) для отправки данных формы с использованием этой кнопки.
`formnovalidate`	Указывает, что данные формы не должны проверяться при отправке.
`formtarget`	Указывает, где отображать ответ, полученный после отправки формы с использованием этой кнопки.

17.8. Отправка формы

```
<input type="submit" value="Отправить">
```

Кнопка отправки при нажатии создает кнопку, отправляющую форму, в которой она находится.

Вы также можете использовать элемент `<button>`, если вам нужна кнопка отправки, которую легче стилизовать или которая может содержать другие элементы:

```
<button type="submit">
    <img src="submit-icon.jpg" /> Отправить
</button>
```

17.9. Сброс формы

```
<input type="reset" value="Reset">
```

Кнопка сброса (reset) создает кнопку, которая при нажатии сбрасывает все элементы управления в форме к их начальным значениям.

- Текст в поле ввода будет сброшен на пустое значение или значение по умолчанию (указанное с помощью атрибута `value`).
- Любые выбранные пункты в меню выбора будут сняты, если они не имеют атрибута `selected`.
- Все флажки и переключатели будут сняты, если они не имеют атрибута `checked`.

Примечание: кнопка сброса должна находиться внутри формы или быть привязана к ней (с помощью атрибута `form`), чтобы иметь эффект. Кнопка сбросит только элементы внутри этой формы.

17.10. Скрытое поле (Hidden)

```
<input type="hidden" name="inputName" value="inputValue">
```

Скрытое поле не будет видно пользователю, но его значение будет отправлено на сервер при отправке формы.

17.11. Телефон

```
<input type="tel" value="+8400000000">
```

Элемент `<input>` с атрибутом `type`, установленным как `tel`, представляет собой однострочное текстовое поле для ввода номера телефона.

17.12. Электронная почта

Версия ≥ 5

Элемент `<input type="email">` используется для полей ввода, которые должны содержать адрес электронной почты:

```
<form>
    <label>E-mail: <label>
    <input type="email" name="email">
</form>
```

Адрес электронной почты может быть автоматически проверен при отправке в зависимости от поддержки браузера.

17.13. Число

Версия ≥ 5

```
<input type="number" value="0" name="quantity">
```

Элемент `<input>` с атрибутом `type`, установленным как `number`, представляет собой точный элемент управления для установки значения в виде строки, представляющей число.

Обратите внимание: это поле не гарантирует ввод корректного числа. Оно позволяет вводить любые символы, которые могут быть частью числа, например, пользователь сможет ввести значение вида `e1e-`,`0`.

17.14. Диапазон

Версия ≥ 5

```
<input type="range" min="" max="" step="" />
```

Элемент управления для ввода числа, точное значение которого неважно.

Атрибут	Описание	Значение по умолчанию
`min`	Минимальное значение диапазона	0
`max`	Максимальное значение диапазона	100
`step`	Шаг увеличения значения	1

17.15. Поиск

Версия ≥ 5

Тип ввода `search` используется для текстового поиска. В большинстве браузеров он добавляет значок лупы рядом с полем для текста:

```
<input type="search" name="googlesearch">
```

17.16. Изображение

```
<input type="image" src="img.png" alt="image_name" height="50px"
width="50px"/>
```

Необходимо использовать атрибут `src` для указания источника изображения и атрибут `alt` для указания альтернативного текста. Вы можете использовать атрибуты `height` и `width` для указания размера изображения в пикселях.

17.17. Неделя

Версия ≥ 5

```
<input type="week" />
```

В зависимости от поддержки браузера будет отображаться элемент управления для выбора номера года и недели без указания часового пояса.

17.18. URL

Версия ≥ 5

```
<input type="url" name="Homepage">
```

Этот тип ввода используется для полей, которые должны содержать URL-адрес.

В зависимости от поддержки браузера поле URL может быть автоматически проверено при отправке.

Некоторые смартфоны распознают тип `url` и добавляют «.com» на клавиатуре для удобства ввода.

17.19. Локальные дата и время

Версия ≥ 5

```
<input type="datetime-local" />
```

В зависимости от поддержки браузера будет отображаться элемент управления для выбора даты и времени.

17.20. Месяц

Версия ≥ 5

```
<input type="month" />
```

В зависимости от поддержки браузера будет отображаться элемент управления для выбора месяца.

17.21. Время

Версия ≥ 5

```
<input type="time" />
```

Тип ввода `time` указывает, что элемент принимает строку, представляющую время. Формат определен в RFC 3339 и должен быть частичным временем, например:

```
19:04:39
08:20:39.04
```

В настоящее время все версии Edge, Chrome, Opera и Chrome для Android поддерживают `type="time"`. Более новые версии Android Browser, начиная с 4.4, также поддерживают его. Safari для iOS предлагает частичную поддержку, не поддерживая атрибуты `min`, `max` и `step`.

17.22. Глобальные дата и время

Элемент `<input>` с атрибутом `type`, установленным как `"datetime"`, представляет собой элемент управления для установки значения в виде строки, представляющей глобальные дату и время (с информацией о часовом поясе):

```
<fieldset>
    <p><label>Meeting time: <input type="datetime" name="meeting.
start"></label></p>
    <!-- Время встречи -->
</fieldset>
```

Допустимые атрибуты:

- Глобальные атрибуты (global attributes)
- name

- disabled
- form
- type
- autocomplete
- autofocus
- list
- `min` и `max`
- `step` (число с плавающей точкой)
- readonly
- required
- value

17.23. Дата

Версия ≥ 5

```
<input type="date" />
```

На экране появится элемент управления для выбора даты. Этот тип ввода не поддерживается в Firefox и Internet Explorer.

Глава 18.
Формы

Атрибут	Описание
accept-charset	Указывает кодировку символов, которая будет использована при отправке формы.
action	Указывает, куда отправлять данные формы при ее отправке.
autocomplete	Указывает, должна ли форма иметь автозаполнение включенным или выключенным.
enctype	Указывает, как данные формы должны быть закодированы при отправке на сервер (только для method="post").
method	Указывает HTTP-метод, используемый при отправке данных формы (POST или GET).
name	Указывает имя формы.
novalidate	Указывает, что форма не должна проверяться при отправке.
target	Указывает, где отображать ответ, полученный после отправки формы.

Для группировки элементов ввода и отправки данных HTML использует элемент form, который объединяет элементы ввода и отправки. Эти формы обрабатывают отправку данных указанным методом на страницу, обрабатываемую сервером или обработчиком. В этой главе объясняется и демонстрируется использование HTML-форм для сбора и отправки входных данных.

18.1. Отправка данных

Атрибут action

Атрибут action определяет действие, которое должно быть выполнено при отправке формы. Обычно это приводит к выполнению скрипта, который собирает отправленную информацию и работает с ней. Если оставить его пустым, данные будут отправлены на тот же файл:

```
<form action="action.php">
```

Атрибут `method`

Атрибут `method` используется для определения HTTP-метода формы, который может быть либо `GET`, либо `POST`:

```
<form action="action.php" method="get">
<form action="action.php" method="post">
```

Метод `GET` чаще всего используется для получения данных, например, для получения записи по ее `ID` или имени или для отправки поискового запроса. Он добавляет данные формы к URL, указанному в атрибуте `action`:

```
www.example.com/action.php?firstname=Mickey&lastname=Mouse
```

Метод `POST` используется при отправке данных в скрипт. Он не добавляет данные формы к URL, а отправляет их с использованием тела запроса.

Для корректной отправки данных из формы необходимо указать атрибут `name`.

Пример отправки значения поля с именем `lastname`:

```
<input type="text" name="lastname" value="Mouse">
```

Дополнительные атрибуты

```
<form action="action.php" method="post" target="_blank" accept-
charset="UTF-8"
enctype="application/x-www-form-urlencoded" autocomplete="off"
novalidate>
<!-- элементы формы -->
</form>
```

18.2. Атрибут target в теге form

Атрибут `target` указывает имя или ключевое слово, определяющее, где отображать ответ, полученный после отправки формы. Этот атрибут определяет имя или ключевое слово для контекста просмотра (например, вкладка, окно или встроенный фрейм).

Пример формы с атрибутом `target`:

```
<form target="_blank">
```

Значения атрибута `target`

Значение	Описание
`_blank`	Ответ отображается в новом окне или вкладке.
`_self`	Ответ отображается в том же фрейме (по умолчанию).
`_parent`	Ответ отображается в родительском фрейме.
`_top`	Ответ отображается во всем теле окна.
`framename`	Ответ отображается во встроенном фрейме с указанным именем.

> **Примечание:** атрибут `target` был устаревшим в HTML 4.01. В HTML5 он поддерживается. Фреймы и фреймсеты не поддерживаются в HTML5, поэтому значения `_parent`, `_top` и `framename` теперь в основном используются с `iframe`.

18.3. Загрузка файлов

Изображения и файлы могут быть загружены/отправлены на сервер, если установить атрибут `enctype` тега `form` в значение `multipart/form-data`. Атрибут `enctype` указывает, как данные формы должны быть закодированы при отправке на сервер.

Пример:

```
<form method="post" enctype="multipart/form-data" action="upload.
php">
    <input type="file" name="pic" />
    <input type="submit" value="Upload" />
</form>
```

18.4. Группировка нескольких полей ввода

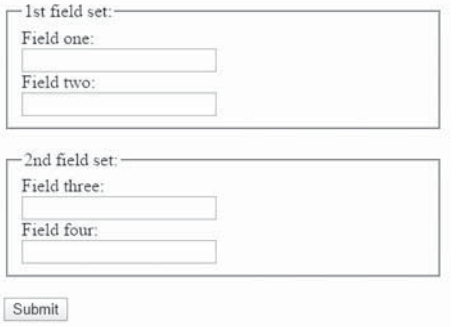
При проектировании формы может потребоваться сгруппировать несколько полей ввода для улучшения организации макета формы. Это можно сделать с помощью тега `<fieldset>`. Для каждого `<fieldset>` можно установить заголовок с помощью тега `<legend>`.

Пример:

```
<form>
    <fieldset>
        <legend>1st field set:</legend> <!-- Первая группа полей: -->
        Field one:<br>
```

```
        <input type="text"><br>
        Field two:<br>
        <input type="text"><br>
    </fieldset><br>
    <fieldset>
        <legend>2nd field set:</legend> <!-- Вторая группа полей: -->
        Field three:<br>
        <input type="text"><br>
        Field four:<br>
        <input type="text"><br>
    </fieldset><br>
    <input type="submit" value="Submit">
</form>
```

Результат:



Поддержка браузерами

Последние версии Chrome, IE, Edge, Firefox, Safari и Opera также поддерживают тег `<fieldset>`.

Глава 19.
Элемент Div

Элемент `div` в HTML является контейнерным элементом, который инкапсулирует другие элементы и может использоваться для группировки и разделения частей веб-страницы. Сам по себе элемент `div` не имеет какого-либо конкретного значения, но является мощным инструментом в веб-дизайне. В этой главе рассматриваются назначение и применение элемента `div`.

19.1. Основное использование

Элемент `<div>` обычно не имеет конкретного семантического значения сам по себе, просто представляя собой раздел, и чаще всего используется для группировки и инкапсуляции других элементов внутри HTML-документа, а также для отделения этих групп от другого контента. Таким образом, каждый `<div>` лучше всего описывается своим содержимым:

```
<div>
    <p>Hello! This is a paragraph.</p> <!-- Привет! Это абзац. -->
</div>
```

Элемент `div` обычно является блочным элементом (block-level element), и это означает, что он отделяет блок HTML-документа и занимает максимальную ширину страницы. Браузеры обычно применяют следующее правило CSS по умолчанию:

```
div {
    display: block;
}
```

> Консорциум Всемирной паутины (W3C) настоятельно рекомендует рассматривать элемент `div` как элемент последнего выбора, когда ни один другой элемент не подходит. Использование более подходящих элементов вместо `div` приводит к лучшей доступности для читателей и упрощению поддержки для авторов.

Например, запись в блоге следует размечать с помощью `<article>`, главу — с помощью `<section>`, навигационные элементы страницы — с помощью `<nav>`, а группу элементов управления формы — с помощью `<fieldset>`.

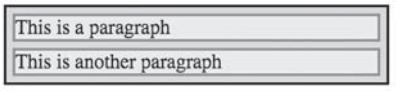
Элементы div могут быть полезны для стилистических целей или для обертывания нескольких абзацев внутри раздела, которые должны быть аннотированы схожим образом.

19.2. Вложенность

Распространенной практикой является размещение нескольких элементов внутри другого . Это обычно называется «вложенностью» элементов и позволяет дополнительно разделять элементы на подразделы или помогать разработчикам с CSS-стилизацией. Элемент используется для группировки двух элементов , каждый из которых содержит элемент <p>:

```
<div class="outer-div">
    <div class="inner-div">
        <p>This is a paragraph</p> <!-- Это абзац. -->
    </div>
    <div class="inner-div">
        <p>This is another paragraph</p> <!-- Это еще один абзац. -->
    </div>
</div>
```

Это приведет к следующему результату (для наглядности применены CSS-стили):



Вложенность строчных и блочных элементов

При вложении элементов следует помнить, что существуют строчные (inline) и блочные (block) элементы. В то время как блочные элементы добавляют разрыв строки в фоновом режиме, а это означает, что другие вложенные элементы автоматически отображаются на следующей строке, строчные элементы по умолчанию могут располагаться рядом друг с другом.

***Избегайте глубокой вложенности ***

Глубокая и часто используемая вложенность контейнеров указывает на плохой стиль кодирования.

Закругленные углы или подобные функции часто создают такой HTML-код. Для большинства современных браузеров существуют аналоги в CSS3. Старайтесь использовать как можно меньше HTML-элементов, чтобы увеличить соотношение контента к тегам и уменьшить время загрузки страницы, что приведет к лучшему ранжированию в поисковых системах.

Элемент div не должен быть вложен глубже, чем на шесть уровней.

Глава 20.
Секционные элементы

20.1. Элемент <nav>

Элемент <nav> предназначен в первую очередь для разделов, содержащих основные блоки навигации на веб-сайте. Это может включать ссылки на другие части веб-страницы (например, якоря для оглавления) или на другие страницы целиком.

Встроенные элементы

Следующий пример отображает набор встроенных гиперссылок:

```
<nav>
    <a href="https://google.com">Google</a>
    <a href="https://www.yahoo.com">Yahoo!</a>
    <a href="https://www.bing.com">Bing</a>
</nav>
```

Использование элементов списка при необходимости

Если контент представляет собой список элементов, используйте элементы списка для улучшения пользовательского опыта.

