С. Г. ТОЛСТЫХ, А. И. ПОПОВ

ОСНОВЫ WEB-ДИЗАЙНА

Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2025

Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тамбовский государственный технический университет»**

С. Г. ТОЛСТЫХ, А. И. ПОПОВ

ОСНОВЫ WEB-ДИЗАЙНА

Утверждено Ученым советом университета
в качестве учебного пособия для студентов 3 и 4 курсов специальностей
09.02.07 «Информационные системы и программирование»
и 09.02.11 «Разработка и управление программным обеспечением»

Учебное электронное издание

Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2025

УДК 004.774.6
ББК 16.263.434
Т52

Рецензенты:

Кандидат технических наук, доцент,
и.о. директора Института автоматики и информационных технологий
ФГБОУ ВО «ТГТУ»
Д. В. Поляков

Начальник отдела мультимедийных систем
и видеоконференцсвязи ТОГБУ «РИТЦ»
Д. Л. Гриднев

Толстых, С. Г.

Т52 Основы Web-дизайна [Электронный ресурс] : учебное пособие / С. Г. Толстых, А. И. Попов. – Тамбов : Издательский центр ФГБОУ ВО «ТГТУ», 2025. – 1 электрон. опт. диск (CD-ROM). – Системные требования : ПК не ниже класса Pentium IV ; RAM 512 Mb ; необходимое место на HDD 2,52 Mb ; Windows 7/8/10/11 ; дисковод CD-ROM ; мышь. – Загл. с экрана.

ISBN 978-5-8265-2963-8

Рассмотрены основные понятия языка HTML, его элементы и атрибуты, описаны подходы к созданию HTML-форм, описаны основы CSS, приведены примеры оформления WEB-страниц с использованием языка стилей CSS.

Предназначено для студентов 3 и 4 курсов специальностей 09.02.07 «Информационные системы и программирование» и 09.02.11 «Разработка и управление программным обеспечением».

УДК 004.774.6
ББК 16.263.434

ISBN 978-5-8265-2963-8

ВВЕДЕНИЕ

Цифровая трансформация экономики и ее направленность на обеспечение условий для технологического прорыва предполагают, что в хозяйственной деятельности ключевым фактором производства являются данные в цифровом виде, при этом активно используются обработка больших объемов информации и результаты их анализа. Это позволяет существенно повысить прибыльность различных видов производства, внедрять передовые технологии, совершенствовать оборудование, эффективнее продавать, доставлять товары и оказывать услуги. Качество управленческой работы и быстрота принятия решений во многом определяются возможностью представления цифровой информации в максимально наглядном виде.

Для повышения эффективности цифровых операций в производственной сфере необходимо спроектировать удобные пользовательские веб-интерфейсы, используемые для сайтов и веб-приложений. Специалист в области веб-разработки должен спроектировать логическую структуру веб-страниц, определить наиболее удобные способы подачи информации, сделать художественное оформление веб-проекта. Для выполнения данных требований разработчику необходимо проявить не только владение цифровыми технологиями, но и показать свое художественное мышление, творческие качества, умение акцентировать внимание пользователя на наиболее важных элементах информации. При разработке веб-страниц желательно рационально сочетать эстетичность ее восприятия пользователем и оптимальный функционал, позволяющий решить запланированные задачи, и удобство при использовании приложения. При этом необходимо выбирать визуальные компоненты с учетом требований заказчика, возрастных и культурных особенностей, специфики решаемых задач.

Работа веб-дизайнера предполагает постоянное взаимодействие с клиентом: от совместного составления технического задания до создания динамичного прототипа и его корректировки по результатам обратной связи с заказчиком, а также при отладке сайта и анализе его функционирования для внесения необходимых корректировок и улучшения в дальнейшем.

Для уверенной конкурентоспособности на рынке труда начинающий веб-дизайнер должен знать основные понятия языка HTML, его элементы и атрибуты, изучить подходы к созданию HTML-форм, ознакомиться с основами CSS. Для осознанного освоения данного сегмента профессиональной готовности программиста в пособии приведен ряд примеров оформления веб-страниц с использованием языка стилей CSS.

1. ОСНОВЫ ЯЗЫКА-HTML

HTML (HyperText Markup Language) – это язык разметки для создания веб-страниц, который представляет простые правила оформления и компактный набор структурных и семантических элементов разметки (тегов). HTML позволяет описывать способ представления логических частей документа (заголовки, абзацы, списки и т.д.) и создавать веб-страницы разной сложности. HTML не является языком программирования, но веб-страницы могут содержать встроенные или загружаемые программы на скриптовых языках (в первую очередь Javascript) и программы-апплеты на языке Java.

HTML состоит из тегов, которые используются для определения различных элементов на странице. Теги начинаются с символа « < » и заканчиваются символом « > ». Некоторые теги имеют атрибуты, которые используются для определения дополнительных свойств элемента.

Все HTML документы должны начинаться с объявления типа документа: <!DOCTYPE html>. Сам HTML документ начинается с <html> и заканчивается </html>. Видимая часть HTML документа находится между <body> и </body>.

Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Элементы заголовок и параграф</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Основной документ</h1>

  <h2>Заголовок</h2>
  <p>Параграф</p>

 </body>
</html>
```

HTML5 Основной документ

Заголовок

Параграф

1.1. ЭЛЕМЕНТЫ И АТРИБУТЫ HTML

Элементы HTML – это теги, которые используются для различных элементов на странице. Каждый элемент имеет свой уникальный тег и может содержать другие элементы внутри себя.

Основные элементы HTML:

Элемент	Описание
<html>	Начало HTML-документа
<head>	Содержит информацию о документе, такую как заголовок, ключевые слова и метаданные
<body>	Основное содержимое документа
	Блок контента, который может быть стилизован с помощью CSS
<p>	Абзац текста
<img>	Изображение на странице
<a>	Гиперссылка на другую страницу или файл
	Неупорядоченный список
	Упорядоченный список
<table>	Таблица на странице

Атрибуты HTML – это дополнительные свойства элементов, которые могут быть определены в теге.

Атрибут	Описание
id	Используется для уникальной идентификации элемента на странице
class	Используется для определения класса элемента, который может быть стилизован с помощью CSS
href	Используется в теге <a> для определения ссылки на другую страницу или файл
src	Используется в теге <img> для определения пути к изображению
style	Используется для определения стилей CSS для элемента
title	Используется для определения всплывающей подсказки при наведении на элемент

Атрибуты добавляются к элементам HTML для определения дополнительных свойств элемента.

2. ОСНОВНЫЕ ЭЛЕМЕНТЫ И АТРИБУТЫ ЯЗЫКА HTML

2.1. HTML-ЗАГОЛОВКИ

Для создания заголовков используется 6 парных тегов: <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, где <h1> – заголовок первого уровня, самый важный и описывающий главную тему текста, а <h6> – заголовок самого низшего уровня.

Каждый заголовок имеет свой уникальный тег, содержит внутри себя текст и другие элементы. Заголовки используются для организации контента на странице и помогают поисковым системам понять структуру документа.

Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Заголовки</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Заголовки</h1>

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

 </body>
</html>
```

HTML5 Заголовки

Заголовок 1

Заголовок 2

Заголовок 3

Заголовок 4

Заголовок 5

Заголовок 6

2.2. HTML-ФОРМАТИРОВАНИЕ

Ряд элементов HTML предназначены для форматирования текстового содержимого, например, для выделения жирным или курсивом и т.д.

Элемент	Описание
<b>	Выделяет текст жирным
<del>	Зачеркивает текст
<i>	Выделяет текст курсивом
	Выделяет текст курсивом, в отличие от тега <i> носит логическое значение, придает выделяемому тексту оттенок важности
<s>	Зачеркивает текст
<small>	Делает текст чуть меньше размером, чем окружающий
	Выделяет текст жирным, в отличие от тега <b> предназначен для логического выделения, чтобы показать важность текста.
<sub>	Помещает текст под строкой
<sup>	Помещает текст над строкой
<u>	Подчеркивает текст
<ins>	Определяет вставленный (или добавленный) текст
<mark>	Выделяет текст цветом, придавая ему оттенок важности

Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Форматирование текста</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Форматирование текста</h1>

  <h2>Форматирование текста</h2>

  <p><b>Жирный текст</b></p>
  <p><i>Курсив</i></p>
  <p>Текст <sub>подстрочный</sub> и <sup>надстрочный</sup></p>

 </body>
</html>
```

HTML5 Форматирование текста

Форматирование текста

Жирный текст

Курсив

Текст $_{\text{подстрочный}}$ и $^{\text{надстрочный}}$

2.3. HTML-ЦВЕТА

В HTML-цвета могут быть заданы с использованием названий цветов (например, red, blue, green), шестнадцатеричных значений (например, #FF0000 для красного цвета), RGB значений (например, rgb(255, 0, 0) для красного цвета).

Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Цвет фона</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Цвет фон</h1>

  <h2 style="background-color:DodgerBlue;">Текст 1</h2>

  <p style="background-color:Tomato;">
   Текст 2
  </p>

 </body>
</html>
```

