Разработка клиентской части пользовательского интерфейса

*Сгенерено Midjourney по запросу **“Разработку клиентской части пользовательского интерфейса”**

Введение

- Для чего читать лекции, если все можно посмотреть в интернете?
  (справочники, обучающий видео, открытые курсы, платные курсы, документация)
- Для чего нужна эта дисциплина?
- Как называется дисциплина?
- Что нужно знать?
- Литература

Что нужно знать?

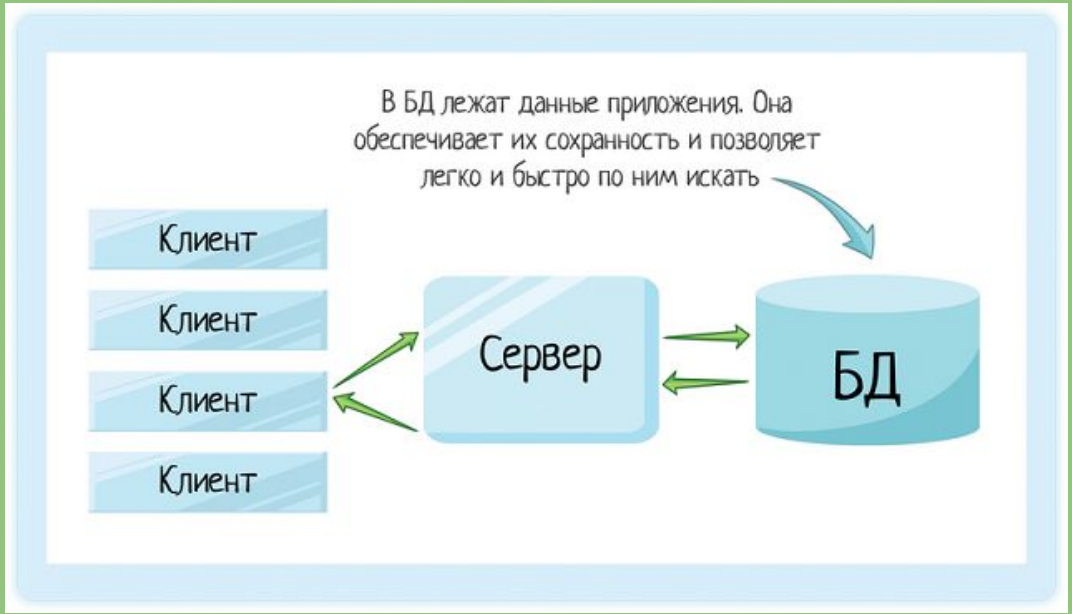
- Развитие профессии
- Git
- HTML
- CSS
- Javascript
- REST API
- Фреймворки JavaScript (React, Angular, Vue и т. д.)

Литература

- https://developer.mozilla.org/
- https://doka.guide/
- https://learn.javascript.ru/
- Серии / Head First (O'Reilly)

Специфика

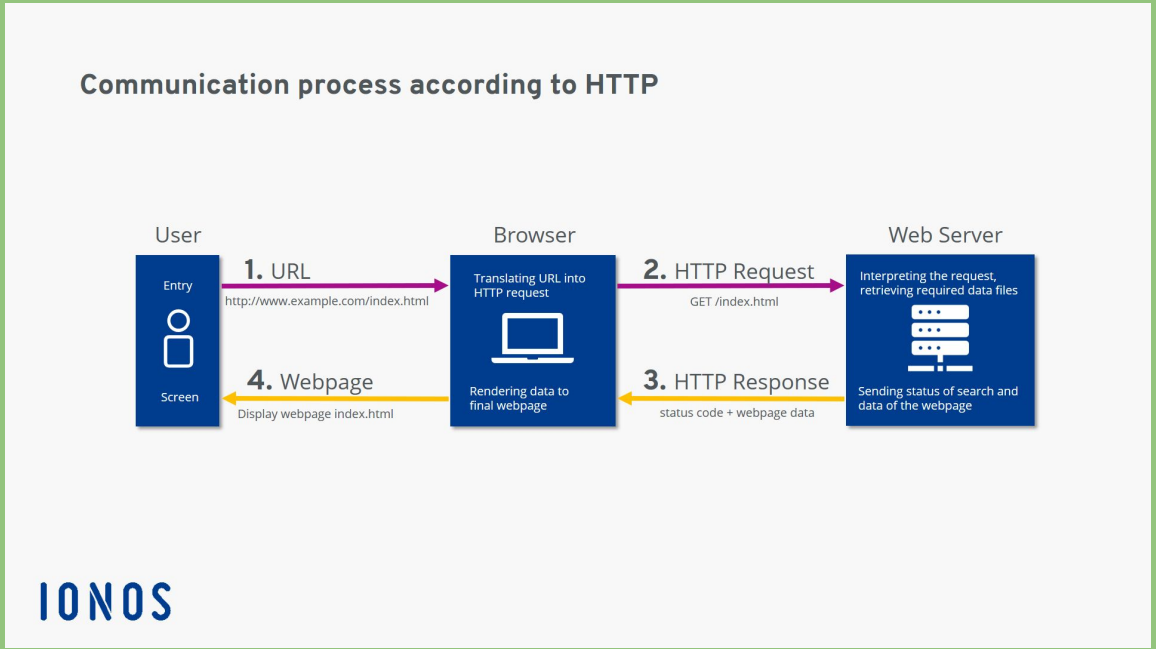
- Почему web-разработка?
- Сильные и слабые стороны
- Альтернативы
- Клиент-серверная архитектура



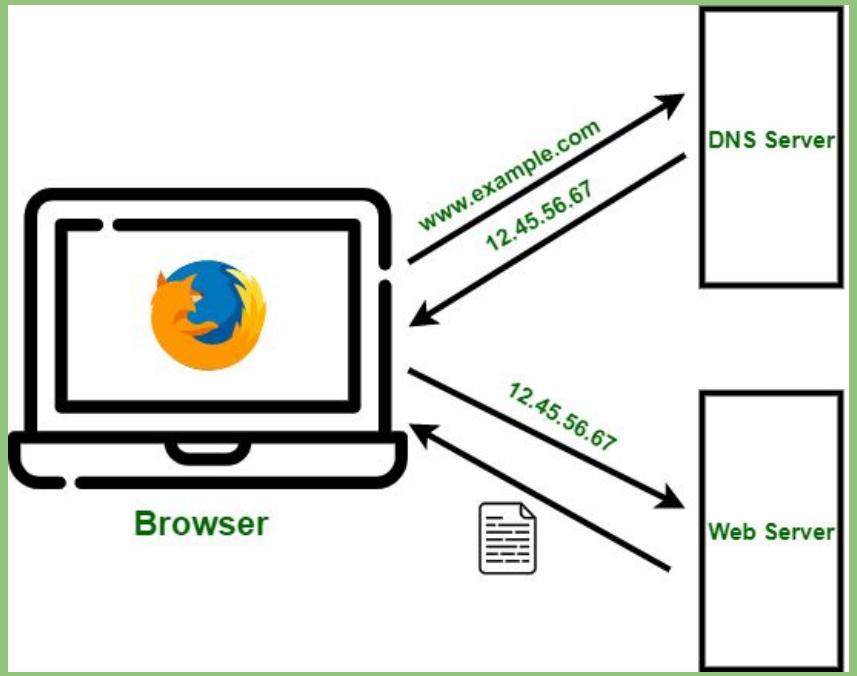
Как мы пришли туда, куда пришли

Что происходит, когда мы переходим на сайт?