Обратите внимание на атрибут role="navigation", подробнее об этом ниже:

```
<nav role="navigation">
    <ul>
        <li><a href="https://google.com">Google</a></li>
        <li><a href="https://www.yahoo.com">Yahoo!</a></li>
        <li><a href="https://www.bing.com">Bing</a></li>
    </ul>
</nav>
```

Избегайте излишнего использования

Элементы <footer> могут содержать список ссылок на другие части сайта (FAQ, условия использования и т. д.). В этом случае достаточно элемента <footer>, и нет необходимости дополнительно оборачивать ссылки элементом <nav>:

```
<!-- Элемент <nav> не требуется внутри <footer> -->
<footer>
    <nav>
        <a href="#">...</a>
    </nav>
</footer>

<!-- Достаточно только элемента <footer> -->
<footer>
    <a href="#">...</a>
</footer>
```

Примечания

- Потомки элемента `<main>` не допускаются внутри элемента `<nav>`.
- Рекомендуется добавлять атрибут `role="navigation"` к элементу `<nav>`, чтобы помочь пользовательским агентам, которые не поддерживают HTML5, а также предоставить больше контекста для тех, которые поддерживают:

```
<nav role="navigation"><!-- ... --></nav>
```

Программы чтения с экрана (screen readers)

Пользовательские агенты, такие как программы чтения с экрана, могут интерпретировать элемент `<nav>` по-разному в зависимости от их требований.

- Они могут придать элементу `<nav>` более высокий приоритет при отображении страницы.
- Они могут задержать отображение элемента.
- Они могут адаптировать страницу определенным образом для удовлетворения потребностей пользователя. Например: увеличение текстовых ссылок внутри элементов `<nav>` для пользователей с нарушениями зрения.

Официальная спецификация HTML5 для элемента `<nav>` (https://html.spec.whatwg.org/multipage/sections.html#the-nav-element).



20.2. Элемент <article>

Элемент `<article>` содержит самодостаточный контент, такой как статьи, записи в блоге, комментарии пользователей или интерактивные виджеты, которые могут распространяться вне контекста страницы, например через RSS.

Когда элементы <article> вложены друг в друга, содержимое внутреннего элемента <article> должно быть связано с внешним элементом <article>.

Пример блога (раздел) с несколькими записями (статья) и комментариями (статья) может выглядеть следующим образом:

```
<section>
    <!-- Каждая отдельная запись в блоге — это <article> -->
    <article>
        <header>
            <h1>Blog Post</h1> <!-- Запись в блоге -->
            <time datetime="2016-03-13">13th March 2016</time>
<!-- 13 марта 2016 года -->
        </header>
        <p>The article element represents a self contained
article or document.</p> <!-- Элемент <article> представляет со-
бой самодостаточную статью или документ. -->
        <p>The section element represents a grouping of
content.</p> <!-- Элемент <section> представляет собой группу
контента. -->
        <section>
            <h2>Comments <small>relating to «Blog Post"</small></
h2> <!-- Комментарии, относящиеся к «Записи в блоге" -->
            <!-- Связанный комментарий также является
самодостаточной статьей -->
            <article id="user-comment-1">
                <p>Excellent!</p> <!-- Отлично! -->
                <footer><p>...</p><time>...</time></footer>
            </article>
        </section>
    </article>
    <!-- Повторить: <article> -->
</section>
<!-- Контент, не связанный с блогом или записями, должен
находиться вне раздела. -->
<footer>
    <p>This content should be unrelated to the blog.</p> <!--
Этот контент не должен быть связан с блогом. -->
</footer>
```

Избегайте излишнего использования

Когда основной контент страницы (за исключением заголовков, нижних колонтитулов, панелей навигации и т. д.) представляет собой просто одну группу элементов, вы можете опустить элемент <article> в пользу элемента <main>:

```
<article>
    <p>This doesn't make sense, this article has no real
`context`.</p>
    <!-- Это не имеет смысла, у этой статьи нет реального
контекста. -->
</article>
```

Вместо этого замените элемент `<article>` на элемент `<main>`, чтобы указать, что это основной контент страницы.

```
<main>
   <p>I'm the main content, I don't need to belong to an article.</p>
    <!-- Я основной контент, мне не нужно принадлежать статье. -->
</main>
```

Если вы используете другой элемент, убедитесь, что указали роль ARIA `<main>` для корректной интерпретации и отображения на различных устройствах и в браузерах, не поддерживающих HTML5:

```
<section role="main">
    <p>This section is the main content of this page.</p>
   <!-- Этот раздел является основным контентом этой страницы. -->
</section>
```

Примечания

- Потомки элемента `<main>` не допускаются внутри элемента `<article>`.

Официальная спецификация HTML5 для элемента `<article>` (https://html.spec.whatwg.org/multipage/sections.html#the-article-element).



20.3. Элемент <main>

Элемент `<main>` содержит основной контент вашей веб-страницы. Этот контент уникален для каждой отдельной страницы и не должен повторяться на других страницах сайта. Повторяющийся контент, такой как заголовки, нижние колонтитулы, навигация, логотипы и т. д., размещается за пределами этого элемента.

- Элемент `<main>` должен использоваться не более одного раза на одной странице.
- Элемент `<main>` не должен быть потомком элементов `<article>`, `<aside>`, `<footer>`, `<header>` или `<nav>`.

В следующем примере мы отображаем отдельную запись в блоге (и связанную с ней информацию, такую как ссылки и комментарии).

```
<body>
    <header>
        <nav>...</nav>
    </header>
    <main>
        <h1>Individual Blog Post</h1> <!-- Отдельная запись
в блоге -->
        <p>An introduction for the post.</p> <!-- Введение
к записи -->
        <article>
            <h2>References</h2> <!-- Ссылки -->
            <p>...</p>
        </article>
        <article>
            <h2>Comments</h2> <!-- Комментарии -->
            ...
        </article>
    </main>
    <footer>...</footer>
</body>
```

Запись в блоге содержится внутри элемента `<main>`, чтобы указать, что это основной контент для этой страницы (и, следовательно, уникален для всего сайта).

Элементы `<header>` и `<footer>` являются соседними элементами для `<main>`.

Примечания

- Спецификация HTML5 определяет элемент `<main>` как группирующий элемент, а не как секционный.
- Атрибуты роли ARIA: `main` (по умолчанию), `presentation`.
- Рекомендуется добавлять атрибут `role="main"` к другим элементам, которые предназначены для использования в качестве основного контента, чтобы помочь пользовательским агентам, не поддерживающим HTML5, а также предоставить больше контекста для тех, которые поддерживают.
- Элемент `<main>` по умолчанию имеет роль `main`, поэтому ее не нужно указывать явно.

Официальная спецификация HTML5 для элемента `<main>` (https://html.spec.whatwg.org/multipage/grouping-content.html#the-main-element).



20.4. Элемент <header>

Элемент <header> представляет вводный контент для своего ближайшего предка, являющегося секционным контентом или корневым секционным элементом. Обычно <header> содержит группу вводных или навигационных элементов.

Примечание: элемент <header> не является секционным контентом; он не создает новый раздел.

Примеры:

```
<header>
    <p>Welcome to...</p> <!-- Добро пожаловать в... -->
    <h1>Voidwars!</h1>
</header>
```

В этом примере элемент <article> содержит <header>:

```
<article>
    <header>
        <h1>Flexbox: The definitive guide</h1> <!-- Flexbox: Полное руководство -->
    </header>
     <p>The guide about Flexbox was supposed to be here, but it turned out Wes wasn't a Flexbox expert either.</p>
   <!-- Руководство по Flexbox должно было быть здесь, но оказалось, что Уэс тоже не эксперт по Flexbox. -->
</article>
```

Официальная спецификация HTML5 для элемента <header> (https://html.spec.whatwg.org/multipage/sections.html#the-header-element).



20.5. Элемент <footer>

Элемент <footer> содержит нижний колонтитул страницы.

Вот пример элемента <footer>, содержащего тег <p> для абзаца:

```
<footer>
    <p>All rights reserved</p> <!-- Все права защищены -->
</footer>
```

20.6. Элемент <section>

Элемент <section> представляет собой общий раздел для тематической группировки контента. Каждый раздел, как правило, должен быть идентифицирован с помощью заголовка, который является дочерним элементом раздела.

- Вы можете использовать элемент <section> внутри элемента <article> и наоборот.
- Каждый раздел должен иметь тему (заголовок, идентифицирующий эту область).
- Не используйте элемент <section> в качестве общего контейнера для стилизации. Если вам нужен контейнер для применения стилей, используйте .

В следующем примере мы отображаем отдельную запись в блоге с несколькими главами, где каждая глава представляет собой раздел (группу тематически связанного контента, который может быть идентифицирован с помощью заголовков в каждом разделе):

```
<article>
    <header>
        <h2>Blog Post</h2> <!-- Запись в блоге -->
    </header>
    <p>An introduction for the post.</p> <!-- Введение к записи -->
    <section>
        <h3>Chapter 1</h3> <!-- Глава 1 -->
        <p>...</p>
    </section>
    <section>
        <h3>Chapter 2</h3> <!-- Глава 2 -->
        <p>...</p>
    </section>
    <section>
        <h3>Comments</h3> <!-- Комментарии -->
        ...
    </section>
</article>
```

Примечание:

- Разработчики должны использовать элемент <article>, если имеет смысл синдицировать содержимое элемента.

Глава 21.
Навигационные панели

21.1. Базовая навигационная панель

Навигационные панели по сути представляют собой список ссылок, поэтому для их создания используются элементы `ul` и `li`:

```
<ul>
    <li><a href="#">Home</a></li> <!-- Главная -->
    <li><a href="#">About</a></li> <!-- О нас -->
    <li><a href="#">Contact</a></li> <!-- Контакты -->
</ul>
```

21.2. Навигационная панель в HTML5

Чтобы создать навигационную панель с использованием элемента `nav` в HTML5, поместите ссылки внутрь тега `nav`:

```
<nav>
    <a href="#">Home</a> <!-- Главная -->
    <a href="#">About</a> <!-- О нас -->
    <a href="#">Contact</a> <!-- Контакты -->
</nav>
```

Глава 22.
Элемент <label>

Атрибут	Описание
`for`	Ссылка на ID целевого элемента. Например: `for="surname"`.
`form`	HTML5, [Устаревший] Ссылка на форму, содержащую целевой элемент. Элементы `<label>` обычно располагаются внутри элемента `<form>`. Если указан `form="someFormId"`, это позволяет разместить `<label>` в любом месте документа.

22.1. Об элементе <label>

Элемент `<label>` используется для связи с элементом формы, выполняющим действие.

В контексте пользовательского интерфейса он упрощает выбор элементов, таких как радиокнопки (radio) или флажки (checkbox).

<label> как обертка

Элемент `<label>` может заключать в себе целевой элемент:

```
<label>
    <input type="checkbox" name="Cats">
    I like Cats! <!-- Мне нравятся кошки! -->
</label>
```

<label> как ссылка

Используя атрибут `for`, не обязательно помещать целевой элемент внутрь `<label>`, но значение `for` должно совпадать с его ID:

```
<input id="cats" type="checkbox" name="Cats">
<label for="cats">I like Cats!</label> <!-- Мне нравятся кошки! -->
```

Примечание

Не используйте более одного управляющего элемента внутри элемента `<label>`.

22.2. Базовое использование

Простая форма с элементами <label>:

```
<form action="/login" method="POST">
    <label for="username">Username:</label> <!-- Имя пользовате-
ля: -->
    <input id="username" type="text" name="username" />
    <label for="pass">Password:</label> <!-- Пароль: -->
    <input id="pass" type="password" name="pass" />
    <input type="submit" name="submit" />
</form>
```

Версия ≥ 5

```
<form id="my-form" action="/login" method="POST">
    <input id="username" type="text" name="username" />
    <label for="pass">Password:</label> <!-- Пароль: -->
    <input id="pass" type="password" name="pass" />
    <input type="submit" name="submit" />
</form>
<label for="username" form="my-form">Username:</label> <!-- Имя
пользователя: -->
```

Глава 23.
Элемент Output (Вывод)

Атрибут	Описание
Глобальные атрибуты	Атрибуты, доступные для любого элемента HTML5.
`name`	Строка, представляющая имя элемента `output` (вывод). Как элемент формы `output` может быть ссылкой по его имени с использованием свойства `document.forms`. Этот атрибут также используется для сбора значений при отправке формы.
`for`	Список идентификаторов элементов формы, разделенных пробелами (например, `<input id="inp1"> for value is "inp1"`), для которых вывод должен отображать вычисления.
`form`	Строка, представляющая `<form>`, с которой связан элемент `output`. Если `output` находится вне `<form>`, этот атрибут гарантирует, что `output` все еще принадлежит форме и будет учитываться при сборе и отправке данных этой формы.

Ссылка на дополнительную информацию по глобальным атрибутам (https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes).



23.1. Использование атрибутов For и Form в элементе Output

Следующий пример демонстрирует использование атрибутов `[for]` и `[form]` в элементе `<output>`. Обратите внимание, что для работы `<output>` требуется JavaScript. Встроенный JavaScript часто используется в формах, как показано в этом примере. Хотя элементы `<input>` имеют тип `type="number"`, их значения являются текстом, а не числами. Поэтому, если вам нужно выполнить вычисления, вы должны преобразовать каждое значение в число с помощью таких методов, как `parseInt()`, `parseFloat()`, `Number()` и т. д.

```
<!-- form1 будет собирать значения in1 и in2 при событии ‘input’. -->
<!-- Значение out1 будет суммой значений in1 и in2. -->
<form id="form1" name="form1" oninput="out1.value = parseInt(in1.
value, 10) + parseInt(in2.value, 10)">
    <fieldset>
        <legend>Пример использования Output</legend>
        <input type="number" id="in1" name="in1" value="0">
        <br/>
        +
        <input type="number" id="in2" name="in2" value="0">
    </fieldset>
</form>

<!-- Атрибут [for] позволяет out1 отображать вычисления для in1
и in2. -->
<!-- Атрибут [form] указывает, что form1 является владельцем out1,
даже если он не является потомком. -->
<output name="out1" for="in1 in2" form="form1">0</output>
```

23.2. Элемент Output с атрибутами

```
<output name="out1" form="form1" for="inp1 inp2"></output>
```

Глава 24.
Пустые элементы (Void Elements)

Не все HTML-теги имеют одинаковую структуру. В то время как большинство элементов требуют открывающего тега, закрывающего тега и содержимого, некоторые элементы — известные как пустые элементы (void elements) — требуют только открывающего тега, так как они сами по себе не содержат никаких элементов. В этой главе объясняется и демонстрируется правильное использование пустых элементов в HTML.