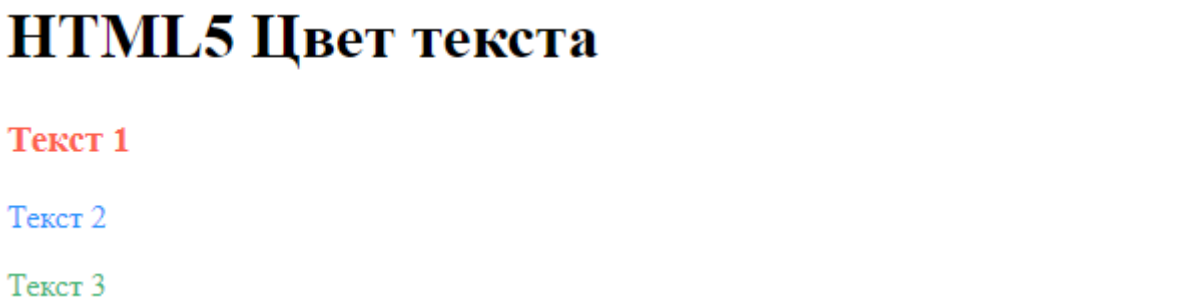
HTML5 Цвет фон

Текст 1

Текст 2

Пример HTML-кода для задания цвета текста.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Цвет текста</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Цвет текста</h1>

  <h3 style="color:Tomato;">Текст 1</h3>

  <p style="color:DodgerBlue;">Текст 2</p>

  <p style="color:MediumSeaGreen;">Текст 3</p>

 </body>
</html>
```



Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Цвет границы</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Цвет границы</h1>

  <h2 style="border: 2px solid Tomato;">Текст 1</h2>

  <h2 style="border: 2px solid DodgerBlue;">Текст 2</h2>

  <h2 style="border: 2px solid Violet;">Текст 3</h2>

 </body>
</html>
```

HTML5 Цвет границы

Текст 1

Текст 2

Текст 3

2.4. HTML-ИЗОБРАЖЕНИЯ

Для добавления изображения на веб-страницу используется тег <img>, который имеет два обязательных атрибута src и alt. Атрибут src необходим для указания адреса графического файла, а атрибут alt должен содержать текстовое описание изображения, которое появится при недоступности изображения.

Пример.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Изображение </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 Изображение</h1>

  <p>Используйте атрибут style для указания ширины и высоты изображения:</p>
  <img src="4.jpg" alt="Кошка" style="width:300px;height:300px;">

 </body>
</html>
```

HTML5 Изображение

Используйте атрибут style для указания ширины и высоты изображения:

2.5. HTML-ССЫЛКИ

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Оба действия выполняются с помощью тега <a>. Общий синтаксис создания ссылок следующий:

```
<a href="URL">текст ссылки</a>
```

Элемент <a> имеет следующие атрибуты:

Атрибут	Описание
href	Адрес ссылки
hreflang	Указывает на язык документа, на который ведет данная ссылка
media	Определяет устройство, для которого предназначена ссылка
rel	Отношение между данным документом и ресурсом, на который ведет ссылка
target	Как документ по ссылке должен открываться

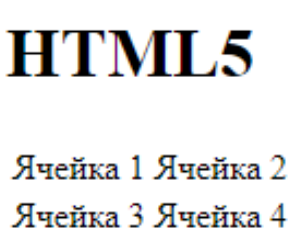
2.6. HTML-ТАБЛИЦЫ

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений. Для добавления таблицы на веб-страницу используется тег <table>. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов <tr> и <td>.

Пример.

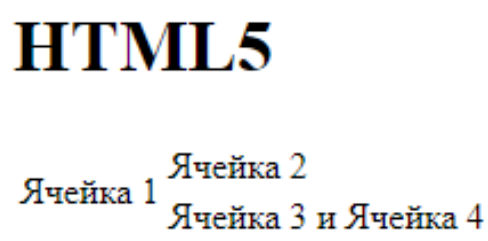
```
<!DOCTYPE html>
<html>
 <head>
    <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>
```

```
  <h1>HTML5 </h1>

  <table>
  <tr>
    <td>Ячейка 1</td>
    <td>Ячейка 2</td>
  </tr>
  <tr>
    <td>Ячейка 3</td>
    <td>Ячейка 4</td>
  </tr>
</table>
 </body>
</html>
```



Для того чтобы объединить ячейки в строках или столбцах, необходимо к тегу td добавить атрибуты rowspan и colspan. В приведенном ниже примере показано создание таблицы с объединенными ячейками:

```
 <table>
  <tr>
    <td rowspan="2">Ячейка 1</td>
    <td>Ячейка 2</td>
  </tr>
  <tr>
    <td colspan="2">Ячейка 3 и Ячейка 4</td>
  </tr>
</table>
```



Для того чтобы добавить границу таблицы, используется атрибут border у тега <table>. Пример создания таблицы с границей толщиной 1 пиксель:

```
<table border="1">
 <tr>
   <td>Ячейка 1</td>
   <td>Ячейка 2</td>
 </tr>
 <tr>
   <td>Ячейка 3</td>
   <td>Ячейка 4</td>
 </tr>
</table>
```

HTML5

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Для настройки внешнего вида можно использовать стили CSS-таблицы, Например, в нижеприведенном коде устанавливается стиль таблицы, в котором границы ячеек будут черными и толщиной 1 пиксель, а расстояние между границами ячеек и их содержимым будет равно 5 пикселям.

```
<!DOCTYPE html>
<html>
 <head>
  <title> – schoolsw3.com</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
  table {
    border-collapse: collapse;
  }
  td {
    border: 1px solid black;
    padding: 5px;
  }
</style>
 <head>
 <body>

  <h1>HTML5 </h1>

  <table style="width:100%">
   <tr>
    <td>Ячейка 1</td>
    <td>Ячейка 2</td>
```

```
    </tr>
    <tr>
     <td>Ячейка 3</td>
     <td>Ячейка 4</td>

    </tr>
   </table>

 </body>
</html>
```

HTML5

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

2.7. HTML-СПИСКИ

HTML поддерживает несколько типов списков.

1. Неупорядоченный список (unordered list) представляет собой список, элементы которого не имеют порядкового номера и обозначаются маркерами (обычно кружками или точками). Для создания списка применяется тег <ul>, а для добавления элементов в список используется тег <li>.

Пример.

```
  <h1>HTML5 Список неупорядоченный</h1>

<ul>
  <li>Первый элемент</li>
  <li>Второй элемент</li>
  <li>Третий элемент</li>
</ul>
```

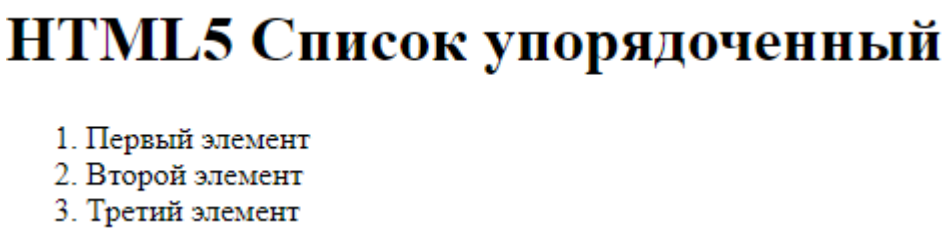
HTML5 Список неупорядоченный

- Первый элемент
- Второй элемент
- Третий элемент

2. Упорядоченный список (ordered list) представляет собой набор элементов с их порядковыми номерами. Вид и тип нумерации зависит от атрибутов тега <ol>, который и применяется для создания списка. Каждый пункт нумерованного списка обозначается тегом <li>, как показано ниже.

Пример:

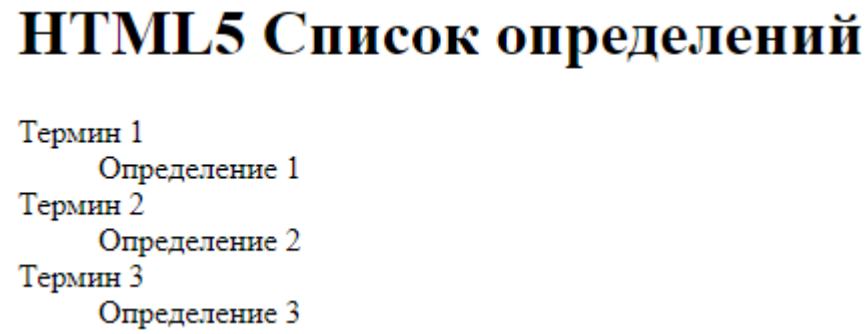
```
  <h1>HTML5 Список упорядоченный</h1>

 <ol>
  <li>Первый элемент</li>
  <li>Второй элемент</li>
  <li>Третий элемент</li>
</ol>
```



3. Список определений состоит из двух элементов – термина и его определения. Сам список задается с помощью контейнера <dl>, термин – тегом <dt>, а его определение – с помощью тега <dd>. Вложение тегов для создания списка определений продемонстрировано в примере ниже.

Пример:

```
  <h1>HTML5 Список определений</h1>

 <dl>
  <dt>Термин 1</dt>
  <dd>Определение 1</dd>
  <dt>Термин 2</dt>
  <dd>Определение 2</dd>
  <dt>Термин 3</dt>
  <dd>Определение 3</dd>
</dl>
```



2.8. HTML-СТИЛИ

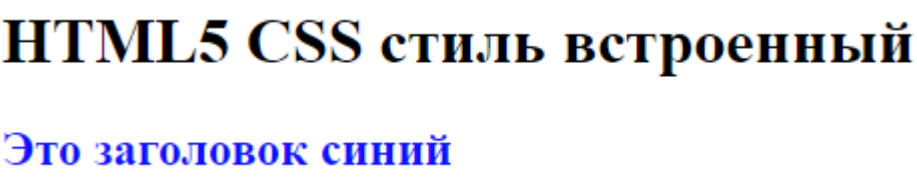
Тег <style> применяется для определения стилей элементов веб-страницы. Тег <style> необходимо использовать внутри контейнера <head>. Можно задавать более чем один тег <style>.

В таблице приведены основные свойства атрибута <style>:

Свойство	Описание
color	Определяет цвет текста
background-color	Определяет цвет фона элемента
font-size	Размер шрифта
font-family	Семейство шрифтов
text-align	Выравнивание текста
border	Границу элемента
padding	Отступы внутри элемента
margin	Отступы вокруг элемента

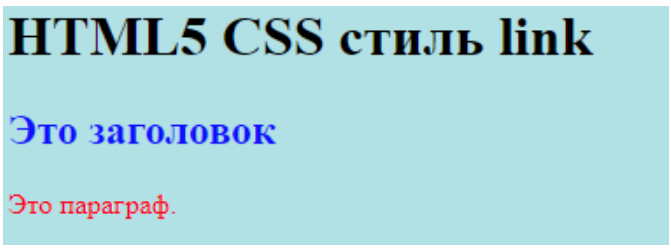
Пример (встроенный CSS).