- Браузер
- Домен
- DNS
- IP адрес
- Хостинг
- Обработка ответа

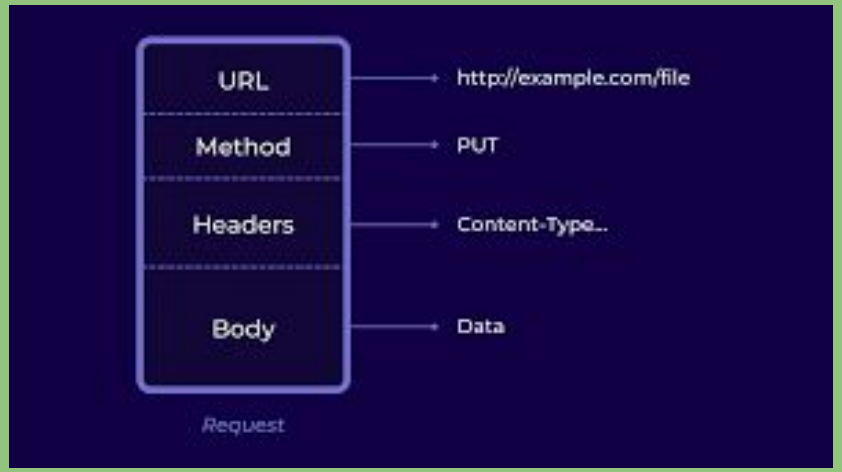


DNS


www.example.com
12.45.56.67
DNS Server
12.45.56.67
Web Server
Browser


Клиент-серверная архитектура - двухуровневая (базовая)
Клиент
Запрос
Ответ
Сервер
Уровень презентации (отображения пользователю)
Уровень логики и обработки данных

HTTP



HTTP Request

Method URL Protocol Version

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
blank line
```

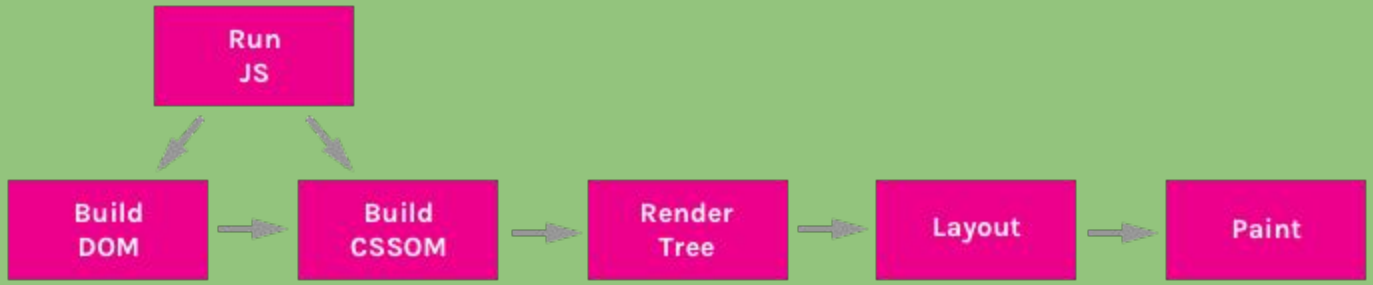
Headers

Body (optional)

Критический путь рендеринга

Существует 6 этапов CRP:

- построение DOM-дерева
- построение CSSOM-дерева
- запуск JavaScript
- создание Render-дерева
- генерация раскладки
- отрисовка



Построение DOM-дерева

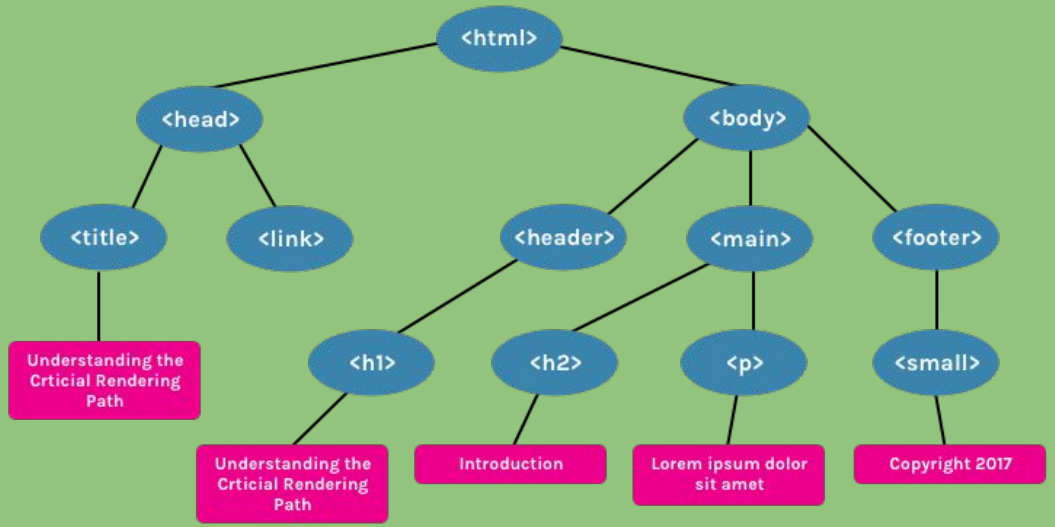
DOM (объектная модель документа) дерево это объект, представляющий полностью разобранную HTML-страницу. Начиная с корневого элемента <html>, узлы создаются для каждого элемента/текста на странице. Элементы, вложенные в другие элементы, представлены в виде дочерних узлов, и каждый узел содержит полный набор атрибутов для этого элемента.Например, элемент <a> будет иметь атрибут href, связанный с узлом.

Возьмём для примера такой документ:

```
<html>
<head>
  <title>
    Understanding the Critical Rendering Path
  </title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
      <h1>
        Understanding the Critical Rendering Path
    </h1>
  </header>
  <main>
      <h2>Introduction</h2>
      <p>Lorem ipsum dolor sit amet</p>
  </main>
  <footer>
      <small>Copyright 2017</small>
  </footer>
</body>
</html>
```

DOM-дерево

Из него будет построено такое DOM-дерево
Хорошая новость, касательно HTML, заключается в том, что он может быть исполнен по частям. Документ не должен быть загружен полностью для того, чтобы контент начал появляться на странице. Однако, другие ресурсы, такие как CSS и JavaScript, могут блокировать отрисовку страницы.



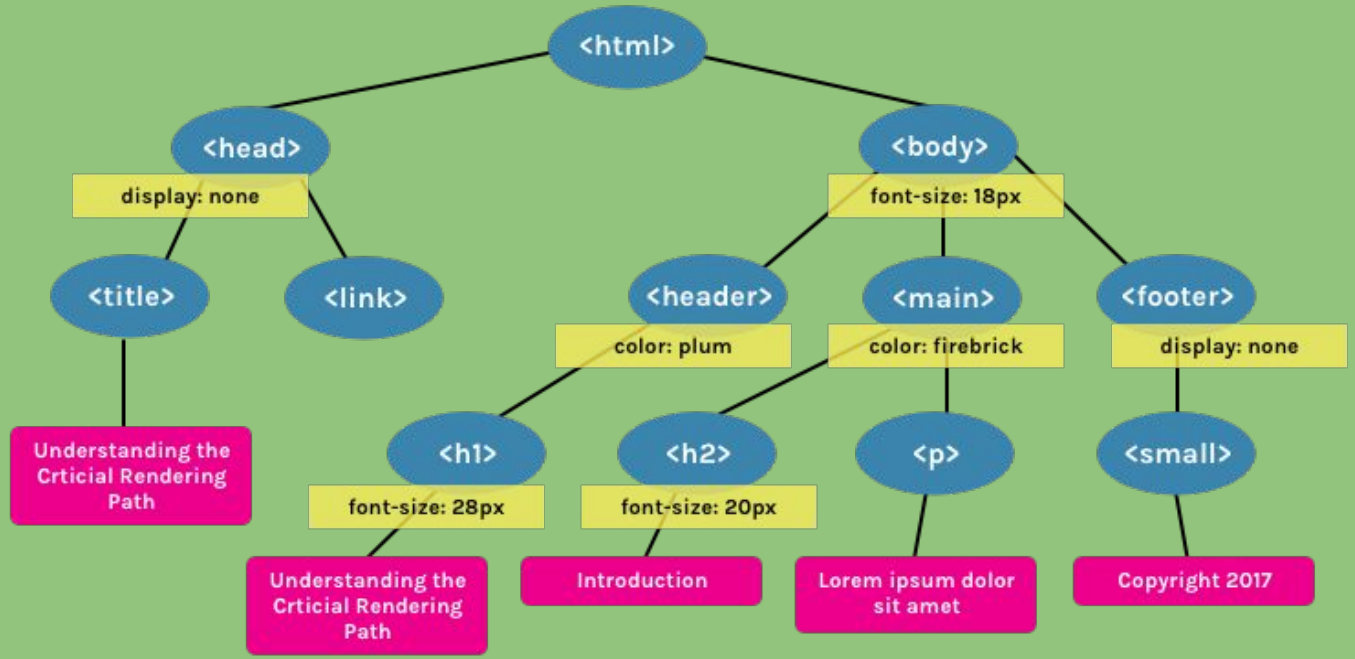
Построение CSSOM-дерева

CSSOM (объектная модель CSS) — это объект, представляющий стили, связанные с DOM. Он выглядит так же как DOM, но с соответствующими стилями для каждого узла. Не имеет значения были ли стили объявлены явно или наследуются.
В файле style.css, подключающемся в ранее упомянутом документе, мы имеем следующий набор стилей:

```
body { font-size: 18px; }

header { color: plum; }
h1 { font-size: 28px; }

main { color: firebrick; }
h2 { font-size: 20px; }

footer { display: none; }
```

Построение CSSOM-дерева

С его помощью получится следующее CSSOM-дерево:



Запуск JavaScript

JavaScript является блокирующим ресурсом для парсера. Это означает, что JavaScript блокирует разбор самого HTML-документа.

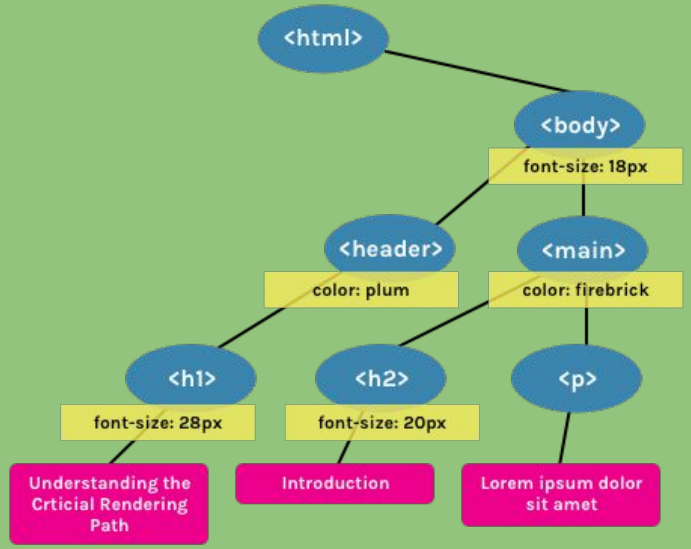
Когда парсер доходит до тега <script> (не важно внутренний он или внешний), он останавливается, забирает файл (если он внешний) и запускает его. Вот почему, если мы имеем JavaScript-файл, который ссылается на элементы документа, мы обязательно должны поместить его после их появления.

JavaScript можно загружать асинхронно, указав атрибут async, для того, чтобы избежать блокировки парсера.

<script async src="script.js">

Создание Render-дерева

Render-дерево — это совокупность DOM и CSSOM. Это дерево, которое даёт представление о том, что в конечном итоге будет отображено на странице. Это означает, что оно захватывает только видимый контент и не включает, например, элементы, которые были скрыты с помощью CSS-правила display: none.
На примерах DOM и CSSOM, представленных выше, будет построено такое Render-дерево:



Генерация раскладки

Раскладка — это то, что определяет размер видимой области документа (viewport), которая обеспечивает контекст для стилей CSS, зависимых от него, например, проценты или единицы вьюпорта.

Размер вьюпорта определяется метатэгом, находящемся в <head> документа или, если тэг не представлен, будет использовано стандартное значение вьюпорта шириною в 980 пикселей.

Например, наиболее частым значением для этого метатэга является размер, соответствующий с шириной устройства.

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

Если пользователь посещает веб-страницу с устройства, ширина которого, например 1000 пикселей, то размеры будут опираться на это значение. Половина видимой области будет равна 500 пикселей, 10 процентов — 100 пикселей, и так далее.

Отрисовка

Наконец, на шаге отрисовки, видимый контент страницы может быть преобразован в пиксели, чтобы появиться на экране.

Время, которое займет этот этап, зависит как от величины DOM, так и от того, какие стили применяются. Некоторые стили требуют больше усилий, чтобы быть применёнными, чем другие. Например, сложное градиентное фоновое изображение потребует больше времени, чем простой сплошной цвет на фоне.

HTML

The HyperText Markup Language (HTML) — язык разметки для создания структуры веб-страницы и представления контента. Благодаря разметке браузер знает в каком порядке отображать элементы, и что они значат.

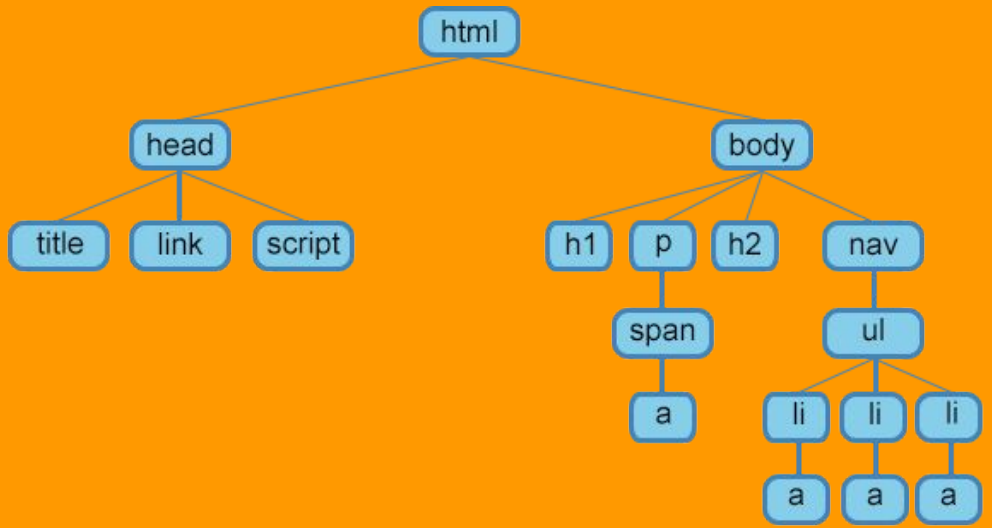
Структура HTML-документа

HTML-документ — это обычный текстовый документ, может быть создан как в обычном текстовом редакторе (Блокнот), так и в специализированном, с подсветкой кода (Notepad++, Visual Studio Code и т.п.). HTML-документ имеет расширение .html.