24.1. Пустые элементы

HTML 4.01/XHTML 1.0 Strict включает следующие пустые элементы:

- `area` — кликабельная, определенная область в изображении
- `base` — указывает базовый URL, от которого основываются все ссылки
- `br` — разрыв строки
- `col` — колонка в таблице [устарел]
- `hr` — горизонтальная линия
- `img` — изображение
- `input` — поле, в которое пользователи вводят данные
- `link` — связывает внешний ресурс с документом
- `meta` — предоставляет информацию о документе
- `param` — определяет параметры для плагинов

Стандарты HTML5 включают все не устаревшие теги из предыдущего списка, а также:

- `command` — представляет команду, которую пользователи могут вызвать [устарел]
- `keygen` — облегчает генерацию открытого ключа для веб-сертификатов [устарел]
- `source` — указывает источники медиа для элементов `picture`, `audio` и `video`

Пример ниже не включает пустые элементы:

```
<div>
    <a href="http://stackoverflow.com/">
        <h3>Click here to visit <i>Stack Overflow!</i></h3>
    </a>
```

```
    <button onclick="alert('Hello!');">Say Hello!</button>
    <p>My favorite language is <b>HTML</b>. Here are my others:</p>
    <ol>
        <li>CSS</li>
        <li>JavaScript</li>
        <li>PHP</li>
    </ol>
</div>
```

Обратите внимание, что каждый элемент имеет открывающий тег, закрывающий тег и текст или другие элементы внутри открывающего и закрывающего тегов. Однако пустые теги показаны в примере ниже:

```
<img src="https://cdn.sstatic.net/Sites/stackoverflow/company/
img/logos/so/so-icon.png" />
<br>
<hr>
<input type="number" placeholder="Enter your favorite number">
```

За исключением тега `img` все эти пустые элементы имеют только открывающий тег. Тег `img`, в отличие от других тегов, имеет самозакрывающийся / перед знаком больше в открывающем теге. Рекомендуется ставить пробел перед слешем.

Глава 25.
Медиаэлементы

Атрибут	Описание
`width`	Устанавливает ширину элемента в пикселях.
`height`	Устанавливает высоту элемента в пикселях.
`<source>`	Определяет ресурсы аудио- или видеофайлов.
`<track>`	Определяет текстовую дорожку для медиаэлементов.
`controls`	Отображает элементы управления.
`autoplay`	Автоматически начинает воспроизведение медиа.
`loop`	Воспроизводит медиа в повторяющемся цикле.
`muted`	Воспроизводит медиа без звука.
`poster`	Назначает изображение, которое будет отображаться до загрузки видео.

25.1. Аудио

HTML5 предоставляет новый стандарт для встраивания аудиофайлов на веб-страницу. Вы можете встроить аудиофайл на страницу с помощью элемента `<audio>`:

```
<audio controls>
    <source src="file.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
    <!-- Ваш браузер не поддерживает элемент audio. -->
</audio>
```

25.2. Видео

Вы также можете встроить видео на веб-страницу с помощью элемента `<video>`:

```
<video width="500" height="700" controls>
    <source src="video.mp4" type="video/mp4">
    Your browser does not support the video tag.
    <!-- Ваш браузер не поддерживает элемент video. -->
</video>
```

25.3. Использование элементов <video> и <audio> для отображения аудио/видео контента

Используйте HTML-элемент <video> или <audio> для встраивания видео/аудио контента в документ. Элемент <video>/<audio> содержит один или несколько источников видео/аудио. Чтобы указать источник, используйте либо атрибут src, либо элемент <source>; браузер выберет наиболее подходящий.

Пример тега <video>:

```
<!-- Простой пример видео -->
<video src="videofile.webm" autoplay poster="posterimage.jpg">
    Sorry, your browser doesn›t support embedded videos,
    but don›t worry, you can <a href="videofile.webm">download
it</a>
    and watch it with your favorite video player!
</video>

<!-- Видео с субтитрами -->
<video src="foo.webm">
    <track kind="subtitles" src="foo.en.vtt" srclang="en"
label="English">
    <track kind="subtitles" src="foo.sv.vtt" srclang="sv"
label="Svenska">
</video>

<!-- Простой пример видео -->
<video width="480" controls poster="https://archive.org/download/
WebmVp8Vorbis/webmvp8.gif">
    <source src="https://archive.org/download/WebmVp8Vorbis/
webmvp8.webm" type="video/webm">
    <source src="https://archive.org/download/WebmVp8Vorbis/
webmvp8_512kb.mp4" type="video/mp4">
    <source src="https://archive.org/download/WebmVp8Vorbis/
webmvp8.ogv" type="video/ogg">
    Your browser doesn›t support HTML5 video tag. <!-- Ваш
браузер не поддерживает тег HTML5 video. -->
</video>
```

Пример тега <audio>:

```
<!-- Простое воспроизведение аудио -->
<audio src="http://developer.mozilla.org/@api/deki/
files/2926/=AudioTest_(1).ogg" autoplay>
```

```
    Your browser does not support the <code>audio</code> element.
<!-- Ваш браузер не поддерживает элемент <code>audio</code>. -->
</audio>

<!-- Audio playback with captions --> <!-- Воспроизведение аудио
с субтитрами -->
<audio src="foo.ogg">
    <track kind="captions" src="foo.en.vtt" srclang="en"
label="English">
    <track kind="captions" src="foo.sv.vtt" srclang="sv"
label="Svenska">
</audio>
```

25.4. Видео в качестве заголовка или фона

Добавление видео, которое будет автоматически воспроизводиться в цикле без элементов управления и звука. Идеально подходит для видеозаголовка или фона:

```
<video width="1280" height="720" autoplay muted loop poster="video.
jpg" id="videobg">
    <source src="video.mp4" type="video/mp4">
    <source src="video.webm" type="video/webm">
    <source src="video.ogg" type="video/ogg">
</video>
```

Этот CSS предоставляет резервный вариант, если видео не может быть загружено. Обратите внимание, что рекомендуется использовать первый кадр видео в качестве изображения-заглушки (poster) `video.jpg`:

```
#videobg {
    background: url(video.jpg) no-repeat;
    background-size: cover;
}
```

Глава 26.
Элемент Progress

Параметр	Значение
`max`	Общий объем работы, необходимый для выполнения задачи
`value`	Количество уже выполненной работы
`position`	Этот атрибут возвращает текущую позицию элемента `<progress>`
`labels`	Этот атрибут возвращает список меток элемента `<progress>` (если они есть)

26.1. Прогресс

Элемент `<progress>` является новым в HTML5 и используется для отображения прогресса выполнения задачи:

```
<progress value="22" max="100"></progress>
```

Это создает полосу, заполненную на 22%.

26.2. Изменение цвета полосы прогресса

С помощью селектора `progress[value]` можно стилизовать полосы прогресса.

В этом примере задается ширина полосы прогресса `250px` и высота `20px`:

```
progress[value] {
    width: 250px;
    height: 20px;
}
```

Стилизация полос прогресса может быть особенно сложной.

Chrome / Safari / Opera

Эти браузеры используют селектор `-webkit-appearance` для стилизации тега `progress`. Чтобы переопределить это, мы можем сбросить внешний вид:

```
progress[value] {
    -webkit-appearance: none;
```

```
    appearance: none;
}
```

Теперь мы можем стилизовать сам контейнер:

```
progress[value]::-webkit-progress-bar {
    background-color: “green";
}
```

Firefox

Firefox стилизует полосу прогресса немного иначе. Мы должны использовать следующие стили:

```
progress[value] {
    -moz-appearance: none;
    appearance: none;
    border: none; /* Firefox также отображает границу */
}
```

Internet Explorer

Internet Explorer 10+ поддерживает элемент `progress`. Однако он не поддерживает свойство `background-color`. Вместо этого вам нужно использовать свойство `color`:

```
progress[value] {
    -webkit-appearance: none;
    -moz-appearance: none;
    appearance: none;
    border: none; /* Удаляем границу в Firefox */
    width: 250px;
    height: 20px;
    color: blue;
}
```

26.3. Запасной вариант с использованием HTML

Для браузеров, которые не поддерживают элемент `progress`, можно использовать следующий обходной путь:

```
<progress max="100" value="20">
    <div class="progress-bar">
        <span style="width: 20%;">Progress: 20%</span>
    </div>
</progress>
```

Браузеры, поддерживающие тег `progress`, проигнорируют вложенный `div`. Устаревшие браузеры, которые не распознают тег `progress`, отобразят `div` вместо него.

Глава 27.
Элементы управления выпадающими меню

27.1. Элемент <select>

Элемент `<select>` создает выпадающее меню, из которого пользователь может выбрать опцию:

```
<select name="">
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
    <option value="4">Four</option>
</select>
```

Изменение размера

Вы можете изменить размер выпадающего меню с помощью атрибута `size`. Значение 0 или 1 отображает стандартное выпадающее меню. Значение больше 1 преобразует выпадающее меню в поле, отображающее указанное количество строк, с одной опцией на строку и полосой прокрутки для навигации по доступным опциям:

```
<select name="" size="4"></select>
```

Меню с множественным выбором

По умолчанию пользователи могут выбрать только одну опцию. Добавление атрибута `multiple` позволяет пользователям выбирать несколько опций одновременно и отправлять все выбранные опции вместе с формой. Использование атрибута `multiple` автоматически преобразует выпадающее меню в поле, как если бы был задан размер. Размер по умолчанию в этом случае определяется конкретным браузером, и невозможно вернуть меню в выпадающий стиль, сохранив возможность множественного выбора:

```
<select name="" multiple></select>
```

При использовании атрибута `multiple` существует разница между значениями 0 и 1 для атрибута `size`, тогда как без этого атрибута разницы нет.

Использование 0 заставит браузер вести себя в соответствии с его стандартными настройками. Использование 1 явно установит размер поля в одну строку.

27.2. Опции

Опции внутри выпадающего меню — это то, что пользователь будет выбирать. Стандартный синтаксис для опции выглядит следующим образом:

```
<option>Some Option</option>
<!–Какая-то опция -->
```

Однако важно отметить, что текст внутри элемента `<option>` не всегда используется и по сути становится значением по умолчанию для атрибутов, которые не указаны.

Атрибуты, которые управляют внешним видом и функциональностью опции — это `value` и `label`. Атрибут `label` представляет текст, который будет отображаться в выпадающем меню (то, что вы видите и на что кликаете для выбора). Атрибут `value` представляет текст, который будет отправлен вместе с формой. Если один из этих атрибутов опущен, используется текст внутри элемента. Таким образом, приведенный выше пример можно «расширить» следующим образом:

```
<option label="Some Option" value="Some Option">
```

Обратите внимание на отсутствие текста внутри элемента и закрывающего тега, которые не требуются для создания опции в меню. Если бы они были включены, внутренний текст был бы проигнорирован, так как оба атрибута уже указаны, и текст не нужен. Однако чаще всего опции записываются с указанием значения, которое будет отправлено на сервер, и текста внутри элемента, который становится атрибутом `label`, например:

```
<option value="option1">Some Option</option>
<!–Какая-то опция -->
```

Выбор опции по умолчанию

Вы также можете указать определенную опцию, которая будет выбрана в меню по умолчанию, добавив атрибут `selected`. По умолчанию, если ни одна опция не указана как выбранная, первой опции в меню будет присвоен статус выбранной при отображении. Если более одной опции имеют атрибут `selected`, то последняя опция с этим атрибутом будет выбрана по умолчанию:

```
<option value="option1" selected>Some option</option>
<!–Какая-то опция -->
```

Если вы используете атрибут в меню с множественным выбором, то все опции с атрибутом `selected` будут выбраны по умолчанию, и ни одна не будет выбрана, если ни у одной опции нет этого атрибута:

```
<select multiple>
    <option value="option1" selected>Some option</option>
    <option value="option2" selected>Some option</option>
    <!-- Какая-то опция -->
</select>
```

27.3. Группы опций

Вы можете аккуратно сгруппировать опции внутри выпадающего меню, чтобы обеспечить более структурированный вид в длинном списке опций, используя элемент `<optgroup>`.

Синтаксис очень прост: используйте элемент с атрибутом `label` для указания заголовка группы и поместите внутрь ноль или более опций, которые должны быть в этой группе:

```
<select name="">
    <option value="milk">Milk</option> <!-- Молоко -->
    <optgroup label="Fruits"> <!-- Фрукты -->
        <option value="banana">Bananas</option> <!-- Бананы -->
        <option value="strawberry">Strawberries</option> <!—Клуб-
ника -->
    </optgroup>
    <optgroup label="Vegetables" disabled> <!-- Овощи -->
        <option value="carrot">Carrots</option> <!—Морковки -->
        <option value="zucchini">Zucchini</option> <!-- Цукини -->
    </optgroup>
</select>
```

При использовании групп опций не все опции должны быть сгруппированы. Также отключение группы опций отключает все опции внутри нее, и невозможно вручную включить отдельную опцию внутри отключенной группы.

27.4. Элемент <datalist>

Элемент `<datalist>` определяет список предопределенных опций для элемента `<input>`. Он предоставляет функцию «автозаполнения» для элементов `<input>`. Пользователи увидят выпадающий список опций по мере ввода:

```
<input list="Languages">
<datalist id="Languages">
    <option value="PHP">
    <option value="Perl">
    <option value="Python">
    <option value="Ruby">
    <option value="C+">
</datalist>
```

Поддержка браузерами

Браузер	Версия
Chrome	20.0
Edge	10.0
Mozilla	4.0
Safari	Не поддерживается
Opera	9.0

Глава 28.
Элемент <embed>

Параметр	Описание
`src`	Адрес ресурса
`type`	Тип встраиваемого ресурса
`width`	Горизонтальный размер
`height`	Вертикальный размер

28.1. Основное использование

Элемент `<embed>` является новым в HTML5. Этот элемент предоставляет точку интеграции для внешнего (обычно не HTML) приложения или интерактивного контента:

```
<embed src="myflash.swf">
```

28.2. Определение MIME-типа

MIME-тип должен быть определен с использованием атрибута `type`:

```
<embed type="video/mp4" src="video.mp4" width="640" height="480">
```

Глава 29.
IFrames (Встроенные фреймы)

Параметр	Описание
`name`	Задает имя элемента, которое используется с тегом `<a>` для изменения атрибута `src iframe`.
`width`	Задает ширину элемента в пикселях.
`height`	Задает высоту элемента в пикселях.
`src`	Указывает страницу, которая будет отображаться во фрейме.
`srcdoc`	Указывает содержимое, которое будет отображаться во фрейме, при условии, что браузер поддерживает этот атрибут. Содержимое должно быть валидным HTML.
`sandbox`	При установке содержимое `iframe` рассматривается как происходящее из уникального источника, и такие функции как скрипты, плагины, формы и всплывающие окна будут отключены. Ограничения могут быть выборочно ослаблены путем добавления списка значений, разделенных пробелами.
`allowfullscreen`	Разрешает ли содержимому `iframe` использовать метод `requestFullscreen()`.