```
<!DOCTYPE html>
<html>
 <head>
  <title>CSS стиль встроенный</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
 <body>

  <h1>HTML5 CSS стиль встроенный</h1>

  <h2 style="color:blue;">Это заголовок синий</h2>

 </body>
</html>
```



Пример (внешний CSS).

```
<!DOCTYPE html>
<html>
 <head>
  <title>CSS стиль </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="styles.css">
 </head>
 <body>

  <h1>HTML5 CSS стиль link</h1>

  <h2>Это заголовок</h2>
  <p>Это параграф.</p>

 </body>
</html>
```



Пример (внешний файл "styles.css").

```
body {
    background-color: powderblue;
}
h1 {
    color: blue;
}
p {
    color: red;
}
```

2.9. HTML-КЛАССЫ

Атрибут class представляет собой универсальный атрибут тега, с помощью которого можно задать имя любому элементу на странице. Имя элемента в дальнейшем используется в качестве селектора в CSS и позволяет управлять стилями элемента. В приведенном ниже примере показано, как одному элементу можно присвоить один или несколько классов.

Пример.

```
<!-- Один класс -->
<div class="container">
  <!-- ... -->
</div>

<!-- Несколько классов -->
<div class="container special-box">
  <!-- ... -->
</div>
```

Чтобы атрибут class сработал, нужно указать имя класса и добавить его в CSS-файл или в теге <style>.

Сначала подключаем класс:

```
<span class="highlighted-text">Этот текст будет выделен.</span>
```

А затем настраиваем класс в CSS:

```
.highlighted-text {
  background-color: yellow;
  font-size: 18px;
}
```

2.10. ДОБАВЛЕНИЕ КОДА JAVASCRIPT В HTML

JavaScript (JS) представляет собой высокоуровневый, интерпретируемый язык программирования, который используется для создания динамических и интерактивных элементов на веб-сайтах. JavaScript можно использовать для создания сложных веб-сайтов, браузерных игр и приложений, а также для подключения серверов к веб-сайтам и веб-приложениям.

С JavaScript разработчики могут: отображать своевременные и актуальные обновления контента; вставлять раскрывающиеся меню; воспроизводить аудио и видео; увеличивать и уменьшать изображения; добавлять 2D/3D графику и анимацию; вставлять эффекты, которые приходят в действие при наведении курсора; отображать и скрывать меню; предупреждать пользователей о недопустимых символах при вводе паролей; создавать сложные браузерные игры.

Для добавления JavaScript в HTML можно воспользоваться двумя способами. Можно непосредственно встроить JavaScript в теги HTML или же создать внешний JavaScript-файл.

Пример, демонстрирующий добавление JavaScript в раздел <head> HTML-страницы:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
document.getElementById(“demo”).innerHTML = “Paragraph changed.”;
}
</script>
</head>
<body>
<h2>Demo JavaScript in Head</h2>
<p id=”demo”>A Paragraph</p>
<button type=”button” onclick=”myFunction()”>Try it</button>
</body>
</html>
```

А теперь добавим JavaScript в тег <head>:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
document.getElementById(“demo”).innerHTML = “Paragraph changed.”;
}
</script>
</head>
<body>
<h2>Demo JavaScript in Head</h2>
<p id=”demo”>A Paragraph</p>
<button type=”button” onclick=”myFunction()”>Try it</button>
</body>
</html>
```

Создадим файл myScript.JavaScript:

```
function myFunction() {
document.getElementById(“demo”).innerHTML = “Paragraph changed.”;
}
```

Для вызова созданного файла добавим имя файла скрипта в атрибут src (source) тега <script>:

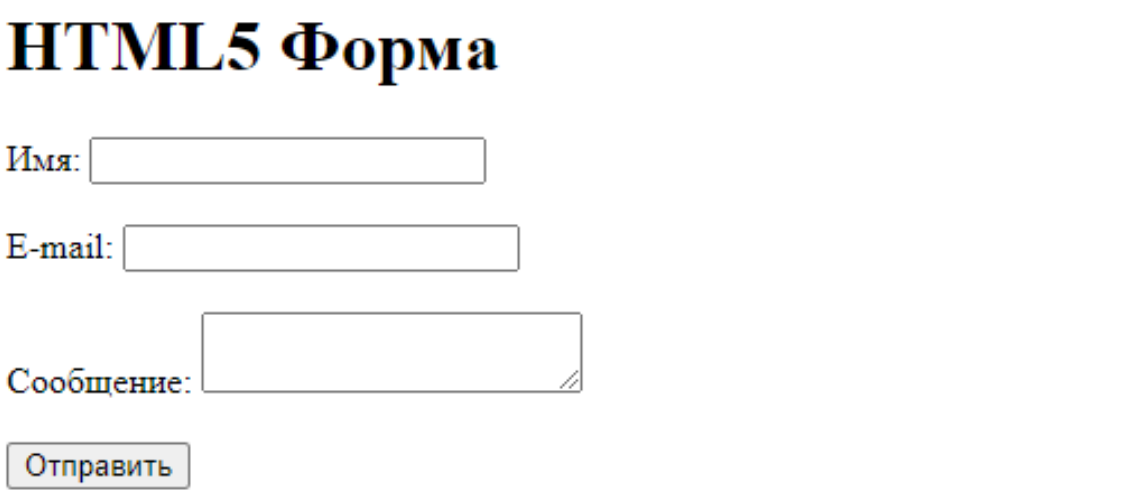
```
<script src=”myScript.JavaScript”></script>
```

3. СОЗДАНИЕ HTML-ФОРМ

HTM-формы позволяют пользователям вводить и отправлять данные на сервер. Форма состоит из одного или нескольких элементов управления, таких как текстовые поля, кнопки, флажки и т.д. Для создания формы используется тег <form>.

Пример:

```
  <h1>HTML5 Форма</h1>

 <form action="http://example.com/submit" method="post">
  <label for="name">Имя:</label>
  <input type="text" id="name" name="name"><br>
  <br>

  <label for="email">E-mail:</label>
  <input type="email" id="email" name="email"><br>
  <br>

  <label for="message">Сообщение:</label>
  <textarea id="message" name="message"></textarea><br>
  <br>

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



Вышеуказанный код создает форму с тремя полями ввода (имя, e-mail и сообщение) и кнопкой отправки. Атрибут action указывает адрес, на который будут отправлены данные формы, а метод method указывает тип запроса (GET или POST).

3.1. Элементы HTML-форм

Сейчас практически ни один сайт не обходится без элементов интерфейса вроде полей ввода текста, кнопок, переключателей и флажков. Они необходимы для взаимодействия с пользователем, для поиска на сайтах по ключевым словам, написания комментариев, прохождения опросов, добавления фотографий и т.д. Именно формы и обеспечивают получение данных от пользователя и передачу их на сервер, где они уже подвергаются анализу и обработке. Рассмотрим основные элементы форм.

1. Текстовое поле

```
<input type="text" name="имя" value="значение">
```

2. Поле для ввода пароля

```
<input type="password" name="имя" value="значение">
```

3. Флажок

```
<input type="checkbox" name="имя" value="значение">
```

4. Выпадающий список

```
<select name="имя">
  <option value="значение1">Вариант 1</option>
  <option value="значение2">Вариант 2</option>
  <option value="значение3">Вариант 3</option>
</select>
```

5. Радиокнопки

```
<input type="radio" name="имя" value="значение1"> Вариант 1
<input type="radio" name="имя" value="значение2"> Вариант 2
<input type="radio" name="имя" value="значение3"> Вариант 3
```

6. Кнопка отправки формы:

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

7. Поле для ввода адреса электронной почты:

```
<input type="email" name="имя" value="значение">
```

8. Поле для ввода номера телефона:

```
<input type="tel" name="имя" value="значение">
```

9. Поле для ввода даты:

```
<input type="date" name="имя" value="значение">
```

10. Поле для ввода времени:

```
<input type="time" name="имя" value="значение">
```

11. Поле для ввода числа:

```
<input type="number" name="имя" value="значение">
```

12. Поле для ввода URL-адреса:

```
<input type="url" name="имя" value="значение">
```

3.2. КНОПКИ

Кнопку на веб-странице можно создать с использованием тега <button>. Данный тег имеет следующие атрибуты.