```
<!DOCTYPE html> <!-- Объявление формата документа -->
<html>
  <head><!-- Техническая информация о документе -->
    <meta charset="UTF-8"> <!-- Определяем кодировку символов документа -->
    <title>...</title> <!-- Задаем заголовок документа -->
    <link rel="stylesheet" type="text/css" href="style.css"> <!-- Подключаем внешнюю таблицу стилей -->
    <script src="script.js"></script> <!-- Подключаем сценарии -->
  </head>

  <body>
    <!-- Основная часть документа -->
  </body>
</html>
```

Структура HTML-документа

Элементы, находящиеся внутри элемента <html>, образуют дерево документа, так называемую объектную модель документа, DOM (document object model). При этом элемент <html> является корневым элементом. Родственные отношения между элементами (предок, потомок, родительский элемент, дочерний элемент, сестринский элемент).



Структура HTML-документа

- Элемент <html> Является корневым элементом документа. Все остальные элементы содержатся внутри <html>...</html>.
- Раздел <head>...</head> содержит техническую информацию о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т.д. Введенная в нем информация не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.
  - <title>
  - <meta>
  - <style>
  - <link>
  - <script>
- В разделе <body> располагается все содержимое документа.

HTML-элементы

HTML-элементы — основа языка HTML. Каждый HTML-документ состоит из дерева HTML-элементов и текста. Каждый HTML-элемент обозначается начальным (открывающим) и конечным (закрывающим) тегом. Открывающий и закрывающий теги содержат имя элемента.
Все HTML-элементы делятся на пять типов:

- пустые элементы (не парные) — <area>, <base>,
, <col>, <embed>, <hr>, <img>, <input>, <link>, <menuitem>, <meta>, <param>, <source>, <track>, <wbr>;
- элементы с неформатированным текстом — <script>, <style>;
- элементы, выводящие неформатированный текст — <textarea>, <title>;
- элементы из другого пространства имён — MathML и SVG; обычные элементы — все остальные элементы (, <h1>, <p> и т. д.).

HTML-атрибуты

HTML-атрибуты это специальные слова, которые управляют поведением HTML-элемента. Они добавляют дополнительную функциональность, либо меняют поведение элемента по умолчанию. Атрибуты элемента выражаются внутри начального тега элемента.
Имена и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.
Некоторые атрибуты не требуют значение, потому что у них есть только одна опция. Они называются логическими атрибутами.
Атрибуты могут быть указаны четырьмя различными способами:

- имя атрибута, например, disabled
- значение атрибута без кавычек, например, autocapitalize=sentences
- значение атрибута в одинарных кавычках, например, type='checkbox'
- значение атрибута в двойных кавычках, например, class="external icon-link"

Поток документа

Поток — одно из важнейших базовых понятий в вёрстке. Это принцип организации элементов на странице при отсутствии стилей: если мы напишем HTML и не напишем CSS, то отображение в браузере будет предсказуемо благодаря тому, что мы абсолютно точно знаем, как браузер располагает элементы в потоке.
Даже если к странице не подключено никаких стилей, к каждому элементу всё равно будут применяться CSS-правила, «зашитые» в движке браузера. Благодаря этим правилам заголовок <h1> крупнее заголовка <h2>, а ссылки — синие и подчёркнутые. На основании этих правил условно все элементы на странице можно разделить на две категории: блочные (block) и строчные (inline). Например, будет блочным, а или <a> — строчным. Поменять стандартное поведение можно при помощи CSS-свойства display. Если вообще не применять никаких стилей, браузер формирует из элементов нормальный поток. Поведение блочных элементов в нормальном потоке отличается от поведения строчных.

HTML-текст

HTML-текст представлен в спецификации элементами для форматирования и группировки текста. Данные элементы являются контейнерами для текста и не имеют визуального отображения.
Элементы для форматирования текста несут смысловую нагрузку и обычно задают для текста, заключенного внутрь, стилевое оформление, например, выделяют текст жирным начертанием или отображают его шрифтом другого семейства (свойство font-family).

HTML-элементы для текста

- Заголовки: <h1...h6>
- Форматирование текста: <b>, , <i>, <small>, , <sub>, <sup>, <ins>, <del>, <mark>
- Ввод «компьютерного» текста: <code>, <kbd>, <samp>, <var>, <pre>
- Оформление цитат и определений: <abbr>, <bdo>, <blockquote>, <q>, <cite>, <dfn>
- Абзацы, средства переноса текста: <p>,
, <hr>

HTML-ссылки

HTML-ссылки создаются с помощью элементов <a>, <area> и <link>. Ссылки представляют собой связь между двумя ресурсами, одним из которых является текущий документ.
Ссылки можно поделить на две категории:

- ссылки на внешние ресурсы — создаются с помощью элемента <link> и используются для расширения возможностей текущего документа при обработке браузером;
- гиперссылки — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Как сделать гиперссылки на сайте

- Структура ссылки
- Абсолютный и относительный путь
- Якорь
- Как сделать изображение-ссылку
- Атрибуты ссылок (href, download, target)

Структура ссылки

```
<!-- метод доступа://имя сервера:порт/путь -->
<a href="http://site.ru">указатель ссылки</a>

<!-- file обеспечивает чтение файла с локального диска: -->
<a href="file:/gallery/pictures/summer.html">указатель ссылки</a>

<!-- http предоставляет доступ к веб-странице по протоколу HTTP: -->
<a href="http://site.ru/">указатель ссылки</a>

<!-- https – специальная реализация протокола HTTP,
использующая шифрование (как правило, SSL или TLS): -->
<a href="https://site.ru/">указатель ссылки</a>

<!-- ftp осуществляет запрос к FTP-серверу на получение файла: -->
<a href="ftp://pgu/directory/library">указатель ссылки</a>