29.1. Основы встроенного фрейма

Термин `IFrame` означает «встроенный фрейм» (Inline Frame). Он может использоваться для включения другой страницы в вашу страницу. Это создаст небольшой фрейм, который отображает точное содержимое файла `base.html`.

```
<iframe src="base.html"></iframe>
```

29.2. Песочница (Sandboxing)

Следующий пример встраивает ненадежную веб-страницу со всеми включенными ограничениями:

```
<iframe sandbox src="http://example.com/"></iframe>
```

Чтобы разрешить странице выполнять скрипты и отправлять формы, добавьте значения `allow-scripts` и `allow-forms` в атрибут `sandbox`:

```
<iframe sandbox="allow-scripts allow-forms" src="http://example.
com/"></iframe>
```

Если на том же домене, на котором находится родительская веб-страница, есть ненадежное содержимое (например, пользовательские комментарии), `iframe` может быть использован для отключения скриптов, при этом позволяя родительскому документу взаимодействовать с его содержимым с помощью JavaScript:

```
<iframe sandbox="allow-same-origin allow-top-navigation"
        src="http://example.com/untrusted/comments/page2">
```

Родительский документ может добавлять обработчики событий и изменять размер `IFrame`, чтобы он соответствовал его содержимому. Это, вместе с `allow-top-navigation`, может сделать песочницу `iframe` частью родительского документа.

Песочница не является заменой санитизации входных данных, но может использоваться как часть стратегии защиты в глубину. Также следует помнить, что эту песочницу можно обойти, если злоумышленник убедит пользователя посетить источник `iframe` напрямую. HTTP-заголовок Content Security Policy (https://w3c.github.io/webappsec-csp/#directive-sandbox) может быть использован для смягчения этой атаки.



29.3. Установка размера фрейма

Размер `IFrame` может быть изменен с помощью атрибутов `width` и `height`, где значения указываются в пикселях (HTML 4.01 допускал процентные значения, но HTML5 разрешает только значения в CSS-пикселях):

```
<iframe src="base.html" width="800" height="600"></iframe>
```

29.4. Использование атрибута srcdoc

Атрибут `srcdoc` может быть использован (вместо атрибута `src`) для указания точного содержимого `iframe` в виде целого HTML-документа. Это создаст `IFrame` с текстом «IFrames are cool!»:

```
<iframe srcdoc="<p>IFrames are cool!</p>"></iframe>
```

Если браузер не поддерживает атрибут srcdoc, IFrame вместо этого будет использовать атрибут src. Однако если оба атрибута src и srcdoc присутствуют и поддерживаются браузером, приоритет имеет srcdoc:

```
<iframe srcdoc="<p>Iframes are cool!</p>" src="base.html"></iframe>
```

В приведенном выше примере, если браузер не поддерживает атрибут srcdoc, он отобразит содержимое страницы base.html.

29.5. Использование якорей с IFrames

Обычно изменение веб-страницы внутри IFrame инициируется изнутри IFrame, например, при нажатии на ссылку внутри IFrame. Однако можно изменить содержимое IFrame извне. Для этого можно использовать тег <a>, у которого атрибут href указывает на нужный URL, а атрибут target — на имя iframe:

```
<iframe src="webpage.html" name="myIframe"></iframe>
<a href="different_webpage.html" target="myIframe">Change the
Iframe content to different_webpage.html</a>
```

Глава 30.
Языки контента

30.1. Основной язык документа

Рекомендуется указывать основной язык документа в элементе `html`:

```
<html lang="en"> <!-- Основной язык документа – английский. -->
```

Если в документе не указаны другие атрибуты `lang`, это означает, что весь контент (содержимое элементов и текстовые значения атрибутов) написан на этом языке. Если документ содержит части на других языках, эти части должны иметь свои собственные атрибуты `lang`, чтобы «переопределить» объявление языка.

30.2. Язык элемента

Атрибут `lang` используется для указания языка содержимого элемента и текстовых значений атрибутов:

```
<p lang="en">The content of this element is in English.</p>
<!-- Содержимое этого элемента на английском языке. -->
<p lang="en" title="The value of this attribute is also in
English.">The content of this element is in English.</p> <!--
Содержимое этого элемента на английском языке. -->
```

Объявление языка наследуется:

```
<div lang="en"> <!-- Основной язык – английский. -->
    <p>This element contains English content.</p> <!-- Этот
элемент содержит контент на английском языке. -->
    <p title="This attribute, too.">Same with this element.</p>
<!-- Этот элемент также содержит контент на английском языке. -->
</div>
```

30.3. Элементы с несколькими языками

Вы можете «переопределить» объявление языка:

```
<p lang="en">This English sentence contains the German word <span
lang="de">Hallo</span>.</p> <!-- Это предложение на английском
содержит немецкое слово «Hallo". -->
```

30.4. Региональные URL

Можно добавить атрибут `hreflang` к элементам `<a>` и `<area>`, которые создают гиперссылки. Таким образом указывается язык связанного ресурса. Указанный язык должен быть действительным тегом BCP 47[1]:

```
<p>
    <a href="example.org" hreflang="en">example.org</a> is one of
IANA's example domains.
    <!-- example.org — один из примеров доменов IANA. -->
</p>
```

Примечание: IETF Network Working Group: RFC 5646 Tags for Identifying Languages, IETF, September 2009 (https://www.ietf.org/rfc/bcp/bcp47.txt).



30.5. Обработка атрибутов с разными языками

Вы можете «переопределить» объявление языка родительского элемента, добавив любой элемент (кроме `applet`, `base`, `basefont`, `br`, `frame`, `frameset`, `hr`, `iframe`, `meta`, `param`, `script` из HTML 4.0) с собственным атрибутом `lang`:

```
<p lang="en" title="An English paragraph"> <!-- Абзац на англий-
ском языке. -->
    <span lang="de" title="A German sentence">Hallo Welt!</span>
    <!-- Предложение на немецком языке. -->
</p>
```

Глава 31.
SVG

SVG (Scalable Vector Graphics) — это масштабируемая векторная графика. SVG используется для определения графики в вебе.

Элемент HTML <svg> является контейнером для SVG-графики.

SVG предоставляет несколько методов для рисования путей, прямоугольников, кругов, текста и графических изображений.

31.1. Встроенный SVG

SVG можно встраивать напрямую в HTML-документ. Встроенный SVG можно стилизовать и управлять им с помощью CSS и JavaScript:

```
<body>
    <svg class="attention" xmlns="http://www.w3.org/2000/svg"
         xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0
1000 1000">
        <path id="attention"
              d="m571,767l0,-106q0,-8,-5,-13t-12,-5l-
108,0q-7,0,-12,5t-5,13l0,106q0,8,5,13t12,6l108,0q7,0,12,-
6t5,-13Zm-1,-208l10,-257q0,-6,-5,-10q-7,-6,-14,-6-
l-122,0q-7,0,-14,6q-5,4,-5,12l9,255q0,5,6,9t13,3l103,
0q8,0,13,-3t6,-9Zm-7,-522l428,786q20,35,-1,70q-10,17,-26,26t-
35,10l-858,0q-18,0,-35,-10t-26,-26q-21,-35,-1,-70l429,-786q9,-
17,26,-27t36,-10t36,10t27,27Z" />
    </svg>
</body>
```

Встроенный SVG можно стилизовать с помощью соответствующего CSS-класса:

```
.attention {
    fill: red;
    width: 50px;
    height: 50px;
}
```

Результат будет выглядеть следующим образом:



31.2. Встраивание внешних SVG-файлов в HTML

Вы можете использовать элементы `<img>` или `<object>` для встраивания внешних SVG-элементов. Указание высоты и ширины необязательно, но настоятельно рекомендуется.

Использование элемента `<img>`

```
<img src="attention.svg" width="50" height="50">
```

Использование `<img>` не позволяет стилизовать SVG с помощью CSS или управлять им с помощью JavaScript.

Использование элемента `<object>`

```
<object type="image/svg+xml" data="attention.svg" width="50"
height="50">
```

В отличие от `<img>`, `<object>` напрямую импортирует SVG в документ, поэтому им можно управлять с помощью JavaScript и CSS.

31.3. Встраивание SVG с помощью CSS

Вы можете добавлять внешние SVG-файлы с помощью свойства `background-image`, как и с любым другим изображением.

HTML

```
<div class="attention"></div>
```

CSS

```
.attention {
    background-image: url(attention.svg);
    background-size: 100% 100%;
    width: 50px;
    height: 50px;
}
```

Также можно встроить изображение напрямую в CSS-файл с использованием data URL:

```
background-image: url(data:image/svg+xml,%3Csvg%20
xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20xmlns
%3Axlink%3D%22http%3A%2F%2Fwww.w3.org%2F1999%2Fxlink%22%20
viewBox%3D%220%200%201000%201000%22%20%3E%0D%0A%3Cpath%20
id%3D%22attention%22%20d%3D%22m571%2C767l0%2C-106q0%2C-8%2C-5-
%2C-13t-12%2C-5l-108%2C0q-7%2C0%2C-12%2C5t-5%2C13l0%2C106q0%2C8%2
C5%2C13t12%2C6l108%2C0q7%2C0%2C12%2C-6t5%2C-13Zm-1%2C-208l10%2C-
257q0%2C-6%2C-5%2C-10q-7%2C-6%2C-14%2C-6l-122%2C0q-7%2C0%2C-
14%2C6q-5%2C4%2C-5%2C12l9%2C255q0%2C5%2C6%2C9t13%2C3l103%2C0q8
%2C0%2C13%2C-3t6%2C-9Zm-7%2C-522l428%2C786q20%2C35%2C-1%2C70q-
10%2C17%2C-26%2C26t-35%2C10l-858%2C0q-18%2C0%2C-35%2C-10t-26%2C-
26q-21%2C-35%2C-1%2C-70l429%2C-786q9%2C-17%2C26%2C-27t36%2C-
10t36%2C10t27%2C27Z%22%20%2F%3E%0D%0A%3C%2Fsvg%3E);
```

Глава 32.
Canvas (Холст)

Атрибут	Описание
`height`	Указывает высоту холста
`width`	Указывает ширину холста

32.1. Базовый пример

Элемент `<canvas>` был введен в HTML5 для рисования графики:

```
<canvas id="myCanvas">
    Cannot display graphic. Canvas is not supported by your browser
(IE<9)
    <!-- Не удается отобразить графику. Ваш браузер (IE<9) не под-
держивает элемент canvas. -->
</canvas>
```

Вышеприведенный код создаст прозрачный элемент HTML `<canvas>` размером 300 × 150 пикселей.

Элемент `<canvas>` можно использовать для рисования удивительных вещей, таких как фигуры, графики, манипуляции с изображениями, создания увлекательных игр и т. д. с помощью JavaScript.

Двумерный слой для рисования на холсте представлен объектом `CanvasRenderingContext2D`, который можно получить из элемента `HTMLCanvasElement` с помощью метода `.getContext("2d")`:

```
var ctx = document.getElementById(“myCanvas").getContext(“2d");
// теперь мы можем ссылаться на 2D-контекст холста с помощью
`ctx`
ctx.fillStyle = «#f00";
ctx.fillRect(0, 0, ctx.canvas.width, ctx.canvas.height); // x, y,
width, height
ctx.fillStyle = «#000";
ctx.fillText(«My red canvas with some black text", 24, 32); //
text, x, y
```

32.2. Рисование двух прямоугольников на <canvas>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Draw two rectangles on the canvas</title> <!--
Нарисуйте два прямоугольника на холсте -->
    <style>
        canvas {
            border: 1px solid gray; <!-- Граница холста: 1px
сплошная серая -->
        }
    </style>
    <script async>
        window.onload = init; // вызов init() после полной за-
грузки окна

        function init() {
            // #1 – получаем ссылку на элемент <canvas>
            var canvas = document.querySelector('canvas');

            // #2 – получаем ссылку на контекст рисования и API
для рисования
            var ctx = canvas.getContext('2d');

            // #3 – все операции заливки теперь выполняются
красным цветом
            ctx.fillStyle = 'red';

            // #4 – заливаем прямоугольник 100x100 на позиции
x=0, y=0
            ctx.fillRect(0, 0, 100, 100);

            // #5 – все операции заливки теперь выполняются
зеленым цветом
            ctx.fillStyle = 'green';

            // #6 – заливаем прямоугольник 50x50 на позиции x=25,
y=25
            ctx.fillRect(25, 25, 50, 50);
        }
```

```
    </script>
</head>
<body>
    <canvas width=300 height=200>Your browser does not support
canvas.</canvas> <!-- Ваш браузер не поддерживает элемент canvas.
-->
</body>
</html>
```

Результат выглядит следующим образом:



Глава 33.
Метаинформация

Метатеги в HTML-документах предоставляют полезную информацию о документе, включая описание, ключевые слова, автора, даты изменений и около 90 других полей. В этой главе рассматриваются использование и назначение этих тегов.

33.1. Информация о странице

`application-name`

Указывает имя веб-приложения, которое представляет страница:

```
<meta name="application-name" content="OpenStreetMap">
```

Если это не веб-приложение, метатег `application-name` не должен использоваться.

`author`

Устанавливает автора страницы:

```
<meta name="author" content="Your Name">
```

Можно указать только одно имя.

`description`

Устанавливает описание страницы:

```
<meta name="description" content="Page Description">
```

Метатег `description` может использоваться различными поисковыми системами при индексации вашей веб-страницы для поиска. Обычно описание, содержащееся в этом теге, представляет собой краткое резюме, которое отображается под основным заголовком страницы/сайта в результатах поиска. Google обычно использует только первые 20–25 слов вашего описания.

`generator`

```
<meta name="generator" content="HTML Generator 1.42">
```

Идентифицирует один из программных пакетов, используемых для генерации документа. Используется только для страниц, где разметка генерируется автоматически.

`keywords`

Устанавливает ключевые слова для поисковых систем (через запятую):

```
<meta name="keywords" content="Keyword1, Keyword2">
```

Метатег `keywords` иногда используется поисковыми системами для определения поискового запроса, релевантного вашей веб-странице.

Как правило, не стоит добавлять слишком много слов, так как большинство поисковых систем, использующих этот метатег для индексации, будут учитывать только первые примерно 20 слов. Убедитесь, что вы указали наиболее важные ключевые слова первыми.

33.2. Кодировка символов

Атрибут `charset` указывает кодировку символов для HTML-документа и должен быть допустимой кодировкой символов (например windows-1252, SO-8859-2, Shift_JIS и UTF-8). UTF-8 (Unicode) является наиболее широко используемой и должна применяться в любых новых проектах:

```
<meta charset="UTF-8">
<meta charset="ISO-8859-1">
```

Все браузеры всегда распознают форму `<meta charset>`, но если по какой-то причине вам нужно, чтобы ваша страница была валидной HTML 4.01, вы можете использовать следующее:

```
<meta http-equiv="content-type" content="text/html;
charset=UTF-8">
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
```

См. также стандарт кодирования (Encoding Standard), чтобы просмотреть все доступные метки кодировки символов, которые распознают браузеры (https://encoding.spec.whatwg.org/#names-and-labels).



33.3. Роботы (robots)

Атрибут `robots`, поддерживаемый несколькими крупными поисковыми системами, контролирует, разрешено ли поисковым роботам индексировать страницу, а также должны ли они переходить по ссылкам на странице.