Атрибут	Описание
name	Имя кнопки предназначено для того, чтобы обработчик формы мог его идентифицировать
value	Значение кнопки и одновременно надпись на ней

Рассмотрим основные виды кнопок.

1. Обычная кнопка:

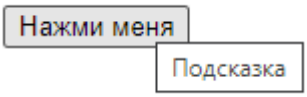
```
<input type="button" value="Нажми меня">
```

2. Кнопка с изображением:

```
<input type="image" src=" button.png " alt="Кнопка">
```

3. Кнопка с подсказкой:

```
<input type="button" value="Нажми меня" title="Подсказка">
```



4. Кнопка для очистки формы:

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

5. Кнопка для загрузки файла:

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

6. Кнопка для перехода по ссылке:

```
<input type="button" onclick="location.href='http://www.example.com';"
value="Перейти">
```

7. Кнопка для вызова JavaScript функции:

```
<input type="button" onclick="myFunction()" value="Выполнить функцию">
```

3.3. ТЕКСТОВЫЕ ПОЛЯ

Для ввода текстовой информации на веб-форме предусмотрены различные виды текстовых полей.

1. Однострочное текстовое поле:

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

2. Многострочное текстовое поле:

```
<textarea name="имя" rows="4" cols="50"></textarea>
```

3. Текстовое поле с предустановленным значением:

```
<input type="text" name="имя" value="Значение">
```

4. Текстовое поле с ограничением на количество символов:

```
<input type="text" name="имя" maxlength="10">
```

5. Текстовое поле с маской ввода:

```
<input type="text" name="имя" placeholder="Маска ввода">
```

6. Текстовое поле, доступное только для чтения:

```
<input type="text" name="имя" value="Значение" readonly>
```

7. Текстовое поле, скрытое от пользователя:

```
<input type="hidden" name="имя" value="Значение">
```

3.4. ФЛАЖКИ И ПЕРЕКЛЮЧАТЕЛИ

Флажки (чек-боксы) используют, когда необходимо выбрать любое количество вариантов из предложенного списка. Если же требуется выбор лишь одного варианта, то для этого используют переключатели (радиокнопки). Рассмотрим примеры создания флажков и переключателей.

1. Флажок:

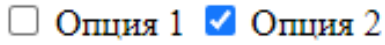
```
<input type="checkbox" name="имя" value="Значение">
```

2. Флажок с предустановленным значением:

```
<input type="checkbox" name="имя" value="Значение" checked>
```

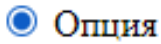
3. Группа флажков:

```
<label><input type="checkbox" name="имя1" value="Значение1"> Опция 1</label>
<label><input type="checkbox" name="имя2" value="Значение2"> Опция 2</label>
```



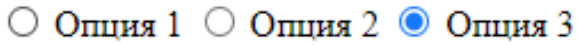
4. Переключатель:

```
<label><input type="radio" name="имя" value="Значение"> Опция </label>
```



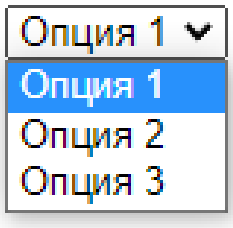
5. Группа переключателей:

```
<label><input type="radio" name="имя" value="Значение1"> Опция 1</label>
<label><input type="radio" name="имя" value="Значение2"> Опция 2</label>
<label><input type="radio" name="имя" value="Значение3"> Опция 3</label>
```



6. Элемент select создает список

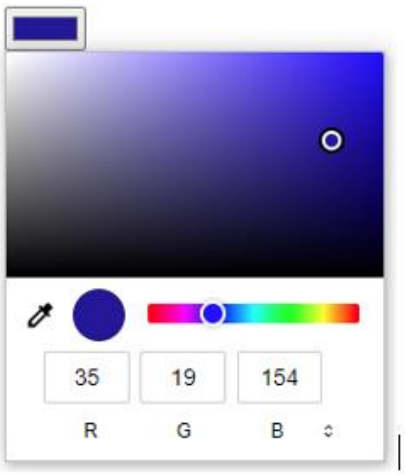
```
<select name="имя">
  <option value="Значение1">Опция 1</option>
  <option value="Значение2">Опция 2</option>
  <option value="Значение3">Опция 3</option>
</select>
```



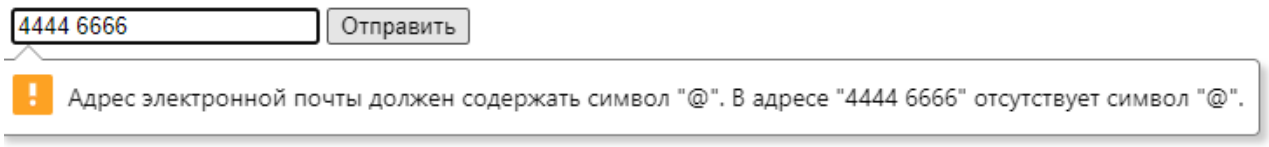
3.5. ЭЛЕМЕНТЫ ФОРМ ДЛЯ ВВОДА ЦВЕТА, url, email, ТЕЛЕФОНА

С помощью тега input с атрибутом type в значении color можно создать поле для цвета.

```
<input type="color" name="цвет">
```



Поля для ввода url, email, телефона используют для проверки ввода соответствующий шаблон. Если будет введено некорректное значение, то при отправке формы браузер отобразит сообщение о некорректном вводе, а форма не будет отправлена.



4. ОСНОВЫ CSS

Стили являются удобным, практичным и эффективным инструментом при верстке веб-страниц и оформлении текста, ссылок, изображений и других элементов. Несмотря на явные плюсы применения стилей, рассмотрим все преимущества CSS, в том числе, и не заметные на первый взгляд.

Разграничение кода и оформления

Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров, стара как мир. В идеале веб-страница должна содержать только теги логического форматирования, а вид элементов задается через стили. При подобном разделении работа над дизайном и верстской сайта может вестись параллельно.

Разное оформление для разных устройств

С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др. Например, на экране монитора отображать страницу в одном оформлении, а при ее печати – в другом. Эта возможность также позволяет скрывать или показывать некоторые элементы документа при отображении на разных устройствах.

Расширенные по сравнению с HTML способы оформления элементов

В отличие от HTML, стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.

Ускорение загрузки сайта

При хранении стилей в отдельном файле он кэшируется и при повторном обращении к нему извлекается из кэша браузера. За счет кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов.

Кэшем называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы при первом обращении к сайту. При следующем обращении к сайту эти файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

Единое стилевое оформление множества документов

Сайт – это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.

Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того, чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем, и оформление нужных документов сразу же поменяется.

CSS позволяет создавать разнообразные дизайны веб-страниц и существенно упрощает обслуживание и изменение элементов дизайна.

CSS можно добавлять в веб-страницы двумя способами: внутри HTML-документа и с помощью внешнего файла CSS.

CSS внутри HTML-документа – это набор стилей, определяемых внутри тега <style> в секции <head> веб-страницы. Это позволяет добавлять стили непосредственно к элементам веб-страницы, но неудобно при работе с несколькими страницами, поскольку потребуется дублирование кода на каждой из них.

4.1. CSS-СИНТАКСИС

Таблицы стилей состоят из набора правил. Каждое правило состоит из одного или нескольких селекторов. Селектор определяет, к какому элементу или элементам HTML применяются стили. Свойство – это характеристика элемента (например, цвет, размер шрифта и т.д.). Значение – это конкретное определение свойства (например, "red" для цвета или "16px" для размера шрифта).

Основная структура

```
…
селектор {
   свойство: значение;
}
```

Пример.

```
  <style>

   h1 {
    color: blue;
    font-size: 24px;
    text-align: center;
}
  </style>
```

В данном случае все заголовки первого уровня (тег <h1>) будут отображаться синим цветом, иметь размер шрифта 24 пикселя и располагаться по центру. Для пояснения отдельных частей кода или временного исключения правил в CSS предусмотрены комментарии.

```
 <style>
/* Это комментарий */
 p {
   color: green; /* Цвет текста зеленый */
  }
 </style>
```

Комментарии разрешено размещать в любом месте CSS-документа, включая многострочные блоки. Вложенные комментарии, тем не менее, не поддерживаются.

4.2. СЕЛЕКТОРЫ CSS

Селектором может быть любой тег HTML, значения, которые мы описываем, например, цвет текста, фона, размеры и т.д. В начале селектора пишется название и через фигурные скобки ставится свойство, после двоеточия пишется его значение. Рассмотрим наиболее популярные варианты.