<!-- mailto запускает сеанс почтовой связи с указанным адресатом и хостом: -->
<a href="mailto:nika@gmail.com">указатель ссылки</a>
```

Абсолютный путь

Абсолютный путь указывает точное местоположение файла в пределах всей структуры папок на компьютере (сервере). Абсолютный путь к файлу даёт доступ к файлу со сторонних ресурсов и содержит следующие компоненты:

- протокол, например, http (опционально);
- домен (доменное имя или IP-адрес компьютера);
- папка (имя папки, указывающей путь к файлу);
- файл (имя файла).

http://site.ru/pages/tips/tips1.html
site.ru/pages/tips/tips1.html

http://site.ru/index.html
http://site.ru/

Относительный путь

Относительный путь описывает путь к указанному документу относительно текущего. Путь определяется с учётом местоположения веб-страницы, на которой находится ссылка. Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте. Когда браузер не находит в ссылке протокол http://, он выполняет поиск указанного документа на том же сервере.

Относительный путь содержит следующие компоненты:

- папка (имя папки, указывающей путь к файлу);
- файл (имя файла).

Путь для относительных ссылок имеет три специальных обозначения:

- **/** указывает на корневую директорию и говорит о том, что нужно начать путь от корневого каталога документов и идти вниз до следующей папки
- **./** указывает на текущую папку
- **../** подняться на одну папку (директорию) выше

Элемент группирует или оборачивает другие элементы и семантически ничего не значит. Сам по себе без стилей ничего из себя не представляет — пользователь увидит пустое место на экране.

Можно представить этот тег как универсальную коробку. В неё можно положить что угодно или не класть ничего и просто оформить как нужно.

Div — от английского division — раздел, секция.

Чтобы сделать что-нибудь полезное, надо обратиться к такому диву и добавить каких-нибудь стилей, обычно с помощью атрибута class, например: class="my-block" в HTML и .my-block в CSS. Можно задать ширину и отцентрировать всё содержимое. Или можно задать ему какие-нибудь наследуемые стили, вроде color: tomato или font-size: 20px, и тогда эти стили применятся ко всему содержимому этого дива.

Якоря и ссылки на изображения

```
<h1>Времена года</h1>
<h2>Оглавление</h2>
<a href="#p1">Лето</a> <!--создаём якорь, указав #id элемента-->
<a href="#p2">Осень</a>
<a href="#p3">Зима</a>
<a href="#p4">Весна</a>
<p id="p1">...</p> <!--добавляем соответствующий id элементу-->
<p id="p2">...</p>
<p id="p3">...</p>
<p id="p4">...</p>
```

```
<a href="http://www.fast-torrent.ru/film/gran-za-granyu-tv.html" target="_blank">
  <img src="//html5book.ru/wp-content/uploads/2014/07/Fringe.jpg" alt="Fringe">
</a>
```

HTML-изображения

HTML-изображения добавляются на веб-страницы с помощью элемента <img>. Использование графики делает веб-страницы визуально привлекательнее. Изображения помогают лучше передать суть и содержание веб-документа.

```
<img
   width="400"
   height="300"
   src="flower.jpg"
   alt="Роза в моём саду"
   srcset="
     img/flower-mobile.jpg 320w,
     img/flower-wide-mobile.jpg 480w,
     img/flower-tablet.jpg 768w,
     img/flower-desktop.jpg 1024w,
     img/flower-hires.jpg 1280w"
   sizes="
     (max-width: 20em) 30vw,
     (max-width: 30em) 60vw,
     (max-width: 40em) 90vw"
 >
```

HTML-списки

HTML-списки используются для группировки связанных между собой фрагментов информации. Существует три вида списков:

- маркированный список — — каждый элемент списка <li> отмечается маркером,
- нумерованный список — — каждый элемент списка <li> отмечается цифрой,
- список определений — <dl> — состоит из пар термин <dt> — <dd> определение.

Каждый список представляет собой контейнер, внутри которого располагаются элементы списка или пары термин-определение. Элементы списка ведут себя как блочные элементы, располагаясь друг под другом и занимая всю ширину блока-контейнера. Каждый элемент списка имеет дополнительный блок, расположенный сбоку, который не участвует в компоновке.

HTML-списки

```
<ol>
   <li>пункт</li>
   <li>пункт
      <ol>
         <li>пункт</li>
         <li>пункт</li>
         <li>пункт
            <ol>
               <li>пункт</li>
               <li>пункт</li>
               <li>пункт</li>
            </ol>
         </li>
         <li>пункт</li>
      </ol>
   </li>
   <li>пункт</li>
   <li>пункт</li>
</ol>
```

```
<ul>
 <li>Пункт 1.</li>
  <li>Пункт 2.
    <ul>
      <li>Подпункт 2.1.</li>
       <li>Подпункт 2.2.
        <ul>
          <li>Подпункт 2.2.1.</li>
          <li>Подпункт 2.2.2.</li>
          </ul>
       </li>
      <li>Подпункт 2.3.</li>
    </ul>
  </li>
 <li>Пункт 3.</li>
</ul>
```

HTML-таблицы

HTML-таблицы упорядочивают и выводят на экран данные с помощью строк или столбцов.
Таблицы состоят из ячеек, образующихся при пересечении строк и столбцов. Ячейки таблиц могут содержать любые HTML-элементы, такие как заголовки, списки, текст, изображения, элементы форм, а также другие таблицы. Каждой таблице можно добавить связанный с ней заголовок, расположив его перед таблицей или после неё.
Таблицы больше не используются для вёрстки веб-страниц и компоновки отдельных элементов, потому что такой приём не обеспечивает гибкость структуры и адаптивность сайта, существенно увеличивая HTML-разметку.
Для всех элементов таблицы доступны глобальные атрибуты, а также собственные атрибуты.

Пример разметки таблицы

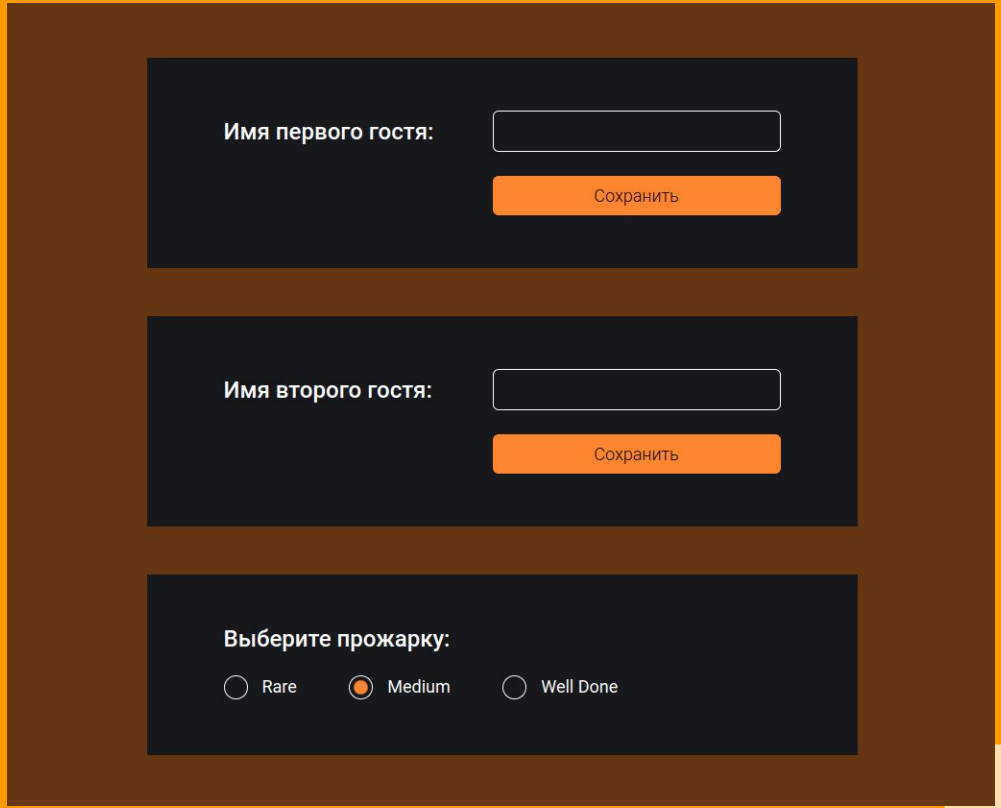
```
<table>
  <caption>Перечень продуктов</caption>