```
<meta name="robots" content="noindex">
```

Этот пример запрещает всем поисковым системам показывать страницу в результатах поиска. Другие допустимые значения:

Значение/Директива	Описание
`all`	По умолчанию. Эквивалентно `index`, `follow`. См. примечание ниже.
`noindex`	Не индексировать страницу.
`nofollow`	Не переходить по ссылкам на этой странице.
`follow`	По ссылкам на странице можно переходить. См. примечание ниже.
`none`	Эквивалентно `noindex`, `nofollow`.
`noarchive`	Не создавать кешированную версию этой страницы в результатах поиска.
`nocache`	Синоним `noarchive`, используемый некоторыми ботами, например Bing.
`nosnippet`	Не показывать сниппет этой страницы в результатах поиска.
`noodp`	Не использовать метаданные этой страницы из проекта Open Directory для заголовков или сниппетов в результатах поиска.
`notranslate`	Не предлагать переводы этой страницы в результатах поиска.
`noimageindex`	Не индексировать изображения на этой странице.
`unavailable_after` [RFC-850 дата/время]	Не показывать эту страницу в результатах поиска после указанной даты/времени. Дата/время должны быть указаны в формате RFC 850.

Примечание: явное определение `index` и/или `follow`, хотя и является допустимым значением, необязательно, так как практически все поисковые системы предполагают, что они могут это делать, если явно не запрещено. Подобно тому как работает файл `robots.txt`, поисковые системы обычно ищут только то, что им запрещено делать. Указание только запретов также предотвращает случайное указание противоположностей (например, `index`, ..., `noindex`), которые не все поисковые системы обрабатывают одинаково.

33.4. Социальные сети

Open Graph — это стандарт метаданных, который расширяет обычную информацию, содержащуюся в разметке `<head>` сайта. Это позволяет таким платформам, как Facebook, отображать более глубокую и детальную информацию о сайте в структурированном формате. Эта информация автоматиче-

ски отображается, когда пользователи делятся ссылками на сайты, содержащие метаданные Open Graph, на Facebook.

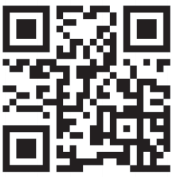
Facebook / Open Graph

```
<meta property="fb:app_id" content="123456789">
<meta property="og:url" content="https://example.com/page.html">
<meta property="og:type" content="website">
<meta property="og:title" content="Content Title"> <!-- Заголовок
контента -->
<meta property="og:image" content="https://example.com/image.jpg">
<meta property="og:description" content="Description Here"> <!--
Описание здесь -->
<meta property="og:site_name" content="Site Name"> <!-- Название
сайта -->
<meta property="og:locale" content="en_US">
<meta property="article:author" content="">
```

Разметка Open Graph для Facebook (https://developers.facebook.com/docs/sharing/webmasters#markup).



Протокол Open Graph (https://ogp.me/).



Facebook / Instant Articles

```
<meta charset="utf-8">
<meta property="op:markup_version" content="v1.0">
<!-- URL веб-версии вашей статьи -->
<link rel="canonical" href="http://example.com/article.html">
<!-- Стиль, используемый для этой статьи -->
<meta property="fb:article_style" content="myarticlestyle">
```

Создание статей для Facebook Instant Articles (https://developers.facebook.com/docs).



Twitter использует собственную разметку для метаданных. Эти метаданные используются для управления отображением твитов, содержащих ссылки на сайт.

Twitter

```
<meta name="twitter:card" content="summary">
<meta name="twitter:site" content="@site_account">
<meta name="twitter:creator" content="@individual_account">
<meta name="twitter:url" content="https://example.com/page.html">
<meta name="twitter:title" content="Content Title"> <!-- Заголовок
контента -->
<meta name="twitter:description" content="Content description less
than 200 characters"> <!-- Описание контента менее 200 символов -->
<meta name="twitter:image" content="https://example.com/image.jpg">
```

Руководство по началу работы с Twitter Cards (https://developer.x.com/en/docs/x-for-websites/cards/guides/getting-started).



Google+ / Schema.org

```
<link href="https://plus.google.com/+YourPage" rel="publisher">
<meta itemprop="name" content="Content Title"> <!-- Заголовок кон-
тента -->
<meta itemprop="description" content="Content description less than
200 characters"> <!-- Описание контента менее 200 символов -->
<meta itemprop="image" content="https://example.com/image.jpg">
```

33.5. Управление мобильной версткой

Оптимизированные для мобильных устройств сайты часто используют тег `<meta name="viewport">` следующим образом:

```
<meta name="viewport" content="width=device-width, initial-
scale=1">
```

Элемент `viewport` дает браузеру инструкции о том, как управлять размерами страницы и масштабированием в зависимости от используемого устройства.

В приведенном выше примере `content="width=device-width"` означает, что браузер будет отображать ширину страницы равной ширине экрана устройства. Таким образом, если ширина экрана составляет 480 пикселей,

ширина окна браузера будет 480 пикселей. `initial-scale=1` указывает начальный масштаб (в данном случае 1, что означает отсутствие масштабирования).

Ниже приведены атрибуты, поддерживаемые этим тегом:

Атрибут	Описание
`width`	Ширина виртуального окна просмотра устройства. Значения: `device-width` или фактическая ширина в пикселях, например 480.
`height`	Высота виртуального окна просмотра устройства. Значения: `device-height` или фактическая высота в пикселях, например 600.
`initial-scale`	Начальный масштаб при загрузке страницы. `1.0` означает отсутствие масштабирования.
`minimum-scale`	Минимальный масштаб, который может установить посетитель на странице. `1.0` означает отсутствие масштабирования.
`maximum-scale`	Максимальный масштаб, который может установить посетитель на странице. `1.0` означает отсутствие масштабирования.
`user-scalable`	Позволяет устройству масштабировать страницу. Значения: `yes` или `no`. Если установлено `no`, пользователь не сможет масштабировать страницу. По умолчанию `yes`. Настройки браузера могут игнорировать это правило.

Примечания

1. Свойство `width` может быть указано либо в пикселях (`width=600`), либо как `device-width` (`width=device-width`), что представляет физическую ширину экрана устройства.

2. Аналогично свойство `height` может быть указано либо в пикселях (`height=600`), либо как `device-height` (`height=device-height`), что представляет физическую высоту экрана устройства.

33.6. Автоматическое обновление

Чтобы обновлять страницу каждые пять секунд, добавьте этот мета-элемент в элемент `<head>`:

```
<meta http-equiv="refresh" content="5">
```

ВНИМАНИЕ! Хотя это допустимая команда, рекомендуется не использовать ее из-за негативного влияния на пользовательский опыт. Слишком частое обновление страницы может привести к ее зависанию, а также часто прокручивает страницу вверх. Если информация на странице должна обновляться непрерывно, существуют гораздо более эффективные способы сделать это, обновляя только часть страницы.

33.7. Распознавание телефонных номеров

Мобильные платформы, такие как iOS, автоматически распознают телефонные номера и преобразуют их в ссылки `tel:`. Хотя эта функция очень удобна, система иногда ошибочно распознает ISBN-коды и другие числа как телефонные номера.

Чтобы отключить автоматическое распознавание и форматирование телефонных номеров в мобильном Safari и некоторых других мобильных браузерах на основе WebKit, используйте этот метатег:

```
<meta name="format-detection" content="telephone=no">
```

33.8. Автоматическое перенаправление

Иногда вашей веб-странице требуется автоматическое перенаправление. Например, для перенаправления на `example.com` через 5 секунд:

```
<meta http-equiv="refresh" content="5;url=https://www.example.com/" />
```

Эта строка перенаправит вас на указанный сайт (в данном случае example.com) через 5 секунд.

Если вам нужно изменить задержку перед перенаправлением, просто измените число перед `;url=`, чтобы задать новую задержку.

33.9. Веб-приложение

Вы можете настроить свое веб-приложение или сайт так, чтобы на главный экран устройства добавлялся ярлык приложения, а само приложение запускалось в полноэкранном режиме (без адресной строки) с помощью пункта меню «Добавить на главный экран» в Chrome для Android.

Следующие метатеги откроют веб-приложение в полноэкранном режиме.

Android Chrome

```
<meta name="mobile-web-app-capable" content="yes">
```

iOS

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

Вы также можете задать цвет строки состояния и адресной строки с помощью метатега.

Android Chrome

```
<meta name="theme-color" content="black">
```

iOS

```
<meta name="apple-mobile-web-app-status-bar-style"
content="black">
```

Глава 34.
Разметка компьютерного кода

34.1. Блок с использованием <pre> и <code>

Если форматирование кода (пробелы, новые строки, отступы) имеет значение, используйте элемент `pre` в сочетании с элементом `code`:

```
<pre>
    <code>
    x = 42
    if x == 42:
        print “x is …           … 42"
    </code>
</pre>
```

Все равно необходимо экранировать символы, имеющие специальное значение в HTML (например, < заменяется на `<`). Поэтому для отображения блока HTML-кода (`<p>This is a paragraph.</p>`) это может выглядеть так:

```
<pre>
    <code>
    &lt;p>This is a paragraph.&lt;/p> <!-- Это абзац. -->
    </code>
</pre>
```

34.2. Встроенный код с использованием <code>

Если предложение содержит компьютерный код (например, название HTML-элемента), используйте элемент `code` для его разметки:

```
<p>The <code>a</code> element creates a hyperlink.</p>
<!-- Элемент <code>a</code> создает гиперссылку. -->
```

Глава 35.
Разметка цитат

35.1. Встроенные цитаты с использованием <q>

Элемент <q> может использоваться для цитаты, которая является частью предложения:

```
<p>She wrote <q>The answer is 42.</q> and everyone agreed.</p>
<!-- Она написала <q>Ответ — 42.</q>, и все согласились. -->
```

Кавычки

Версия ≤ 4.01

Кавычки не следует добавлять вручную. Пользовательские агенты должны (в HTML 4.01) или обязаны (в HTML 4.0) отображать их автоматически.

Версия = 5

Кавычки не должны добавляться вручную. Пользовательские агенты будут отображать их автоматически.

URL источника (атрибут `cite`)

Атрибут `cite` может использоваться для указания URL источника цитаты:

```
<p>She wrote <q cite="http://example.com/blog/hello-world">The
answer is 42.</q> and everyone agreed.</p>
<!-- Она написала <q cite="http://example.com/blog/
hello-world">Ответ — 42.</q>, и все согласились. -->
```

Обратите внимание, что браузеры обычно не отображают этот URL, поэтому, если источник важен, следует добавить гиперссылку (элемент <a>) дополнительно.

35.2. Блочные цитаты с использованием <blockquote>

Элемент <blockquote> может использоваться для блочной цитаты:

```
<blockquote>
    <p>The answer is 42.</p>
```

```
    <!-- Ответ — 42. -->
</blockquote>
```

URL источника (атрибут `cite`)

Атрибут `cite` может использоваться для указания URL источника цитаты:

```
<blockquote cite="http://example.com/blog/hello-world">
    <p>The answer is 42.</p>
    <!-- Ответ — 42. -->
</blockquote>
```

Обратите внимание, что браузеры обычно не отображают этот URL, поэтому, если источник важен, следует добавить гиперссылку (элемент `<a>`) дополнительно (см. дополнительно ниже, где разместить эту ссылку).

Цитирование/Атрибуция

Версия ≤ 4.01

Цитирование/атрибуция не должны быть частью элемента `<blockquote>`:

```
<blockquote cite="http://example.com/blog/hello-world">
    <p>The answer is 42.</p>
    <!-- Ответ — 42. -->
</blockquote>
<p>Source: <cite><a href="http://example.com/blog/hello-world"
rel="external">Hello World</a></cite></p>
<!-- Источник: <cite><a href="http://example.com/blog/hello-
world" rel="external">Hello World</a></cite> -->
```

Вы можете добавить элемент `<div>` для группировки цитаты и атрибуции, но семантически связать их невозможно.

Элемент `<cite>` может использоваться для указания источника цитаты (но не для имени автора).

Версия = 5

Цитирование/атрибуция (например, гиперссылка, указывающая на URL источника) может находиться внутри `<blockquote>`, но в этом случае она должна быть внутри элемента `<cite>` (для встроенной атрибуции) или элемента `<footer>`:

```
<blockquote cite="http://example.com/blog/hello-world">
    <p>The answer is 42.</p>
    <!-- Ответ — 42. -->
    <footer>
```

```
        <p>Source: <cite><a href="http://example.com/blog/hello-
world" rel="external">Hello World</a></cite></p>
        <!-- Источник: <cite><a href="http://example.com/blog/
hello-world" rel="external">Hello World</a></cite> -->
    </footer>
</blockquote>
```

Элемент `<cite>` может использоваться для указания источника цитаты или имени автора цитаты.

Глава 36.
Tabindex

Значение	Описание
отрицательное	Элемент будет доступен для фокусировки, но не должен быть доступен через последовательную навигацию с клавиатуры.
0	Элемент будет доступен для фокусировки и доступен через последовательную навигацию с клавиатуры, но его относительный порядок определяется соглашениями платформы.
положительное	Элемент должен быть доступен для фокусировки и доступен через последовательную навигацию с клавиатуры; его относительный порядок определяется значением атрибута: последовательность следует возрастающему числу `tabindex`.

36.1. Добавление элемента в порядок табуляции

```
<div tabindex="0">Some button</div> <!-- Какая-то кнопка -->
```

Примечание: по возможности используйте нативные HTML-элементы, такие как `<button>` или `<a>`.

36.2. Удаление элемента из порядка табуляции

```
<button tabindex="-1">This button will not be reachable by tab</
button> <!-- Эта кнопка не будет доступна через табуляцию -->
```

Элемент будет удален из порядка табуляции, но останется доступным для фокусировки.

36.3. Определение пользовательского порядка табуляции (не рекомендуется)