Селектор	Описание

Теговые	Определяют все элементы одного типа:
	p {
	margin: 20px;
	}

Классовые	Применяют к элементам всех тегов с определенным классом: .my-class { background-color: yellow;}

Идентификаторные	Применяют к элементу одного тега с определенным id:
	my-id {
	font-weight: bold;
	}

Комбинированные	Можно применить не только одного типа,
	а последовательно несколько:
	div.my-class {
	border: 1px solid black;
	}

4.3. CSS-ПСЕВДОЭЛЕМЕНТЫ

Псевдоэлементы предоставляют возможность стилизовать отдельные части элементов, например ::before или ::after:

```
p::first-line {
font-weight: bold;
}
```

Для наглядной и удобной организации кода рекомендуется распределять стили по категориям:

```
/* Основные стили */
body {
font-family: Arial, sans-serif;
}
/* Стили заголовков */
h1, h2, h3 {
margin: 0;
padding: 10px;
}
/* Стили параграфов */
p {
line-height: 1.5;
}
```

Существуют три способа подключения CSS-стилей к HTML-документам: использование внутренних стилей, внешних стилей и инлайн-стилей.

Подключение стилей	Описание
Использование внутренних стилей	Этот подход позволяет включить CSS непосредственно в HTML-документ внутри элемента `<style>` в секции `<head>`. Такой метод удобен для небольших проектов или временных стилей, применяемых только на одной странице

	<!DOCTYPE html>
	<html lang="ru">
	<head>
	<meta charset="UTF-8">
	<title>Пример внутреннего стиля</title>
	<style>
	body {
	background-color: lightblue;
	}
	h1 {
	color: darkblue;
	}
	</style>
	</head>
	<body>
	 Привет, мир!
	</body>
	</html>

Использование внешних стилей	Внешний стиль применяется для подключения отдельно созданного CSS-файла к HTML-документу. Это наиболее распространенный метод, поскольку он обеспечивает разделение содержания и оформления, а также позволяет использовать один CSS-файл для нескольких веб-страниц. Для подключения используется элемент <link>, который размещается внутри секции <head> и указывает на путь к CSS-файлу. <!DOCTYPE html> <html lang="ru"> <head> <meta charset="UTF-8"> <title>Пример внешнего стиля</title> <link rel="stylesheet" href="styles.css"> </head> <body> Привет, мир! </body> </html>

Использование инлайн-стилей	Инлайн-стили применяются напрямую к HTML-элементам через атрибут style. Этот подход лучше всего подходит для изменения стиля отдельных элементов в случаях, когда требуется оперативное внесение изменений. <!DOCTYPE html> <html lang="ru"> <head> <meta charset="UTF-8"> <title>Пример инлайн-стиля</title> </head> <body> Привет, мир! </body> </html>

Выбор способа подключения CSS зависит от специфики проекта и структуры веб-страницы. На практике часто комбинируют внешние и внутренние стили, дополняя их инлайн-стилями для решения конкретных задач.

4.4. CSS-ТЕКСТ

Форматирование текста с использованием CSS помогает создать стили, которые определяют визуальное отображение текста на веб-странице. Рассмотрим основные свойства, которые можно применить для стилизации текста.

Стиль	Описание
Шрифт	– font-family – устанавливает шрифт текста; – font-size – устанавливает размер текста; – font-weight – устанавливает жирность текста (обычный, жирный и т.д.); – font-style – устанавливает стиль шрифта (обычный, курсив, наклонный); – text-transform – преобразует текст (например, в верхний регистр). H1 {

	font-family: 'Arial', sans-serif; font-size: 36px; font-weight: bold; font-style: italic; text-transform: uppercase; /* Изменяет текст на верхний регистр */ }

Цвет и заливка	color – задание цвета текста
	background-color – установка цвета фона текста
	p {
	color: #333333; /* Dark gray text */
	background-color: #f0f0f0; /* Light gray background */
	}

Отступы и расстояния	– line-height – на шрифт задает его высоту, что со своей стороны влияет на удобочитаемость текста; – letter-spacing – задает расстояние между буквами; – word-spacing – задает расстояние между словами; – text-indent – задает отступ первой строки. P { line-height: 1.5; /* Height of line */ letter-spacing: 1px; /* Interletter space */ text-indent: 20px; /* First-line indentation */ }

Текстовое выравнивание	– text-align – значение, определяющее, как выравнивается текст (по левому краю, по основанию, по правому краю или по ширине);
	– text-decoration – устанавливает декорирование текста (подчеркивание, зачеркивание и т.д.).
	h2 {
	text-align: center; /*Выравнивание по центру */
	text-decoration: underline; /* Подчеркивание текста */
	}

Эффекты текста	– text-shadow – принимает тень на текст;
	– text-overflow – конфигурирует отображение текста; если он выходит за рамки контейнера;
	– white-space – определяет, как произвести обработку пробелов и переносов строки.
	H3 {
	text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /* Тень текста*/}
	p {
	text-overflow: ellipsis; /* Вывести три точки (…) при переполнении */
	white-space: nowrap; /* Отменить перенос строки */
	overflow: hidden; /* Скрыть переполненный текст */}

4.5. ОФОРМЛЕНИЕ ФОНА В CSS

Рассмотрим различные примеры оформления фона в CSS.

Стиль	Пример

Цвет фона	body {
	background-color: lightblue; /* Цвет по имени */
	}
	div {
	background-color: #f0f0f0; /* Шестнадцатеричный цвет */
	}
	header {
	background-color: rgb(255, 0, 0); /* RGB цвет */
	}
	footer {
	background-color: rgba(0, 0, 255, 0.5); /* RGBA цвет с прозрачностью */}

Фоновое изображение	div {
	background-image: url('image.jpg'); /* Путь к изображению */
	background-size: cover; /* Масштабирование изображения для заполнения элемента */
	}

Повторение фона	div {
	background-image: url('pattern.png');
	background-repeat: repeat; /* Изображение будет повторяться по обеим осям */
	}
	header {
	background-image: url('banner.jpg');
	background-repeat: no-repeat; /* Изображение не будет повторяться */
	}

Позиционирование фона	div {
	background-image: url('banner.jpg');
	background-position: center; /* По центру */
	}
	footer {

	background-image: url('footer-bg.png');
	background-position: bottom right; /* Внизу справа */
	}

Размер фона	div {
	background-image: url('image.jpg');
	background-size: cover; /* Изображение заполнит весь элемент */
	}
	header {
	background-image: url('image.jpg');
	background-size: contain; /* Изображение полностью вписывается в элемент */
	}

Множественные фоны	div {
	background-image: url('image1.jpg'), url('image2.png');
	background-position: left top, right bottom;
	background-size: 200px 100px, auto; /* Размер для каждого изображения */
	}

Градиенты	/* Линейный градиент */
	div {
	background: linear-gradient(to right, red, blue);
	}
	/* Радиальный градиент */
	div {
	background: radial-gradient(circle, red, yellow, green);
	}

4.6. БЛОЧНАЯ МОДЕЛЬ CSS

Блочная модель CSS (или блоковая модель) представляет собой концепцию, описывающую, как элементы на веб-странице взаимодействуют друг с другом и занимают пространство. Это базовый инструмент для веб-дизайнеров и разработчиков, поскольку понимание блочной модели напрямую влияет на структуру и оформление веб-страниц.

Основные компоненты блочной модели.

Блочная модель включает несколько ключевых элементов, определяющих пространство, занимаемое компонентами:

1. Содержимое (Content) – это сам элемент с его содержимым, например текстом, изображениями или другими вложенными элементами.

2. Внутренний отступ (Padding) – область между содержимым и границей элемента. Она увеличивает расстояние между содержимым и границей, а значения отступов можно задавать отдельно для каждой из сторон: сверху, справа, снизу и слева.

3. Граница (Border) – линия, окружающая элемент. Ее можно настроить по стилю, толщине и цвету, чтобы задать визуальное выделение.

4. Внешний отступ (Margin) – пространство между элементом и соседними элементами. Margin регулирует расстояние между объектами на странице и, аналогично padding, может быть настроен индивидуально для каждой стороны.

Для наглядности общая схема блочной модели представлена на рис. 1.