  <thead>
    <tr>
      <th>№ п/п</th>
      <th>Наименование товара</th>
      <th>Ед. изм.</th>
      <th>Количество</th>
      <th>Цена за ед. изм., руб.</th>
      <th>Стоимость, руб.</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td colspan="5" style="text-align:right">ИТОГО:</td><td>1168,80</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>1.</td>
      <td>Томаты свежие</td>
      <td>кг</td>
      <td>15,20</td>
      <td>69,00</td>
      <td>1048,80</td>
    </tr>
    <tr>
      <td>2.</td>
      <td>Огурцы свежие</td>
      <td>кг</td>
      <td>2,50</td>
      <td>48,00</td>
      <td>120,00</td>
    </tr>
  </tbody>
</table>
```

HTML5-формы

HTML-формы являются элементами управления, которые применяются для сбора информации от посетителей веб-сайта.
Веб-формы состоят из набора текстовых полей, кнопок, списков и других элементов управления, которые активизируются щелчком мыши. Технически формы передают данные от пользователя удаленному серверу.
Для получения и обработки данных форм используются языки веб-программирования, такие как PHP, Perl.
До появления HTML5 веб-формы представляли собой набор нескольких элементов <input type="text">, <input type="password">, завершающихся кнопкой <input type="submit">. Для стилизации форм в разных браузерах приходилось прилагать немало усилий. Кроме того, формы требовали применения JavaScript для проверки введенных данных, а также были лишены специфических типов полей ввода для указания повседневной информации типа дат, адресов электронной почты и URL-адресов. HTML5-формы решили большинство этих распространенных проблем.

Примеры разметки форм

```html
<!-- Эта форма отправит значение методом GET — мы получим URL с ответом -->
  <form action="" method="get">
    <label>
      Имя первого гостя:
      <input type="text" name="name">
    </label>
    <button type="submit">Сохранить</button>
  </form>

<!-- Эта форма отправит данные методом POST -->
<form action="" method="post">
  <label for="post-name">
    Имя второго гостя:
    <input id="post-name" type="text" name="name">
  </label>
  <button type="submit">Сохранить</button>
</form>

<!-- Форма с радиокнопками -->
<form action="" method="post">
  <fieldset>
    <legend>Выберите прожарку</legend>
    <label>
      <input type="radio" name="level">
      Rare
    </label>
    <label>
      <input type="radio" name="level"checked>
      Medium
    </label>
    <label>
      <input type="radio" name="level">
      Well Done
    </label>
  </fieldset>
</form>
```



Семантика и доступность

Давным-давно почти все делали сайты и не переживали о том, что у них внутри. Верстали таблицами, использовали всё, что попадётся под руку (а попадались в основном и) и не особо заморачивались о доступности. А потом случился HTML5 и понеслось.

Семантическая вёрстка — подход к разметке, который опирается не на содержание сайта, а на смысловое предназначение каждого блока и логическую структуру документа. Даже в этой статье есть заголовки разных уровней — это помогает читателю выстроить в голове структуру документа. Так и на странице сайта — только читатели будут немного другими.

Чтобы сделать сайт доступным. Зрячие пользователи могут без проблем с первого взгляда понять, где какая часть страницы находится — где заголовок, списки или изображения. Для незрячих или частично незрячих всё сложнее. Основной инструмент для просмотра сайтов не браузер, который отрисовывает страницу, а скринридер, который читает текст со страницы вслух.

Генераторы HTML

- Генерация на сервере (PHP, Perl)
- HTML шаблонизаторы
- Конструкторы сайтов

CSS

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML).
Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.
Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Структура CSS правила

Объявление стиля состоит из двух частей: селектора и объявления. В HTML имена элементов нечувствительны к регистру, поэтому «h1» работает так же, как и «H1». Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

Виды таблиц стилей

```
<!-- Внешняя таблица стилей -->
<head>
  <link rel="stylesheet" href="css/style.css">
</head>

<!-- Внутренние стили -->
<head>
  <style>
    h1,
    h2 {
      color: red;
      font-family: "Times New Roman", Georgia, Serif;
      line-height: 1.3em;
    }
  </style>
</head>
<body>
...
</body>

<!-- Встроенные стили -->
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>

<!-- Правило @import -->
<style>
  @import url(mobile.css);
  @import url(https://fonts.googleapis.com/css?
family=Open+Sans&subset=latin,cyrillic);
  p {
    font-size: 0.9em;
    color: grey;
  }
</style>
```

Виды селекторов

```css
/*  Универсальный селектор */
* {
  margin: 0;
}

/* Селектор элемента */
h1 {
  font-family: Lobster, cursive;
}

/* Селектор класса */
/* <h1 class="headline">Инструкция пользования персональным компьютером</h1> */
.headline {
  text-transform: uppercase;
  color: lightblue;
}

/* Селектор идентификатора */
/* <div id="sidebar"></div> */
#sidebar {
  width: 300px;
  float: left;
}

/* Селектор потомка */
ul li {
  text-transform: uppercase;
}
p.first a {
  color: green;
}

/* Дочерний селектор */
p > strong {
    color: tomato;
}

/* Сестринский селектор */
h1 + p {
  margin: 0;
}
h1 ~ p {
  margin: 0;
}

/* Селектор атрибута */
img[title="flower"] {
  width: 100%;
}
```

```css
/* Селектор псевдокласса */
a:hover {
  color: gold;
}
input:checked {
  outline: 2px solid blue;
}

/* Селектор структурных псевдоклассов */
li:nth-child(even) {
  border-bottom: 1px solid black;
}
p:nth-child(3n+2) {
  padding-left: 20px;
}
p:last-child {
  margin-bottom: 0;
}

/* Селектор структурных псевдоклассов типа */
p:first-of-type {
  padding: 0;
}

/* Селектор псевдоэлемента */
.paragraph:first-letter {
  font-size: 32px;
}
.card:before {
  content: '';
  position: absolute;
  top: 0;
  right: 0;
  transform: translate(-50%, -50%);
  border-radius: 50%;
  background-color: red;
}
```

Комбинация и группировка селекторов

```css
/* Комбинация селекторов */
a[href][title] {
  color: blue;
}
img[alt*="css"]:nth-of-type(even)
{ width: 100%;
}

/* Группировка селекторов */
h1,
h2,
p,
span {
  color: tomato;
  background: white;
}
```

Наследование

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space и word-spacing. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются. Это background, border, display, float и clear, height и width, margin, min-max-height и -width, outline, overflow, padding, position, text-decoration, vertical-align и z-index.

Принудительное наследование

С помощью ключевого слова inherit можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

Как задаются и работают CSS-стили

Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения inherit).

Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.

Каскад

Каскадирование — это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила. Существует три критерия, которые определяют порядок применения свойств — **правило !important, специфичность и порядок, в котором подключены таблицы стилей**.

Правило !important

Вес правила можно задать с помощью ключевого слова !important, которое добавляется сразу после значения свойства, например, span {font-weight: bold!important;}. Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Специфичность

Для каждого правила браузер вычисляет специфичность селектора, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: 0, 0, 0, 0. Специфичность селектора определяется следующим образом:

- для id добавляется 0, 1, 0, 0;
- для class добавляется 0, 0, 1, 0;
- для каждого элемента и псевдоэлемента добавляется 0, 0, 0, 1;
- для встроенного стиля,
  добавленного непосредственно к элементу —1, 0, 0, 0;
- универсальный селектор не имеет специфичности.