```
<div tabindex="2">Second</div> <!-- Второй -->
<div tabindex="1">First</div> <!-- Первый -->
```

Положительные значения вставят элемент в порядок табуляции на позицию, соответствующую его значению. Элементы без предпочтения (например, `tabindex="0"` или нативные элементы, такие как `<button>` и `<a>`) будут добавлены после тех, у которых есть предпочтение.

Положительные значения не рекомендуются, так как они нарушают ожидаемое поведение табуляции и могут сбить с толку пользователей, которые полагаются на скринридеры. Попробуйте создать естественный порядок, изменив структуру DOM.

Глава 37.
Глобальные атрибуты

Атрибут	Описание
`class`	Определяет одно или несколько имен классов для элемента.
`contenteditable`	Устанавливает, можно ли редактировать содержимое элемента.
`contextmenu`	Определяет контекстное меню, которое отображается при правом клике на элемент.
`dir`	Устанавливает направление текста внутри элемента.
`draggable`	Устанавливает, можно ли перетаскивать элемент.
`hidden`	Скрывает элемент, который в данный момент не используется на странице.
`id`	Определяет уникальный идентификатор для элемента.
`lang`	Определяет язык содержимого элемента и значений его текстовых атрибутов.
`spellcheck`	Устанавливает, нужно ли проверять орфографию и грамматику содержимого элемента.
`style`	Определяет набор встроенных CSS-стилей для элемента.
`tabindex`	Устанавливает порядок, в котором элементы на странице будут переключаться с помощью клавиши Tab.
`title`	Определяет дополнительную информацию об элементе, обычно в виде текста подсказки при наведении курсора.
`translate`	Определяет, нужно ли переводить содержимое элемента.

37.1. Атрибут contenteditable

```
<p contenteditable>This is an editable paragraph.</p> <!-- Это редактируемый абзац. -->
```

При клике на абзац его содержимое можно редактировать, как в текстовом поле.

Если атрибут `contenteditable` не установлен для элемента, он будет унаследован от родительского элемента. Таким образом, весь дочерний

текст редактируемого элемента также будет редактируемым, но вы можете отключить это для конкретного текста, например:

```
<p contenteditable>
    This is an editable paragraph. <!-- Это редактируемый абзац.
-->
    <span contenteditable="false">But not this.</span> <!--
Но не этот. -->
</p>
```

Обратите внимание, что нередактируемый текстовый элемент внутри редактируемого элемента все равно будет иметь текстовый курсор, унаследованный от родительского элемента.

Глава 38.
Кеш HTML5

38.1. Базовый пример кеша HTML5

Это наш файл `index.html`:

```
<!DOCTYPE html>
<html manifest="index.appcache">
<body>
    <p>Content</p> <!-- Содержимое -->
</body>
</html>
```

Затем мы создадим файл `index.appcache` со следующим содержимым:

```
CACHE MANIFEST
index.html
```

Укажите файлы, которые вы хотите кешировать. Загрузите `index.html`, затем перейдите в автономный режим и перезагрузите вкладку.

Примечание: в этом примере оба файла должны находиться в одной папке.

Глава 39.
Атрибуты событий HTML

39.1. События форм HTML

События, вызываемые действиями внутри HTML-формы (применяются почти ко всем HTML-элементам, но чаще всего используются в элементах форм):

Атрибут	Описание
`onblur`	Срабатывает в момент, когда элемент теряет фокус.
`onchange`	Срабатывает в момент изменения значения элемента.
`oncontextmenu`	Скрипт, выполняемый при вызове контекстного меню.
`onfocus`	Срабатывает в момент, когда элемент получает фокус.
`oninput`	Скрипт, выполняемый при получении элементом пользовательского ввода.
`oninvalid`	Скрипт, выполняемый при невалидности элемента.
`onreset`	Срабатывает при нажатии кнопки «Сброс» (reset) в форме.
`onsearch`	Срабатывает, когда пользователь вводит что-то в поле поиска (для `<input type="search">`).
`onselect`	Срабатывает после выделения текста в элементе.
`onsubmit`	Срабатывает при отправке формы.

39.2. События клавиатуры

Атрибут	Описание
`onkeydown`	Срабатывает, когда пользователь нажимает клавишу.
`onkeypress`	Срабатывает, когда пользователь нажимает клавишу.
`onkeyup`	Срабатывает, когда пользователь отпускает клавишу.

Глава 40.
Символьные сущности (Character Entities)

40.1. Символьные сущности в HTML

Многие символы и специальные знаки необходимы при разработке веб-страниц на HTML. Однако, как мы знаем, иногда использование символов напрямую может конфликтовать с самим HTML-кодом, в котором некоторые символы зарезервированы, а также могут быть недоступны на клавиатуре. Таким образом, чтобы избежать конфликтов и при этом иметь возможность использовать различные символы в нашем коде, консорциум W3C предоставляет нам так называемые символьные сущности (Character Entities).

Символьные сущности предопределены с использованием имени сущности (Entity Name) — `&имя_сущности;` и номера сущности (Entity Number) — `&#номер_сущности;`. Таким образом, нам нужно использовать один из этих двух вариантов, чтобы отобразить нужный символ на нашей странице.

Список некоторых символьных сущностей можно найти по адресу: https://dev.w3.org/html5/html-author/charref.



Простой пример использования символьной сущности для «лупы»:

```
<input type="text" placeholder=" &#128269; Search"/>
```

Этот код отображается следующим образом:



40.2. Распространенные специальные символы

Некоторые символы могут быть зарезервированы для HTML и не могут быть использованы напрямую, так как это может нарушить работу HTML-

кода. Например, попытка отобразить угловые скобки (< и >) в исходном коде может привести к неожиданным результатам в выводе. Аналогично пробелы, написанные в исходном коде, могут отображаться не так, как ожидается, в итоговом HTML. Некоторые символы, такие как ☎, отсутствуют в наборе символов ASCII.

Для этой цели созданы символьные сущности. Они имеют вид `&имя_сущности;` или `&#номер_сущности;`. Ниже приведены некоторые из доступных HTML-сущностей.

Символ	Описание	Имя сущности (Entity Name)	Номер сущности (Entity Number)
« »	неразрывный пробел	` `	` `
«<»	меньше	`<`	`<`
«>»	больше	`>`	`>`
«&»	амперсанд	`&`	`&`
«–»	длинное тире	`—`	`—`
«—»	короткое тире	`–`	`–`
«©»	знак авторского права	`©`	`©`
«®»	зарегистрированный знак	`®`	`®`
«™»	товарный знак	`™`	`™`
«☎»	телефон	`☎`	`☎`

Таким образом, чтобы написать:

© 2016 Stack Exchange Inc.

используется следующий HTML-код:

```
<b>&copy; 2016 Stack Exchange Inc.</b>
```

Глава 41.
ARIA

41.1. role="presentation"

Элемент, чья неявная семантика роли не будет отображаться в API доступности:

```
<div style="float:left;">Some content on the left.</div> <!-- Некоторый контент слева. -->
<div style="float:right;">Some content on the right</div> <!-- Некоторый контент справа. -->
<div role="presentation" style="clear:both;"></div> <!-- Используется только для очистки плавающих элементов. -->
```

41.2. role="alert"

Сообщение с важной, обычно срочной информацией.

```
<div role="alert" aria-live="assertive">Your session will expire in 60 seconds.</div>
<!-- Ваш сеанс истечет через 60 секунд. -->
```

> Обратите внимание, что включены и `role="alert"`, и `aria-live="assertive"` одновременно. Эти атрибуты являются синонимами, но некоторые программы чтения с экрана поддерживают только один из них. Используя оба одновременно, мы максимизируем шансы, что живая область будет работать как ожидалось.
>
> Источник — Хейдон Пикеринг, «Некоторые практические примеры ARIA» (https://heydonworks.com/practical_aria_examples/#offline-alert).



41.3. role="alertdialog"

Тип диалога, содержащего предупреждающее сообщение, где начальный фокус переходит на элемент внутри диалога.

```
<div role="alertdialog">
    <h1>Warning</h1> <!-- Предупреждение -->
    <div role="alert">Your session will expire in 60 seconds.</
div> <!-- Ваш сеанс истечет через 60 секунд. -->
</div>
```

41.4. role="application"

Область, объявленная как веб-приложение, в отличие от веб-документа. В этом примере приложение представляет собой простой калькулятор, который может складывать два числа:

```
<div role="application">
    <h1>Calculator</h1> <!-- Калькулятор -->
    <input id="num1" type="text"> + <input id="num2" type="text"> =
    <span id="result"></span>
</div>
```

41.5. role="article"

Раздел страницы, состоящий из композиции, которая образует независимую часть документа, страницы или сайта.

> Установка роли ARIA и/или атрибута `aria-*`, который соответствует неявной семантике ARIA по умолчанию, избыточна и не рекомендуется, так как эти свойства уже установлены браузером.
>
> Источник: https://www.w3.org/TR/html-aria/#sec-strong-native-semantics.



```
<article>
    <h1>My first article</h1> <!-- Моя первая статья -->
    <p>Lorem ipsum...</p>
</article>
```

Вы можете использовать `role="article"` на несемантических элементах (не рекомендуется, невалидно):

```
<div role="article">
    <h1>My first article</h1> <!-- Моя первая статья -->
    <p>Lorem ipsum...</p>
</div>
```

Запись W3C для `role="article"` (https://html.spec.whatwg.org/multipage/sections.html#the-article-element).



41.6. role="banner"

Область, содержащая в основном контент, связанный с сайтом, а не с конкретной страницей:

```
<div role="banner">
    <h1>My Site</h1> <!-- Мой сайт -->
    <ul>
        <li><a href="/">Home</a></li> <!-- Главная -->
        <li><a href="/about">About</a></li> <!-- О нас -->
        <li><a href="/contact">Contact</a></li> <!-- Контакты -->
    </ul>
</div>
```

41.7. role="button"

Элемент ввода, который позволяет выполнять действия, инициируемые пользователем, при нажатии:

```
<button role="button">Add</button> <!-- Добавить -->
```

41.8. role="cell"

Ячейка в табличном контейнере:

```
<table>
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <td role="cell">95</td>
        <td role="cell">14</td>
        <td role="cell">25</td>
    </tbody>
</table>
```

41.9. role="checkbox"

Элемент ввода, который может быть отмечен и имеет три возможных значения `true`, `false` или `mixed`:

```
<p>
    <input type="checkbox" role="checkbox" aria-checked="false">
```

```
    I agree to the terms <!-- Я согласен с условиями -->
</p>
```

41.10. role="columnheader"

Ячейка, содержащая заголовочную информацию для столбца:

```
<table role="grid">
    <thead>
        <tr>
            <th role="columnheader">Day 1</th> <!-- День 1 -->
            <th role="columnheader">Day 2</th> <!-- День 2 -->
            <th role="columnheader">Day 3</th> <!-- День 3 -->
        </tr>
    </thead>
    <tbody>
        <!-- и т.д. -->
    </tbody>
<table>
```

41.11. role="combobox"

Представление элемента выбора (`select`); обычно похоже на текстовое поле, где пользователи могут вводить текст для выбора опции или добавлять произвольный текст как новый элемент в список:

```
<input type="text" role="combobox" aria-expanded="false">
```

Обычно для реализации функциональности автодополнения или выбора из списка используется JavaScript.

41.12. role="complementary"

Вспомогательный раздел документа, предназначенный для дополнения основного контента на том же уровне иерархии DOM, но остающийся значимым при отделении от основного контента:

```
<div role="complementary">
    <h2>More Articles</h2> <!-- Дополнительные статьи -->
    <ul>
        <!-- и т.д. -->
    </ul>
</div>
```

41.13. role="contentinfo"

Большая воспринимаемая область, содержащая информацию о родительском документе:

```
<p role="contentinfo">
    Author: Albert Einstein<br> <!-- Автор: Альберт Эйнштейн -->
    Published: August 15, 1940 <!-- Опубликовано: 15 августа 1940 -->
</p>
```

41.14. role="definition"

Определение термина или концепции:

```
<span role="term" aria-labelledby="def1">Love</span> <!-- Любовь -->
<span id="def1" role="definition">an intense feeling of deep affection.</
span> <!-- интенсивное чувство глубокой привязанности. -->
```

41.15. role="dialog"

Диалог — это окно приложения, предназначенное для прерывания текущего процесса с целью запроса информации или ответа от пользователя:

```
<div role="dialog">
    <p>Are you sure?</p> <!-- Вы уверены? -->
    <button role="button">Yes</button> <!-- Да -->
    <button role="button">No</button> <!-- Нет -->
</div>
```

41.16. role="directory"

Список ссылок на элементы группы, например, статичное оглавление:

```
<ul role="directory">
    <li><a href="/chapter-1">Chapter 1</a></li> <!-- Глава 1 -->
    <li><a href="/chapter-2">Chapter 2</a></li> <!-- Глава 2 -->
    <li><a href="/chapter-3">Chapter 3</a></li> <!-- Глава 3 -->
</ul>
```

41.17. role="document"

Область, содержащая связанную информацию, объявленную как контент документа, в отличие от веб-приложения:

```
<div role="document">
    <h1>The Life of Albert Einstein</h1> <!-- Жизнь Альберта Эйн-
штейна -->
    <p>Lorem ipsum...</p>
</div>
```

41.18. role="form"

Лендмарк-область, содержащая коллекцию элементов и объектов, которые вместе образуют форму.

Использование семантически корректного HTML-элемента `<form>` подразумевает неявную семантику ARIA, поэтому использование `role="form"` не требуется, так как не следует применять противоречащую роль к элементу, который уже является семантическим, поскольку добавление роли переопределяет нативную семантику элемента.

> Установка роли ARIA и/или атрибута `aria-*`, который соответствует неявной семантике ARIA по умолчанию, избыточна и не рекомендуется, так как эти свойства уже установлены браузером.
>
> Источник: https://www.w3.org/TR/html-aria/#sec-strong-native-semantics.



```
<form action="">
    <fieldset>
        <legend>Login form</legend> <!-- Форма входа -->
        <div>
            <label for="username">Your username</label> <!-- Ваше
имя пользователя -->
            <input type="text" id="username" aria-
describedby="username-tip" required />
            <div role="tooltip" id="username-tip">Your username
is your email address</div> <!-- Ваше имя пользователя – это ваш
email. -->
        </div>
        <div>
            <label for="password">Your password</label> <!-- Ваш
пароль -->
            <input type="text" id="password" aria-
describedby="password-tip" required />
            <div role="tooltip" id="password-tip">Was emailed
to you when you signed up</div> <!-- Был отправлен вам по email
при регистрации. -->
        </div>
    </fieldset>
</form>
```

Вы можете использовать `role="form"` на несемантических элементах (не рекомендуется, невалидно):

```
<div role="form">
    <input type="email" placeholder="Your email address"> <!-- Ваш
email -->
    <button>Sign up</button> <!-- Зарегистрироваться -->
</div>
```

41.19. role="grid"

Сетка (grid) — это интерактивный элемент управления, содержащий ячейки табличных данных, организованные в строки и столбцы, как таблица.