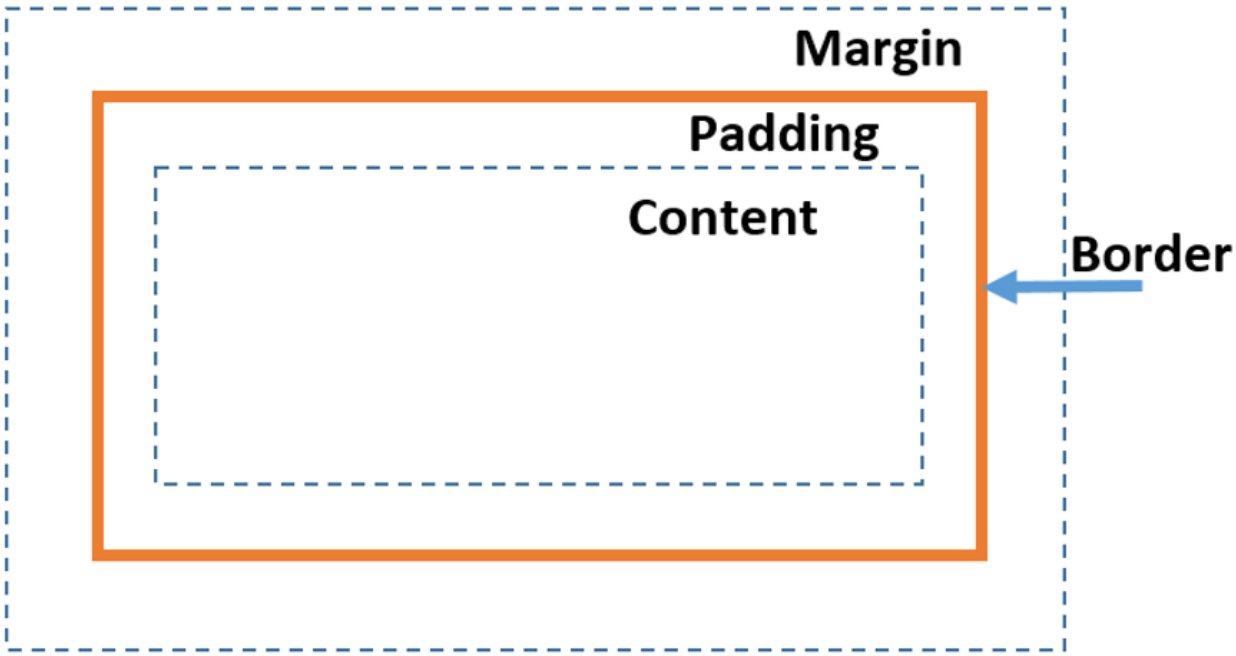
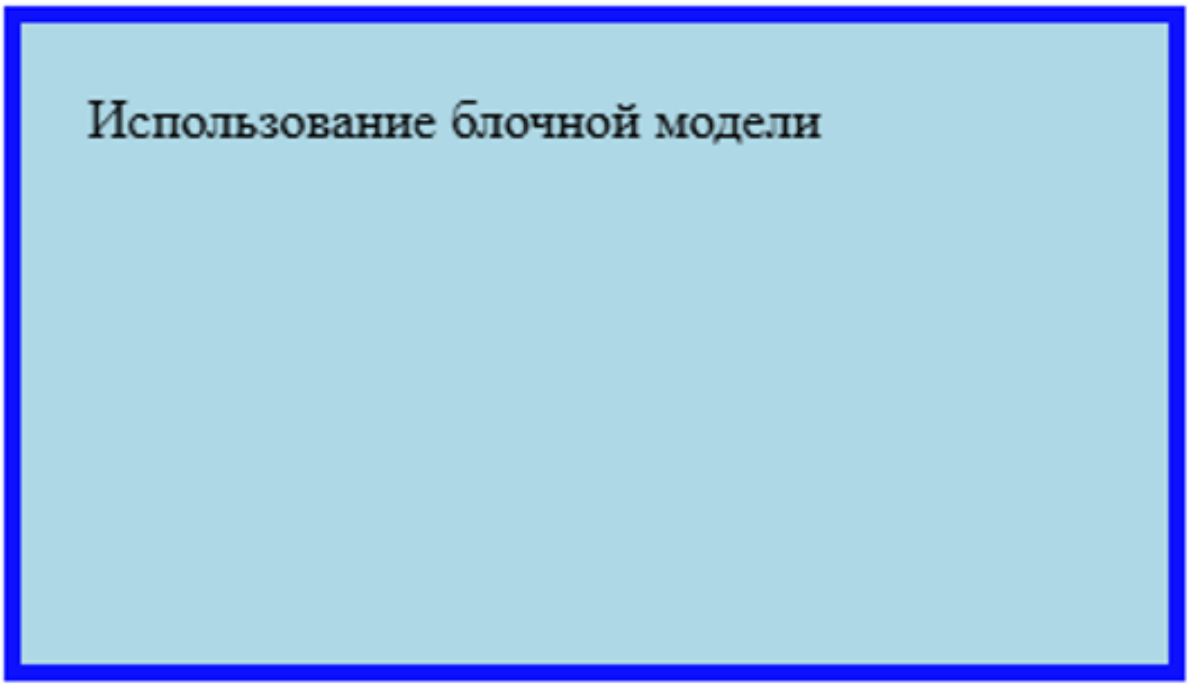


Рис. 1. Общая схема блочной модели

Пример.

```
  <style>
  .box {
   width: 300px; /* Ширина содержимого */
   height: 150px; /* Высота содержимого */
   background-color: lightblue; /* Цвет фона содержимого */
   padding: 20px; /* Внутренний отступ */
   border: 5px solid blue; /* Граница */
   margin: 30px; /* Внешний отступ */
   }
 </style>
</head>
<body>
 <div class="box"> Использование блочной модели </div>
</body>
```

Результат.



Размер блока в CSS обычно состоит из таких частей:

– ширина области с содержимым (content width);
– отступы между содержимым и рамкой слева и справа (padding);
– рамка слева и справа (border);
– отступы снаружи слева и справа (margin).

Можно просто сложить все части, и получить полную ширину блока:

Ширина блока = Ширина области с содержимым + Левый padding + + Правый padding + Левый border + Правый border + Левый margin + + Правый margin

К примеру, если у элемента установлена ширина width: 300px; то добавление padding и border сделает его фактическую ширину на странице больше. По умолчанию размеры блока учитывают только ширину и высоту области с содержимым.

Пример.

```
.box {
box-sizing: border-box; /* Размеры включают padding и border в расчет */
width: 300px;
height: 150px;
padding: 20px;
border: 5px solid blue;
}
```

4.7. СТИЛИЗАЦИЯ ГРАНИЦ В CSS

С помощью границ можно визуально выделить блок, создать рамку вокруг изображения, сделать намного выразительнее кнопки и добавить декоративные эффекты. Для создания границ используется следующий синтаксис: border: <ширина> <стиль> <цвет>.

Пример.

```
Div {
  border: 1px solid black;
}
```

Основные свойства таблицы.

Свойство	Описание

border-width	Устанавливает ширину границы
	div {
	border-width: 1px 2px 3px 4px; /* Вверх, Право, Вниз, Лево */
	}

border-style	Задает стиль границы
	div {
	border-style: dashed; /* Штриховая граница */
	}

border-color	Определяет цвет границы.
	Div {
	border-color: red; /* Красный цвет для всех сторон */
	border-color: red blue green yellow; /* Вверх, Право, Вниз, Лево */
	}

border-radius	Закругление границ
	div {
	border: 2px solid black;
	border-radius: 10px; /* Закругление углов на 10 пикселей */
	}

border-top	Устанавливает границу для верхней стороны
	div {
	border-top: 2px solid red; /* Верхняя граница */
	}

border-right	Устанавливает границу для правой стороны
	div {
	border-right: 3px dashed green; /* Правая граница */
	}

border-bottom	Устанавливает границу для нижней стороны
	div {
	border-bottom: 4px double blue; /* Нижняя граница */
	}

border-left	Устанавливает границу для левой стороны
	div {
	border-left: 1px dotted orange; /* Левая граница */
	}

Приведем пример, демонстрирующий различные стили границ.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Стилизация границ</title>
    <style>
        .solid {
            border: 2px solid black;
        }
        .dotted {
            border: 2px dotted red;
        }
        .dashed {
            border: 3px dashed green;
        }
        .double {
            border: 4px double blue;
        }
        .radius {
            border: 2px solid purple;
            border-radius: 15px;
        }
    </style>
</head>
<body>
    <div class="solid">Сплошная граница</div>
    <div class="dotted">Пунктирная граница</div>
    <div class="dashed">Штриховая граница</div>
    <div class="double">Двойная граница</div>
    <div class="radius">Закругленные углы</div>
</body>
</html>
```

Сплошная граница

Пунктирная граница

Штриховая граница

Двойная граница

Закругленные углы

4.8. ОТСТУПЫ В CSS

Отступы помогают управлять пространством вокруг элементов на странице. Они влияют на читаемость, композицию и общее восприятие интерфейса. В CSS есть два вида отступов: margin – внешний отступ, padding – внутренний отступ. Оба свойства работают с четырьмя сторонами элемента: сверху, справа, снизу и слева.

Пример базового использования:

```
.box {
    margin: 20px;      /* внешний отступ вокруг блока */
    padding: 15px;     /* внутренний отступ внутри блока */
}
```

Значения можно задавать по одной стороне:

```
.item {
    margin-top: 10px;
    margin-right: 5px;
    margin-bottom: 15px;
    margin-left: 5px;
}
```

Или более компактно в одном свойстве:

```
.item {
    margin: 10px 5px 15px 5px; /* сверху, справа, снизу, слева */
}
```

CSS поддерживает сокращенную запись, где стороны читаются по часовой стрелке, начиная сверху.

Запись	Расшифровка
margin: 20px;	все стороны по 20px
margin: 10px 20px;	сверху/снизу 10px, справа/слева 20px
margin: 10px 15px 20px;	сверху 10px, слева/справа 15px, снизу 20px
margin: 5px 10px 15px 20px;	все стороны отдельно

Иногда начинающие разработчики путают эти свойства, но они различаются между собой.

Таблица сравнения margin и padding.

	Margin	Padding
Расположение	Снаружи элемента	Внутри элемента
Влияет на фон	Нет	Да
Может быть отрицательным	Да	Нет

Пример, показывающий различия:

```
<div class="outer">
   <div class="inner">Текст</div>
</div>
.outer {
   background: lightgray;
}
.inner {
   background: lightblue;
   margin: 20px;
   padding: 20px;
}
```

Во внутреннем блоке будет видно, что padding увеличивает голубую область, а margin – расстояние от нее до границ серого блока.

Особенность margin: если два вертикальных внешних отступа касаются друг друга, они не складываются, а «схлопываются» – берется больший из двух.