```
h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
em {color: silver;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1,
2*/
.sidebar {color: grey;} /*специфичность 0, 0, 1, 0*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar  {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0, 1*/
```

Порядок подключённых таблиц

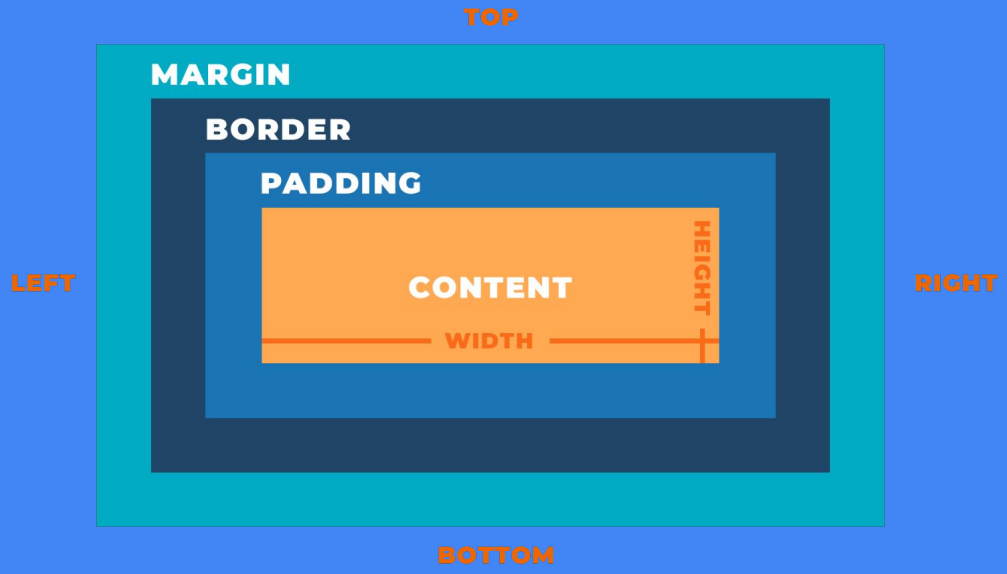
Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице. Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

CSS блочная модель

Модуль CSS Box Model описывает свойства padding и margin, которые создают поля внутри и отступы снаружи CSS блока. Размеры блока также могут быть увеличены за счет рамки.
Каждый блок имеет прямоугольную область содержимого в центре, поля вокруг содержимого, рамку вокруг полей и отступ за пределами рамки. Размеры этих областей определяют свойства padding и его подсвойства — padding-left, padding-topи т.д., border и его подсвойства, margin и его подсвойства.

Определение блочной модели

Каждый блок имеет область содержимого, в которой находится текст, дочерние элементы, изображение и т.п., и необязательные окружающие ее padding, border и margin. Размер каждой области определяется соответствующими свойствами и может быть нулевым, или, в случае margin, отрицательным.



Блочные и строчные элементы

Выделяют две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы — блочные и строчные элементы. С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов <area> и <img>).

Разделение элементов на блочные и строчные используется в спецификации HTML до версии 4.01. В HTML5 эти понятия заменены более сложным набором категорий контента, согласно которым каждый HTML-элемент должен следовать правилам, определяющим, какой контент для него допустим.

Модель визуального форматирования

HTML-документ организован в виде дерева элементов и текстовых узлов. Модель визуального форматирования CSS представляет собой алгоритм, который обрабатывает HTML-документ и выводит его на экран устройства.
Каждый блок в дереве представляет соответствующий элемент или псевдоэлемент, а текст (буквы, цифры, пробелы), находящийся между открывающим и закрывающим тегами, представляет содержимое текстовых узлов.
Чтобы создать дерево блоков, CSS сначала использует каскадирование и наследование, позволяющие назначить вычисленное значение для каждого css-свойства каждому элементу и текстовому узлу в исходном дереве.
Затем для каждого элемента CSS генерирует ноль или более блоков в соответствии со значением свойства display этого элемента. Как правило, элемент генерирует один основной блок, который представляет самого себя и содержит свое содержимое. Некоторые значение свойства display, например, display: list-item;, генерируют блок основного блока и блок дочернего маркера. Другие, например, display: none;, приводят к тому, что элемент и/или его потомки вообще не генерируют блоки.
Положение блоков на странице определяется следующими факторами:

- размером элемента (с учётом того, заданы они явно или нет);
- типом элемента (строчный или блочный);
- схемой позиционирования (нормальный поток, позиционированные или плавающие элементы);
- отношениями между элементами в DOM (родительский — дочерний элемент);
- внутренними размерами содержащихся изображений;
- внешней информацией (например, размеры окна браузера).

CSS-позиционирование

Нормальный поток
Нормальный поток включает блочный контекст форматирования (элементы с display block, list-item или table), строчный (встроенный) контекст форматирования (элементы с display inline, inline-block или inline-table), и относительное и «липкое» позиционирование элементов уровня блока и строки.

Обтекание
В обтекающей модели блок удаляется из нормального потока и позиционируется влево или вправо. Содержимое обтекает правую сторону элемента с float: left и левую сторону элемента с float: right.

Абсолютное позиционирование
В модели абсолютного позиционирования блок полностью удаляется из нормального потока и ему присваивается позиция относительно содержащего блока.
Абсолютное позиционирование реализуется с помощью значений position: absolute; и position: fixed;.Элементом «вне потока» может быть плавающий, абсолютно позиционированный или корневой элемент.

Базовые свойства CSS-шрифтов

- Семейство шрифтов: свойство font-family
- Насыщенность шрифта: свойство font-weight
- Ширина шрифта: свойство font-stretch
- Начертание шрифта: свойство font-style
- Размер шрифта: свойство font-size

CSS-текст

- Преобразование текста: свойство text-transform
- Обработка пробелов и переносы строк: свойство white-space
- Настройка табуляции: свойство tab-size
- Разрыв строки и границы слов
  - Правила разрыва для букв: свойство word-break
  - Разрыв строки: свойство line-break
  - Расстановка переносов: свойство hyphens
  - Переполнение блока-обертки: свойство overflow-wrap/word-wrap
- Выравнивание строк
  - Выравнивание текста: свойство text-align
  - Выравнивание текста по умолчанию: свойство text-align-all
  - Выравнивание последней строки: свойство text-align-last
- Промежутки
  - Промежутки между словами: свойство word-spacing
  - Трекинг: свойство letter-spacing
- Отступ первой строки: свойство text-indent

CSS-фон

- Базовый цвет: свойство background-color
- Источник изображения: свойство background-image
- Укладка изображений: свойство background-repeat
- Фиксация изображения: свойство background-attachment
- Позиционирование изображений: свойство background-position
- Область рисования: свойство background-clip
- Область расположения фона: свойство background-origin
- Размер изображений: свойство background-size
- Краткая запись свойств фона: свойство background
- Множественные фоны

Кастомные свойства в CSS

CSS-функция var() позволяет подставлять кастомные свойства в качестве значения свойств.

:root { --color-cyan: #00ffff; }

Функция var() возвращает текущее значение кастомного свойства. Если оно поменяется, то функция var() сразу вернёт актуальное значение.

Псевдоклассы

Псевдоклассы — особый вид селектора, который уточняет тип или состояние. Обычно это какой-то качественный признак: реакция на наведение курсора, порядок следования и другие.

Рассмотрим CSS для подсветки строки таблицы при наведении курсора.
В обычном состоянии цвет фона — тёмный:
tr { background-color: #18191C; }

Скопировать При наведении курсора цвет фона изменится на голубой:
tr:hover { background-color: #123E66; }

Благодаря псевдоклассам мы можем контролировать динамические параметры селекторов. Эти свойства сработают, когда селектор подходит под дополнительный признак.

Псевдоэлементы

Псевдоэлементы — это элементы, которых не существует в HTML-разметке. Они создаются и позиционируются исключительно при помощи CSS. Чаще всего используются для создания различных декоративных элементов (которые не несут содержательного смысла).
Также псевдоэлементы приходят на помощь, когда нужно наложить поверх картинки так называемый оверлей (перекрывающий слой). На этом польза от псевдоэлементов не заканчивается.

Самый частый сценарий использования псевдоэлемента — наложение оверлея, полупрозрачной заливки поверх картинки. Чаще всего это требуется на первом экране, но этим способом можно наложить оверлей на любое изображение на сайте.

Псевдоэлементы

Определение приставки «псевдо» в словаре: «...Приставка, означающая ложность, ненастоящесть следующего за ней».
Из этого определения можно понять, что псевдоэлементы не существуют на самом деле. Это порождение CSS, которое можно изменять, удалять, добавлять, не трогая реальную HTML-разметку.

Как пишется
Существует несколько псевдоэлементов.

::before и ::after
Два самых часто встречающихся псевдоэлемента. Они очень похожи. Единственная разница заключается в том, что ::before по умолчанию располагается перед содержимым элемента, для которого задаётся, а ::after — после. Эта разница отражена в названии: слово before переводится с английского как «до» или «перед»; слово after переводится как «после».
Для обоих элементов обязательным является свойство content — содержимое псевдоэлемента. С его помощью можно, например, вставить какое-нибудь слово до или после текста. Без свойства content псевдоэлемент не отобразится.

Псевдоэлементы