```
<table role="grid">
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <!-- и т.д. -->
    </tbody>
</table>
```

41.20. role="gridcell"

Ячейка в сетке (grid) или древовидной сетке (treegrid):

```
<table role="grid">
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <tr>
            <td role="gridcell">17</td>
            <td role="gridcell">64</td>
            <td role="gridcell">18</td>
        </tr>
    </tbody>
<table>
```

41.21. role="group"

Набор объектов пользовательского интерфейса, которые не предназначены для включения в краткое содержание страницы или оглавление с помощью вспомогательных технологий:

```
<div role="group">
    <button role="button">Previous</button> <!-- Назад -->
```

```
    <button role="button">Next</button> <!-- Вперед -->
</div>
```

41.22. role="heading"

Заголовок для раздела страницы:

```
<h1 role="heading">Introduction</h1> <!-- Введение -->
<p>Lorem ipsum...</p>
```

41.23. role="img"

Контейнер для набора элементов, которые формируют изображение:

```
<figure role="img">
    <img alt="A cute cat." src="albert.jpg"> <!-- Милый кот. -->
    <figcaption>This is my cat, Albert.</figcaption> <!-- Это мой
кот, Альберт. -->
<figure>
```

41.24. role="link"

Интерактивная ссылка на внутренний или внешний ресурс, которая при активации вызывает переход пользователя к этому ресурсу.

> В большинстве случаев установка роли ARIA и/или атрибута aria-*, который соответствует неявной семантике ARIA по умолчанию, избыточна и не рекомендуется, так как эти свойства уже установлены браузером.
>
> Источник: https://www.w3.org/TR/html5/dom.html#aria-usage-note



41.25. role="list"

Группа неинтерактивных элементов списка:

```
<ul role="list">
    <li role="listitem">One</li> <!-- Один -->
    <li role="listitem">Two</li> <!-- Два -->
    <li role="listitem">Three</li> <!-- Три -->
</ul>
```

41.26. role="listbox"

Виджет, который позволяет пользователю выбрать один или несколько элементов из списка:

```
<ul role="listbox">
    <li>One</li> <!-- Один -->
```

```
    <li>Two</li> <!-- Два -->
    <li>Three</li> <!-- Три -->
</ul>
```

Обычно для реализации функциональности множественного выбора используется JavaScript.

41.27. role="listitem"

Отдельный элемент в списке или директории:

```
<ul role="list">
    <li role="listitem">One</li> <!-- Один -->
    <li role="listitem">Two</li> <!-- Два -->
    <li role="listitem">Three</li> <!-- Три -->
</ul>
```

41.28. role="log"

Тип живой области (live region), в которую новая информация добавляется в осмысленном порядке, а старая информация может исчезать:

```
<ul role="log">
    <li>User 1 logged in.</li> <!-- Пользователь 1 вошел
в систему. -->
    <li>User 2 logged in.</li> <!-- Пользователь 2 вошел
в систему. -->
    <li>User 1 logged out.</li> <!-- Пользователь 1 вышел
из системы. -->
</ul>
```

41.29. role="main"

Основное содержимое документа:

```
<!-- header & nav здесь -->
<div role="main">
    <p>Lorem ipsum...</p>
</div>
<!-- footer здесь -->
```

41.30. role="marquee"

Тип живой области (live region), в которой несущественная информация часто изменяется:

```
<ul role="marquee">
    <li>Dow +0.26%</li> <!-- Dow +0.26% -->
```

```
    <li>Nasdaq +0.54%</li> <!-- Nasdaq +0.54% -->
    <li>S&amp;P +0.44%</li> <!-- S&P +0.44% -->
</ul>
```

41.31. role="math"

Контент, представляющий математическое выражение:

```
<img role="math" alt="y=mx+b" src="slope.png"> <!-- y=mx+b -->
```

41.32. role="menu"

Тип виджета, который предоставляет пользователю список вариантов выбора:

```
<ul role="menu">
    <li role="menuitem">New</li> <!-- Новый -->
    <li role="menuitem">Open</li> <!-- Открыть -->
    <li role="menuitem">Save</li> <!-- Сохранить -->
    <li role="menuitem">Close</li> <!-- Закрыть -->
</ul>
```

41.33. role="menubar"

Представление меню, которое обычно остается видимым и отображается горизонтально:

```
<ul role="menubar">
    <li role="menuitem">File</li> <!-- Файл -->
    <li role="menuitem">Edit</li> <!-- Редактировать -->
    <li role="menuitem">View</li> <!-- Вид -->
    <li role="menuitem">Help</li> <!-- Помощь -->
</ul>
```

41.34. role="menuitem"

Вариант выбора в группе опций, содержащейся в меню или строке меню:

```
<ul role="menubar">
    <li role="menuitem">File</li> <!-- Файл -->
    <li role="menuitem">Edit</li> <!-- Редактировать -->
    <li role="menuitem">View</li> <!-- Вид -->
    <li role="menuitem">Help</li> <!-- Помощь -->
</ul>
```

41.35. role="menuitemcheckbox"

Элемент меню с возможностью выбора, который может иметь три значения `true`, `false` или `mixed`:

```
<ul role="menu">
    <li role="menuitem">Console</li> <!-- Консоль -->
    <li role="menuitem">Layout</li> <!-- Макет -->
    <li role="menuitemcheckbox" aria-checked="true">Word wrap</
li> <!-- Перенос слов -->
</ul>
```

41.36. role="menuitemradio"

Элемент меню с возможностью выбора в группе элементов типа `menuitemradio`, из которых одновременно может быть выбран только один:

```
<ul role="menu">
    <li role="menuitemradio" aria-checked="true">Left</li> <!--
Слева -->
    <li role="menuitemradio" aria-checked="false">Center</li>
<!-- По центру -->
    <li role="menuitemradio" aria-checked="false">Right</li> <!--
Справа -->
</ul>
```

41.37. role="navigation"

Коллекция навигационных элементов (обычно ссылок) для навигации по документу или связанным документам:

```
<ul role="navigation">
    <li><a href="/">Home</a></li> <!-- Главная -->
    <li><a href="/about">About</a></li> <!-- О нас -->
    <li><a href="/contact">Contact</a></li> <!-- Контакты -->
</ul>
```

41.38. role="note"

Раздел, содержимое которого является дополнительным или вспомогательным по отношению к основному контенту ресурса:

```
<p>Lorem ipsum...</p>
<p>Lorem ipsum...</p>
<p role="note">Lorem ipsum...</p> <!-- Примечание -->
```

41.39. role="option"

Выбираемый элемент в списке выбора:

```
<ul role="listbox">
    <li role="option">Option 1</li> <!-- Опция 1 -->
    <li role="option">Option 2</li> <!-- Опция 2 -->
    <li role="option">Option 3</li> <!-- Опция 3 -->
</ul>
```

41.40. role="progressbar"

Элемент, отображающий статус выполнения задач, которые занимают длительное время:

```
<progress role="progressbar" value="25" max="100">25%</progress>
<!-- 25% -->
```

41.41. role="radio"

Элемент выбора в группе радиокнопок, из которых одновременно может быть выбрана только одна:

```
<div role="radiogroup">
    <input role="radio" type="radio" aria-checked="true"> One<br>
<!-- Один -->
    <input role="radio" type="radio" aria-checked="false">
Two<br> <!-- Два -->
    <input role="radio" type="radio" aria-checked="false"> Three
<!-- Три -->
</div>
```

41.42. role="region"

Большая воспринимаемая область веб-страницы или документа, которую автор считает достаточно важной для включения в краткое содержание страницы или оглавление, например, область страницы, содержащая статистику живого спортивного события:

```
<div role="region">
    Home team: 4<br> <!-- Домашняя команда: 4 -->
    Away team: 2 <!-- Гостевая команда: 2 -->
</div>
```

41.43. role="radiogroup"

Группа радиокнопок:

```
<div role="radiogroup">
    <input role="radio" type="radio" aria-checked="true"> One<br>
<!-- Один -->
```

```
    <input role="radio" type="radio" aria-checked="false">
Two<br> <!-- Два -->
    <input role="radio" type="radio" aria-checked="false"> Three
<!-- Три -->
</div>
```

41.44. role="row"

Строка ячеек в табличном контейнере:

```
<table>
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <tr role="row">
            <!-- и т.д. -->
        </tr>
    </tbody>
</table>
```

41.45. role="rowgroup"

Группа, содержащая одну или несколько строк (row) в сетке (grid):

```
<table>
    <thead role="rowgroup">
        <!-- и т.д. -->
    </thead>
    <tbody role="rowgroup">
        <!-- и т.д. -->
    </tbody>
</table>
```

41.46. role="rowheader"

Ячейка, содержащая заголовочную информацию для строки в сетке (grid):

```
<table role="grid">
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <tr>
            <th role="rowheader">Day 1</th> <!-- День 1 -->
```

```
            <td>65</td>
        </tr>
        <tr>
            <th role="rowheader">Day 2</th> <!-- День 2 -->
            <td>74</td>
        </tr>
    </tbody>
</table>
```

41.47. role="scrollbar"

Графический объект, управляющий прокруткой содержимого в пределах области просмотра, независимо от того, отображается ли содержимое полностью в этой области:

```
<div id="content1">Lorem ipsum...</div>
<div
    role="scrollbar"
    aria-controls="content1"
    aria-orientation="vertical"
    aria-valuemax="100"
    aria-valuemin="0"
    aria-valuenow="25">
    <div class="scrollhandle"></div>
</div>
```

41.48. role="search"

Landmark-область, содержащая набор элементов и объектов, которые вместе образуют поисковый механизм:

```
<div role="search">
    <input role="searchbox" type="text">
    <button role="button">Search</button> <!-- Поиск -->
</div>
```

41.49. role="searchbox"

Тип текстового поля, предназначенного для указания критериев поиска:

```
<div role="search">
    <input role="searchbox" type="text">
    <button role="button">Search</button> <!-- Поиск -->
</div>
```

41.50. role="separator"

Разделитель, который отделяет и различает разделы контента или группы элементов меню:

```
<p>Lorem ipsum...</p>
<hr role="separator">
<p>Lorem ipsum...</p>
```

41.51. role="slider"

Элемент ввода, позволяющий пользователю выбрать значение из заданного диапазона:

```
<div
    role="slider"
    aria-valuemax="100"
    aria-valuemin="0"
    aria-valuenow="25">
    <div class="sliderhandle"></div>
</div>
```

41.52. role="spinbutton"

Форма диапазона, в которой пользователь выбирает из дискретных значений:

```
<input
    role="spinbutton"
    aria-valuemax="100"
    aria-valuemin="0"
    aria-valuenow="25"
    type="number"
    value="25">
```

41.53. role="status"

Контейнер, содержимое которого представляет собой справочную информацию для пользователя, но не настолько важную, чтобы оправдать использование предупреждения (alert). Часто, но не всегда, отображается в виде строки состояния:

```
<div role="status">Online</div> <!-- В сети -->
```

41.54. role="switch"

Тип флажка (checkbox), представляющий значения вкл/выкл, в отличие от значений отмечено/не отмечено:

```
<select role="switch" aria-checked="false">
    <option>On</option> <!-- Вкл -->
    <option selected>Off</option> <!-- Выкл -->
</select>
```

41.55. role="tab"

Группирующая метка, предоставляющая механизм для выбора содержимого вкладки, которое будет показано пользователю:

```
<ul role="tablist">
    <li role="tab">Introduction</li> <!-- Введение -->
    <li role="tab">Chapter 1</li> <!-- Глава 1 -->
    <li role="tab">Chapter 2</li> <!-- Глава 2 -->
</ul>
```

41.56. role="table"

Раздел, содержащий данные, организованные в строки и столбцы. Роль `table` предназначена для табличных контейнеров, которые не являются интерактивными:

```
<table role="table">
    <thead>
        <!-- и т.д. -->
    </thead>
    <tbody>
        <!-- и т.д. -->
    </tbody>
</table>
```

41.57. role="tablist"

Список элементов вкладок (tab), которые ссылаются на элементы `tabpanel`:

```
<ul role="tablist">
    <li role="tab">Introduction</li> <!-- Введение -->
    <li role="tab">Chapter 1</li> <!-- Глава 1 -->
    <li role="tab">Chapter 2</li> <!-- Глава 2 -->
</ul>
```

41.58. role="tabpanel"

Контейнер для ресурсов, связанных с вкладкой, где каждая вкладка содержится в `tablist`:

```
<ul role="tablist">
    <li role="tab">Introduction</li> <!-- Введение -->
    <li role="tab">Chapter 1</li> <!-- Глава 1 -->
    <li role="tab">Chapter 2</li> <!-- Глава 2 -->
</ul>
<div role="tabpanel">
    <!-- и т.д. -->
</div>
```

41.59. role="textbox"

Элемент ввода, позволяющий вводить произвольный текст в качестве значения:

```
<textarea role="textbox"></textarea>
```

41.60. role="timer"

Тип активной области (live region), содержащей числовой счетчик, который указывает количество прошедшего времени с начальной точки или оставшееся время до конечной точки:

```
<p>
    <span role="timer">60</span> seconds remaining. <!-- Осталось
60 секунд. -->
</p>
```

41.61. role="toolbar"

Коллекция часто используемых кнопок функций, представленных в компактной визуальной форме:

```
<ul role="toolbar">
    <li><img alt="New" src="new.png"></li> <!-- Новый -->
    <li><img alt="Open" src="open.png"></li> <!-- Открыть -->
    <li><img alt="Save" src="save.png"></li> <!-- Сохранить -->
    <li><img alt="Close" src="close.png"></li> <!-- Закрыть -->
</ul>
```

41.62. role="tooltip"

Контекстное всплывающее окно, которое отображает описание элемента:

```
<span aria-describedby="slopedesc">Slope</span> <!-- Наклон -->
<div role="tooltip" id="slopedesc">y=mx+b</div> <!-- y=mx+b -->
```

Обычно подсказка (tooltip) скрыта. С использованием JavaScript подсказка отображается с задержкой, когда пользователь наводит курсор на элемент, который она описывает.

41.63. role="tree"

Тип списка, содержащий вложенные группы, которые можно сворачивать и разворачивать.