```
.block1 {
   margin-bottom: 30px;
}
.block2 {
   margin-top: 20px;
}
```

Расстояние между block1 и block2 будет не 50px, а 30px. Это поведение иногда мешает при макетировании, но это можно исправить.

Чтобы отключить схлопывание, можно:

– добавить обертке padding или border
– задать overflow: hidden
– использовать display: flow-root

CSS Grid, как и Flexbox, поддерживает удобное свойство gap – оно задает расстояние между строками и колонками сетки.

```
.grid {
   display: grid;
   grid-template-columns: 1fr 1fr 1fr;
   gap: 15px;
}
```

В отличие от margin, использование gap не требует дополнительных вычислений и не ломает сетку даже при изменении количества элементов.

Можно задавать разное расстояние по горизонтали и вертикали:

```
.grid {
   display: grid;
   gap: 10px 30px; /* 10px – между строками, 30px – между колонками */
}
```

Как уже говорилось раньше, вертикальные margin у блочных элементов могут «схлопываться» (collapse). Это часто встречается между заголовком и блоком или между секциями:

```
<section class="a"></section>
<section class="b"></section>
.a { margin-bottom: 40px; }
.b { margin-top: 40px; }
```

В примере выше расстояние будет не 80px, а 40px. Существует три способа исправить эту ситуацию.

1) Добавить внутренний отступ родителю:

```
.parent {
   padding-top: 1px;
}
```

2) Использовать overflow:

```
.parent {
   overflow: hidden;
}
```

3) Использовать современное решение display:

```
.parent {
   display: flow-root;
}
```

Приемы центрирования с отступами.

Горизонтальное центрирование	Вертикальное центрирование

.block {
width: 300px;
margin: 0 auto;
}
	.container {
	display: flex;
	align-items: center;
	height: 300px;
	}

.wrap {
display: flex;
justify-content: center;
}
	.container {
	display: grid;
	place-items: center;
	}

Практические советы:

1) Старайтесь использовать gap вместо margin, когда работаете с flex или grid.

2) Для одинаковых отступов между секциями используйте модульную сетку (например: 20, 40, 60px).

3) Не злоупотребляйте отрицательными margin – они усложняют поддержку кода.

4) Всегда подключайте box-sizing: border-box.

5) Сбрасывайте стандартные отступы браузера в начале стилей.

Типичные ошибки разработчика.

Ошибка разработчика	Как исправить
Использует margin внутри flex-контейнера для расстояний между карточками	Использовать gap
У заголовков странные отступы сверху	Сбросить margin у <h1>–<h6>

Элементы «налипают» друг на друга	Использовать margin-bottom у секций
Center не работает через margin: auto	Проверить display и задана ли ширина
Макет «едет» при добавлении padding	Установить box-sizing: border-box

4.9. РАЗМЕЩЕНИЕ ЭЛЕМЕНТОВ CSS

Одной из самых главных задач разработчика при создании веб-страниц является работа с расположением элементов на странице. Правильное позиционирование влияет на внешний вид сайта, удобство восприятия и адаптивность дизайна. CSS предлагает несколько подходов к размещению: от простых свойств до более мощных современных инструментов. В HTML каждый элемент изначально имеет определенный тип отображения: блочный или строчный.

Тип	Особенности	Примеры
block	Начинается с новой строки, занимает всю ширину контейнера	, <p>, <section>

inline	Располагается в строку, не переносит элементы на новую линию	<style>
		div {
		  border: 1px solid gray;
		}
		span {
		  border: 1px dashed red;
		}
		</style>

При необходимости можно изменить тип элемента с помощью свойства display:

```
p {
  display: inline-block;
}
```

В результате абзацы выстроятся в одну линию и при этом сохранят возможность изменения ширины и высоты. Свойство display: определяет, как элемент участвует в потоке документа и как взаимодействует с соседними элементами.

Значение свойства display	Описание
block	Делает элемент блочным
inline	Делает элемент строчным
inline-block	Совмещает свойства обоих типов
flex	Включает флекс-контейнер
grid	Включает сетку
none	Полностью скрывает элемент

Пример использования inline-block для горизонтального меню:

```
<style>
.menu-item {
  display: inline-block;
  padding: 10px 15px;
  border: 1px solid #333;
}
</style>
<div class="menu-item">Главная</div>
<div class="menu-item">О нас</div>
<div class="menu-item">Контакты</div>
```

Раньше для создания колонок и обтекания текста применялся метод float. Сейчас его используют редко, но он все еще встречается в старых шаблонах.

```
<style>
.left {
  float: left;
```

```
  width: 50%;
  background: lightblue;
}
.right {
  float: right;
  width: 50%;
  background: lightgreen;
}
</style>

<div class="left">Левая колонка</div>
<div class="right">Правая колонка</div>
```

В целях учета высоты плавающих элементов в родительский блок добавляют «очистку» потока:

```
.clearfix::after {
  content: "";
  display: block;
  clear: both;
}
```

Свойство position предназначено для управления местоположением элемента относительно страницы или других элементов.

Значение	Особенности
static	Стандартное поведение, элементы идут в потоке
relative	Смещает элемент относительно исходного положения
absolute	Позиционирует относительно ближайшего позиционированного родителя
fixed	Фиксирует элемент относительно окна браузера
sticky	Смещает элемент, но «прилипает» при прокрутке

Пример фиксированной панели:

```
<style>
.header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  background: #333;
  color: white;
  padding: 10px;
}
</style>
<div class="header">Навигация</div>
```

А теперь приведем пример относительного и абсолютного позиционирования:

```
<style>
.container {
  position: relative;
  width: 300px;
  height: 200px;
  background: lightgray;
}
.box {
  position: absolute;
  top: 20px;
  right: 20px;
  background: coral;
  padding: 10px;
}
```

```
</style>
<div class="container">
  <div class="box">Абсолютный элемент</div>
</div>
```

Для удобного выравнивания и распределения элементов в строке или колонке также можно использовать Flexbox (Flexible Box Layout). Чтобы включить flex-режим, родителю задают display: flex.

```
<style>
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
.item {
  background: #ddd;
  padding: 20px;
}
</style>
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

Основные свойства контейнера.

Свойство	Назначение
justify-content	Выравнивание по горизонтали
align-items	Выравнивание по вертикали
flex-direction	Направление: row, column
gap	Расстояние между элементами

Для построения сложных макетов с рядами и колонками используется CSS Grid. Для активации сетки задается display: grid.

```
<style>
.grid {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
  gap: 10px;
}
.cell {
  background: lightblue;
  padding: 15px;
}
</style>
<div class="grid">
  <div class="cell">1</div>
  <div class="cell">2</div>
  <div class="cell">3</div>
</div>
```

Инструмент Grid является незаменимым при адаптивной верстке, так как позволяет точно управлять областями, размерами и выравниванием. Центрировать элемент можно с использованием трех популярных способов:

1. Через текстовое выравнивание (для inline-элементов):

```
.parent {
  text-align: center;
}
```

2. Через margin-auto (для блочных элементов):

```
.block {
  width: 200px;
  margin: 0 auto;
}
```

3. Через Flexbox (универсальный способ):

```
.center {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px;
}
```

Пример.

Ниже приведен простой макет страницы, комбинирующий несколько способов размещения:

```
<style>
body {
  margin: 0;
}
.header {
  position: fixed;
  top: 0;
  width: 100%;
  background: #333;
  color: white;
  padding: 10px;
  text-align: center;
}
.main {
  margin-top: 60px;
  display: flex;
  justify-content: space-around;
}
.section {
```

```
  background: #f3f3f3;
  width: 45%;
  padding: 20px;
 }
 .footer {
  text-align: center;
  padding: 10px;
  background: #222;
  color: #fff;
 }
 </style>
 <div class="header">Шапка сайта</div>
 <div class="main">
  <div class="section">Контент слева</div>
  <div class="section">Контент справа</div>
 </div>
 <div class="footer">Подвал</div>
```

Таким образом создается чистый, понятный макет с фиксированной шапкой, двумя колонками и подвалом.

Типичные ошибки при размещении элементов

При работе с макетами часто возникают проблемы, которые портят верстку и усложняют поддержку кода. Ниже – самые распространенные ошибки и как их избежать.

1. Смешивание несовместимых методов

Пытаются одновременно применять float, flex и position без необходимости. В итоге элементы ведут себя непредсказуемо.

✕ Плохо:

```
.block {
  float: left;
```

```
  display: flex;
  position: relative;
}
```

✅ Лучше: выбрать один метод под задачу. Например, Flexbox для строк и расположения по горизонтали.

2. Использование position: absolute для всего макета

Абсолютное позиционирование убирает элемент из потока. Если злоупотреблять этим свойством – макет «ломается» при малейшем изменении.

❌ Плохо:

```
.header {
  position: absolute;
  top: 0;
}
```

✅ Лучше:

```
.header {
  position: fixed; /* если нужен фиксированный хедер */
}
```

3. Забывают про box-sizing

Без box-sizing: border-box размеры блоков «плывут», потому что padding и border добавляются к ширине и высоте.

✅ Всегда добавляйте в начало проекта:

```
* {
  box-sizing: border-box;
}
```

4. Неправильное центрирование

Многие начинают двигать элемент «на глаз» отступами.

❌ Плохо:

```
.block {
  margin-left: 37%;
}
```

✅ Правильно:

```
.block {
  margin: 0 auto;
}
```

или

```
.parent {
  display: flex;
  justify-content: center;
}
```

5. Не учитывают адаптивность

При фиксированных ширинах макет ломается на телефонах.