```css
.header {
  position: relative;
  z-index: 0;
  background: #999999
    url("background.svg")
    no-repeat center / cover;
}

.header::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0 0 0 /
0.7);index: -1;
}
```

Единицы измерения длины или расстояния

- **Относительные величины**
  Используются для задания размера или расстояния относительно чего-либо. Например, vh считается относительно высоты вьюпорта (области просмотра страницы в окне браузера).
- **Абсолютные величины**
  Абсолютные величины ни от чего не зависят и привязаны к физическим единицам измерения: дюймам или сантиметрам.
- **Единицы измерения углов**
  Используются, как правило, для задания угла поворота элемента, направления линейного градиента или угла поворота конического градиента.
- **Единицы измерения времени**
  Используются для задания длительности анимации или перехода в свойствах animation и transition.
- **Проценты**
  Используются, чтобы указать, что значение представляет собой долю от другой величины. Исходное значение, от которого берётся часть, может относиться как к самому элементу, так и к его предку. Всё зависит от того, для какого свойства мы применяем проценты.

Flexbox

Долгое время веб-интерфейсы были статичными — сайты разрабатывались и просматривались только на экранах мониторов стационарных компьютеров. Однако с десяток лет назад, совсем недавно по историческим меркам, у нас появилось огромное разнообразие экранов — от мобильных телефонов до телевизоров, — на которых мы можем взаимодействовать с сайтами. Так родилась необходимость в гибких системах раскладки.

Идея флексбоксов появилась ещё в 2009 году, и этот стандарт до сих пор развивается и прорабатывается. Основная идея флексов — гибкое распределение места между элементами, гибкая расстановка, выравнивание, гибкое управление. Ключевое слово — гибкое, что и отражено в названии (flex — англ. гибко).

CSS Grid Layout

CSS Grid Layout (спецификация) или просто гриды — это удобная технология для раскладки элементов на веб-страницах. В отличие от флексбоксов, одновременно работающих только с одним измерением, гриды дают возможность работать одновременно с двумя: горизонталью и вертикалью.

На заре веба для создания раскладки поинтереснее одноколоночной часто использовались таблицы. Но проблема была в том, что таблицы были сделаны не для этого. Даже если отбросить семантическую нагруженность этого элемента, отдельные свойства для стилизации были крайне неудобными. По сути, это был костыль за неимением более удобных способов. Гриды же изначально были придуманы и реализованы как инструмент для создания раскладки.

Принцип работы гридов чем-то похож на таблицы. Вместо работы только с рядами или только с колонками с помощью гридов можно работать с так называемыми грид-ячейками, позиционируя элементы по вертикали и горизонтали одновременно.

БЭМ (Блок, Элемент, Модификатор)

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste».

Блок
Функционально независимый компонент страницы, который может быть повторно использован. В HTML блоки представлены атрибутом class.

Элемент
Составная часть блока, которая не может использоваться в отрыве от него.

Модификатор
Сущность, определяющая внешний вид, состояние или поведение блока либо элемента.

@media

Директива, которая позволяет задавать разные стили для разных параметров экрана — ширины, высоты, ориентации и даже типа устройства.

При вёрстке адаптивных сайтов часто нужно, чтобы какой-то набор стилей применился только при соблюдении определённых условий. Например, текст должен стать зелёным только при горизонтальной ориентации смартфона. Или блок займёт всю ширину родителя, если ширина экрана больше или равна 1500 пикселям. Для подобных случаев и существуют медиавыражения. Они помогают адаптировать вёрстку под разные экраны и устройства и при этом не писать лишний код.

@media (min-width: 900px) { .block { display: flex; } }

CSS препроцессор

CSS препроцессор (CSS preprocessor) - это программа, которая имеет свой собственный синтаксис (syntax (en-US)), но может сгенерировать из него CSS код . Существует множество препроцессоров. Большинство из них расширяет возможности чистого CSS, добавляя такие опции как: примеси, вложенные правила, селекторы наследования и др. Эти особенности облегчают работу с CSS: упрощают чтение кода и его дальнейшую поддержку.
Для использования CSS препроцессора нужно установить CSS компилятор на ваш веб-сервер

- SASS
- LESS
- Stylus
- PostCSS

Кроссбраузерность

Кросс-браузерность — свойство веб-сайта отображаться и функционировать во всех часто используемых браузерах идентично. Под идентичностью функционирования подразумевается: отсутствие некорректной работы, ошибок в вёрстке и способность отображать материал с одинаковой степенью читабельности. Вследствие постоянного развития веб-технологий, приемлемую кросс-браузерность возможно обеспечить только для последних версий браузеров.

Термин начали использовать во время браузерных войн, происходивших в конце 1990-х годов. В этом контексте термин относится к сайтам, которые работали как в Internet Explorer, так и в Netscape Navigator. В то время производители стали внедрять собственные функции для браузеров, что привело к некоторым особенностям отображения и концептуальным различиям в работе.

Часто разработчики сайтов устанавливают «страницу-заглушку» для пользователей неподдерживаемых браузеров, на которой предлагается сменить браузер.