```
<ul role="tree">
    <li role="treeitem">
        Part 1 <!-- Часть 1 -->
        <ul>
            <li role="treeitem">Chapter 1</li> <!-- Глава 1 -->
            <li role="treeitem">Chapter 2</li> <!-- Глава 2 -->
            <li role="treeitem">Chapter 3</li> <!-- Глава 3 -->
        </ul>
    </li>
    <li role="treeitem">
        Part 2 <!-- Часть 2 -->
        <ul>
            <li role="treeitem">Chapter 4</li> <!-- Глава 4 -->
            <li role="treeitem">Chapter 5</li> <!-- Глава 5 -->
            <li role="treeitem">Chapter 6</li> <!-- Глава 6 -->
        </ul>
    </li>
    <li role="treeitem">
        Part 3 <!-- Часть 3 -->
        <ul>
            <li role="treeitem">Chapter 7</li> <!-- Глава 7 -->
            <li role="treeitem">Chapter 8</li> <!-- Глава 8 -->
            <li role="treeitem">Chapter 9</li> <!-- Глава 9 -->
        </ul>
    </li>
</ul>
```

41.64. role="treegrid"

Сетка, строки которой можно разворачивать и сворачивать, как в дереве (tree).

41.65. role="treeitem"

Элемент списка (опция) в дереве (tree). Это элемент внутри дерева, который может быть развернут или свернут, если он содержит вложенную группу элементов (treeitems):

```
<ul role="tree">
    <li role="treeitem">
```

```
        Part 1 <!-- Часть 1 -->
        <ul>
            <li role="treeitem">Chapter 1</li> <!-- Глава 1 -->
            <li role="treeitem">Chapter 2</li> <!-- Глава 2 -->
            <li role="treeitem">Chapter 3</li> <!-- Глава 3 -->
        </ul>
    </li>
    <li role="treeitem">
        Part 2 <!-- Часть 2 -->
        <ul>
            <li role="treeitem">Chapter 4</li> <!-- Глава 4 -->
            <li role="treeitem">Chapter 5</li> <!-- Глава 5 -->
            <li role="treeitem">Chapter 6</li> <!-- Глава 6 -->
        </ul>
    </li>
    <li role="treeitem">
        Part 3 <!-- Часть 3 -->
        <ul>
            <li role="treeitem">Chapter 7</li> <!-- Глава 7 -->
            <li role="treeitem">Chapter 8</li> <!-- Глава 8 -->
            <li role="treeitem">Chapter 9</li> <!-- Глава 9 -->
        </ul>
    </li>
</ul>
```

Благодарности

Огромное спасибо всем участникам Stack Overflow Documentation, которые помогли в создании этого контента.

Дополнительные изменения или предложения можно отправить на адрес web@petercv.com для публикации или обновления материалов.

AA2992 Глава 12
Abhishek Pandey Главы 17 и 33
Abrar Jahin Глава 4
Adjit Глава 29
ahmednawazbutt Глава 1
Al.G. Главы 2, 6 и 17
albert Глава 8
Alex Глава 15
Alexander Wigmore Глава 33
Alexandre N. Главы 8, 13, 15, 28 и 29
Ali Almoullim Главы 18 и 27
Amanda Ahn Глава 1
amflare Главы 1 и 27
Amitay Stern Главы 1 и 17
andreaem Глава 13
andreas Глава 31
Andrew Brooke Глава 20
Angelos Chalaris Главы 2, 10, 17 и 33
Ani Menon Главы 2, 3, 9, 17, 18 и 33
Anil Глава 20
animuson Главы 1, 2, 5, 6, 7, 8, 9, 15, 17, 18, 27, 29, 33, 37 и 40
Anselm Главы 6, 12 и 13
Anthony Pham Глава 6
Aown Muhammad Глава 18
Araknid Глава 19
Aravind Suresh Глава 12
Ashwin Ramaswami Глава 9
Axe Глава 33
Bálint Глава 33
ban17 Глава 6
bdkopen Главы 9, 14 и 33
Becca Глава 37
Ben Rhys Глава 5
bhansa Глава 17
Bhavya Singh Глава 41
Bookeater Глава 33
Boris Глава 33
Boysenb3rry Глава 15
brandaemon Глава 10
Caleb Kleveter Главы 10 и 15
Callan Heard Главы 1 и 12
Cedric Zoppolo Глава 8
Charles Глава 13
Chris Главы 2 и 19
Chris Rutherfurd Глава 12
ChrisD Главы 1, 9, 24, 29 и 31
Christian Ternus Глава 14
Christophe Strobbe Глава 17
cone56 Главы 12 и 32
Content Solutions Главы 2, 19, 26, 35 и 36
Cullub Глава 18
D M Глава 19
Daniel Глава 17
Daniel Käfer Глава 7
DannyTheDev Глава 17
Dave Everitt Глава 1
DawnPaladin Главы 6, 12 и 17
Dipen Shah Глава 17
dippas Глава 3
Domenic Главы 12 и 33
Druzion Глава 17

Emanuel Vintilă Глава 29
Emil Главы 5, 6 и 10
Evan Глава 3
Farhad Глава 38
feeela Главы 12 и 25
FelipeAls Глава 30
FlyingPiMonster Глава 6
gabe3886 Глава 17
Gabriel Chi Hong Lee Главы 15 и 18
Gal Ratzkin Глава 19
geek1011 Главы 1 и 33
geeksal Глава 17
Gerold Broser Главы 1 и 30
GoatsWearHats Глава 10
Grace Note Глава 7
Grant Palin Глава 8
gustavohenke Глава 5
Gytis Tenovimas Глава 17
H Mirza Глава 1
H. Pauwelyn Глава 1
Habel Philip Глава 33
Hafidz Ilham Aji Permana Глава 33
Hanif Formoly Глава 20
heerfk Глава 2
Henrique Barcelos Главы 12 и 17
Huy Nguyen Глава 17
Infuzion Главы 3 и 12
insertusernamehere Глава 6
intboolstring Глава 1
Isak Combrinck Главы 25 и 30
ivn Глава 15
J F Главы 5 и 8
j08691 Главы 8 и 18
jhnance Глава 10
jhoanna Глава 33
JHS Главы 6, 8, 9, 12 и 19
jkdev Глава 9
joe_young Главы 1, 3 и 8
John Slegers Главы 10 и 17
Jojodmo Глава 6
Jon Ericson Глава 7
Jonathan Lam Главы 1, 6, 12 и 16
Jones Joseph Главы 6 и 14
Kake_Fisk Глава 15
kcpike Глава 7
kelvinelove Главы 1 и 33
Kimmax Глава 18
Lahiru Ashan Глава 17
Leonidas Menendez Глава 1
Luca langella Глава 18
Luca Putzu Глава 10
Marjorie Pickard Глава 13
Marvin Главы 7 и 17
Matas Vaitkevicius Главы 6, 7, 12 и 17
Matt Глава 1
Maximillian Laumeister Главы 15 и 17
MervS Главы 13 и 40
Michael Moriarty Глава 33
Michael_B Глава 10
mnoronha Глава 1
Mohd Samir Khan Глава 15
morewry Глава 17
Mosh Feu Глава 27
Mottie Глава 8
Mr Lister Глава 8
Mr. Alien Глава 33
Muntasir Глава 30
MySpeed Глава 19
m_callens Главы 22 и 33
nalply Глава 20
Natalie Глава 10
Nathan Tuggy Глава 6
Nhan Главы 12 и 15
Niek Brouwer Глава 18
Nijin22 Глава 8
Nil Llisterri Глава 17
Nishchay Главы 1 и 33
NoobCoder Глава 17
Ojen Глава 29
Ortomala Lokni Главы 17 и 33
Paresh Maghodiya Глава 39
Paul Sweatte Глава 41
Persijn Глава 1
Peter L. Глава 5

Pi Programs Глава 17
pinjasaur Глава 2
platy11 Глава 7
Pranav Глава 6
Prateek Главы 6, 7, 8 и 17
PrAtik Lochawala Глава 8
Praveen Kumar Глава 8
Psaniko Главы 17 и 36
Pseudonym Patel Главы 3 и 7
Racil Hilan Глава 11
rajarshig Глава 11
RamenChef Главы 16 и 17
Ranjit Singh Глава 17
Raystafarian Главы 5 и 6
Richard Hamilton Главы 7, 10, 26 и 32
Robert Columbia Глава 6
Roko C. Buljan Главы 13, 17, 22 и 32
Safoor Safdar Глава 33
sasha Глава 3
SeinopSys Глава 17
Senjuti Mahapatra Глава 33
Shannon Young Главы 20 и 41
Sharavnan Kv Глава 17
Shiva Глава 25
Shivangi Chaurasia Главы 15 и 17
Sildoreth Главы 8 и 13
Simone Carletti Глава 10
SJDS Глава 17
stack Глава 40
Stephen Leppik Главы 17 и 33
Stewartside Главы 7 и 17
Steyn van Esveld Глава 6
Sumner Evans Глава 2
Sunny R Gupta Главы 1 и 17
SuperStormer Главы 13, 19 и 34
sv3k Глава 17
svarog Глава 8
Ted Goas Главы 8 и 33
the12 Главы 1 и 17
think123 Глава 17
Thomas Gerot Глава 24
Thomas Landauer Глава 8
thouusten Глава 14
Timon Глава 17
Timothy Miller Глава 31
tmg Главы 2, 7 и 17
Tom Johnson Глава 7
tonethar Глава 32
Tot Zam Главы 17 и 27
Travis Глава 41
Trevor Clarke Глава 32
trungk18 Глава 17
Tyler Zika Глава 10
Ulrich Schwarz Глава 9
Undo Глава 17
unor Главы 1, 5, 6, 7, 12, 30, 33, 34, 35 и 37
user3130333 Глава 1
user5389107 Глава 17
V4karian Глава 12
Valor Naram Главы 1, 4 и 38
vkopio Глава 3
vladdobra Главы 12 и 17
w5m Глава 31
Wolfgang Главы 6, 9 и 15
xims Глава 33
Yasir T Глава 11
Yossi Aharon Главы 20 и 25
Zack Глава 7
Zange Глава 37
Zaz Глава 1
zer00ne Главы 8 и 23
zygimantus Глава 1
Zze Глава 1
zzzzBov Глава 17

16+

Справочное издание
Анықтамалық басылым

Серия «Программирование. Оффер на миллион»

HTML5 ДЛЯ ПРОФЕССИОНАЛОВ РУКОВОДСТВО ПО РАЗРАБОТКЕ В ПРИМЕРАХ ОТ СООБЩЕСТВА STACK OVERFLOW

Ответственный за выпуск: *И. Резько*
Ответственный редактор: *А. Ходякова*
Оформление обложки: *С. Дарсич*

Подписано в печать 14.10.2025.
Формат 70x100 1/16. Бумага офсетная. Печать офсетная.
Гарнитура Noto Serif SemiCondensed.
Усл. печ. л. 13. Тираж 1500 экз. Заказ №

Общероссийский классификатор продукции ОК-034-2014 (КПЕС 2008);
58.11.1 — книги, брошюры печатные.

Изготовлено в 2025 г.
Произведено в Российской Федерации.
Изготовитель: ООО «Издательство АСТ».
129085, Российская Федерация, г. Москва, Звёздный бульвар, дом 21,
строение 1, комната 705, пом. I, 7 этаж.
Адрес места осуществления деятельности по изготовлению продукции:
123112, Российская Федерация, г. Москва, Пресненская набережная, д. 6, стр. 2,
Деловой комплекс «Империя», 14, 15 этаж.
Наш электронный адрес: ask@ast.ru
Наш сайт: www.ast.ru
Интернет-магазин: www.book24.ru

Өндіруші: «Издательство АСТ» ЖШҚ 129085, Ресей Федерациясы,
Мәскеу, Звёздный бульвары, 21-үй, 1-құрылыс, 705-бөлме, I үй-жай, 7-қабат.
Өнім өндіру қызметін жүзеге асыру мекенжайы:
123112, Ресей Федерациясы, Мәскеу, Пресненская жағ., 6-үй, 2-құр.,
«Империя» іскерлік кешені, 14, 15-қабат.
Біздің электрондық мекенжайымыз: www.ast.ru E-mail: ask@ast.ru
Интернет-магазин: www.book24.ru Интернет-дүкен: www.book24.kz
Импортер в Республику Казахстан, дистрибьютор и представитель по приёму
претензий на продукцию в Республике Казахстан: ТОО «РДЦ-Алматы».
г. Алматы, ул. Домбровского, 3«а», литер Б, офис 1.
Қазақстан Республикасына импорттаушы дистрибьютор және
Қазақстан Республикасында өнімге шағымдар қабылдау жөніндегі өкіл:
«РДЦ-Алматы» ЖШС. Алматы қ., Домбровский көш., 3«а», Б литері, офис 1.
Тел.: 8(727) 2 51 59 90,91, факс: 8 (727) 251 59 92 ішкі 107;
E-mail: RDC-Almaty@eksmo.kz, www.book24.kz
Өндірілген жылы: 2025. Өнімнің жарамдылық мерзімі шектелмеген.
Сертификаттауға жатпайды.
Ресей Федерациясында өндірілген.

ДЛЯ ПРОФЕССИОНАЛОВ

HTML5

Эта книга — не очередной справочник по разметке, а реальное практическое руководство, где каждый пример сразу же пригодится при верстке веб-страниц. Ведь здесь рассматривается HTML, который действительно необходим в решении конкретных задач — без перегрузки и интуитивных догадок.

ДЛЯ КОГО ЭТА КНИГА?

- Для тех, кто начинает верстать с нуля и желает получить быстрый, но качественный результат.
- Для разработчиков, которым необходимо освежить знания или закрыть пробелы.
- Для всех, кто хочет писать читаемый, современный и надежный HTML-код.

ПОЧЕМУ СТОИТ ВЫБРАТЬ ИМЕННО ЭТО РУКОВОДСТВО?

- **Только практическая ценность** — никакой теории ради теории, весь материал направлен на решение реальных задач.
- **HTML от А до Я** — формы, таблицы, мультимедиа, доступность, семантика и даже SVG с <canvas>.
- **Все понятно без специальной подготовки** — объяснения простым языком, даже если вы впервые видите тег <section>.
- Приведенные **примеры можно сразу использовать в коде** — и в небольшом собственном проекте, и в масштабной работе.
- **Легко найти нужные сведения** — удобная навигация по главам и логичная структура.

НАЧНИТЕ РАБОТАТЬ С HTML БЕЗ ТЕОРЕТИЧЕСКИХ ПРЕМУДРОСТЕЙ! ЭТА КНИГА — ВАШ НАДЕЖНЫЙ ПРАКТИЧЕСКИЙ ПУТЕВОДИТЕЛЬ ПО МИРУ ВЕБ-РАЗРАБОТКИ!



КНИГИ ДЛЯ ЛЮБОГО НАСТРОЕНИЯ ЗДЕСЬ:
www.ast.ru / www.book24.ru
vk.com/izdatelstvoast
ok.ru/izdatelstvoast

ИЗДАТЕЛЬСКАЯ ГРУППА АСТ

ISBN 978-5-17-178629-8