❌ Плохо:

```
.column {
  width: 500px;
}
```

✅ Лучше:

```
.column {
  max-width: 500px;
  width: 100%;
}
```

6. Злоупотребляют !important

Пытаются «побороть» неправильную структуру CSS с помощью !important, что в будущем делает стили неконтролируемыми.

✅ Решение: работать с каскадом и специфичностью, а не с !important.

7. Отсутствие отладки через границы

Новички часто пытаются разобраться, что «поехало» в макете, не видя границ блоков.

✅ Совсем просто для отладки:

```
* {
  outline: 1px solid lightgray;
}
```

4.10. СКРЫТИЕ ЭЛЕМЕНТОВ В CSS

Иногда на странице нужно скрыть элемент: временно убрать блок, спрятать всплывающее окно до активации, скрыть часть интерфейса на мобильных устройствах или подготовить элемент к появлению через JavaScript. В CSS есть несколько способов сделать это, и каждый из них ведет себя по-разному. Важно понимать разницу, иначе можно случайно нарушить верстку или доступность сайта.

display: none – полное скрытие элемента

Самый простой и самый распространенный способ – скрыть элемент. Он полностью удаляет его из визуальной части страницы – элемент не занимает места и не влияет на расположение других блоков.

```
.hidden {
  display: none;
}
<p>Этот текст будет виден.</p>
<p class="hidden">А этот – скрыт полностью.</p>
```

Особенности: элемент полностью исчезает со страницы; другие элементы занимают его место; нельзя взаимодействовать с элементом с клавиатуры или мышью; часто используется вместе с JavaScript, чтобы показать/спрятать блок по клику.

```
button.addEventListener('click', () => {
  box.style.display = 'none';
});
```

visibility: hidden – скрытие с сохранением места

Этот способ тоже скрывает элемент, но при этом оставляет пустое пространство на его месте.

```
.invisible {
  visibility: hidden;
}
<p>Пункт меню 1</p>
<p class="invisible">Пункт меню 2 (спрятан)</p>
<p>Пункт меню 3</p>
```

Пользователь не видит второй пункт меню, но между соседними элементами остается зазор. Такой вариант подходит, когда нужно временно скрыть элемент, не нарушая построенную сетку.

opacity – визуальное скрытие без удаления

Иногда элемент нужно визуально скрыть, но при этом оставить его на странице, сохранив при этом место. В этом случае используют opacity.

```
.hidden-opacity {
  opacity: 0;
}
```

При этом элемент становится полностью прозрачным, но остается на странице, продолжает занимать место, остается кликабельным (можно нажать, даже если его не видно), доступен для программ чтения с экрана, участвует в анимациях (его удобно плавно показывать и скрывать). Чаще всего этот способ применяют вместе с JavaScript и CSS-анимациями:

```
.fade {
  transition: opacity 0.3s;
}
.fade.hidden {
  opacity: 0;
}
.fade.visible {
  opacity: 1;
}
```

Перемещение за пределы экрана

Иногда элемент нужно спрятать, но оставить доступным для чтения экрана и SEO. Это порой требуется для скрытых заголовков, которые помогают алгоритмам понимать структуру страницы, но не должны мешать дизайну.

```
.visually-hidden {
  position: absolute;
  left: -9999px;
}
<h1 class="visually-hidden">Каталог электроники</h1>
```

overflow – скрытие части содержимого

Свойство overflow не скрывает сам элемент, а ограничивает его содержимое. Оно используется, когда внутри блока информации больше, чем он может вместить.

```
.box {
  width: 200px;
  height: 100px;
  overflow: hidden;
}
```

Если содержимого слишком много, все, что выходит за границы блока – будет скрыто. Часто это применяют в карточках товаров, превью новостей и текстовых блоках. Можно использовать хитрый прием – добавление полосы прокрутки:

```
.scroll {
  width: 200px;
  height: 100px;
  overflow: auto;
}
```

Разные методы скрытия работают по-разному, поэтому важно понимать, какой метод, когда применять.

Скрытие в адаптивной верстке

При адаптивной верстке часто нужно показывать одни элементы на компьютере и скрывать на телефоне – и наоборот. Для этого используют медиа-запросы:

```
/* Скрыть на мобильных */
@media (max-width: 768px) {
  .sidebar {
    display: none;
  }
}
```

И наоборот:

```
/* Скрыть на десктопах */
@media (min-width: 769px) {
  .mobile-menu {
    display: none;
  }
}
```

Так делают, например, при верстке меню: на компьютере – обычная навигация, на телефоне – бургер-кнопка.

Сравнение способов скрытия элементов

Метод	Видимость	Место в верстке	Можно кликнуть	Поддерживает анимацию	SEO/ доступность
display: none	✗ Нет	✗ Нет	✗ Нет	⛔ Нет	✗ Прячется
visibility: hidden	✗ Нет	✓ Да	✗ Нет	⛔ Нет	⚠ Частично
opacity: 0	✗ Нет	✓ Да	✓ Да	✓ Да	✓ Да
overflow: hidden	✓ Частично	✓ Да	✓ Да	⚠ Иногда	✓ Да
position: absolute; left: -9999px	✗ Нет	✗ Нет	✗ Нет	⛔ Нет	✓ Да
.sr-only	✗ Нет	✗ Нет	✓ Да	⛔ Нет	✓ Отлично

4.11. ПСЕВДОКЛАССЫ В CSS

При создании интерфейсов часто нужно управлять стилями не только для статичных элементов, но и для их состояний. Например, как должен выглядеть элемент при наведении курсора, когда он активен, выделен, нажат или находится в фокусе. Для таких ситуаций в CSS существуют псевдоклассы.

Псевдокласс – это особое состояние элемента, которое нельзя описать с помощью обычного класса или тега. Он записывается через двоеточие (:) и добавляется к селектору. Псевдоклассы чаще всего используются при работе с кнопками, ссылками и формами.

Основные интерактивные псевдоклассы

Эти псевдоклассы используются чаще всего, особенно при работе с кнопками, ссылками и формами.

Псевдокласс	Описание	Пример кода

:hover	Псевдокласс срабатывает при наведении пользователем курсора на элемент. Используется для добавления интерактивности – например, небольшого затемнения кнопки при наведении, а также для имитации нажатия – элемент визуально «прижимается»	button:hover {
		background-color: #eee;
		}

:focus	Фокус используется для улучшения доступности элементов, он срабатывает, когда элемент активен через клавиатуру или форму, например, при клике по полю ввода	input:focus {
		outline: 2px solid #007bff;
		}

:visited и :link	Эти псевдоклассы позволяют стилизовать посещенные и непосещенные ссылки	a:link {
		color: blue;
		}
		a:visited {
		color: purple;
		}

Структурные псевдоклассы: выбор по положению в разметке

Иногда нужно управлять стилями не по классу или id, а по расположению элемента. Например, убрать лишний отступ у первого абзаца, выделить каждую вторую строку таблицы или стилизовать только третий пункт списка. Для таких задач используют структурные псевдоклассы.

Псевдокласс	Пример	Применение
:first-child – первый элемент	p:first-child	Убрать первый отступ
:last-child – последний элемент	p:last-child	Убрать нижний отступ
:nth-child(n) – элемент по номеру	li:nth-child(3)	Третий элемент списка

:nth-child(odd) – нечетные элементы	tr:nth-child(odd)	Полосатые таблицы
:nth-child(even) – четные элементы	tr:nth-child(even)	Улучшение читаемости
:only-child – единственный элемент	.item:only-child	Особый стиль
:only-of-type – единственный среди своего типа	h2:only-of-type	Уникальный заголовок

Пример использования:

```
<ul>
  <li>Первый</li>
  <li>Второй</li>
  <li>Третий</li>
</ul>
li:first-child {
    font-weight: bold;
}
li:nth-child(2) {
    color: darkred;
}
li:last-child {
    text-decoration: underline;
}
```

Псевдоклассы взаимодействия

Теперь переходим к псевдоклассам, которые особенно полезны в интерактивных интерфейсах: они реагируют на действия пользователя, делают элементы удобнее и понятнее, помогают с доступностью. Эти псевдоклассы чаще всего используются с кнопками, ссылками, формами и интерактивными блоками.

Псевдокласс	Пример	Применение
:hover Реакция на наведение курсора	.card:hover { box-shadow: 0 4px 10px rgba(0,0,0,.2); }	Подсветка элементов
:active активность В момент клика	button:active { transform: scale(.97); }	Визуальный эффект «нажатия»
:focus Элемент в фокусе	input:focus { outline: 2px solid blue; }	Доступность элемента для ввода информации с клавиатуры
:focus-within Фокус внутри контейнера	.form-group:focus-within { border-color: teal; }	Подсветка активных блоков

Пример для формы (поле подсвечивается, а весь блок становится активным). Такой эффект часто встречается на современных сайтах, он помогает интуитивно понять, что происходит.

```
<div class="form-group">
  <label>Ваше имя</label>
  <input type="text" />
</div>
.form-group {
    border: 1px solid #ccc;
    padding: 10px;
    transition: 0.3s;
}
.form-group:focus-within {
    border-color: #007acc; /* подсветили блок, когда в нем фокус */
}
input:focus {
    outline: none;
}
```

Псевдоклассы формы и валидации

Работая с формами, важно давать пользователю подсказки: какое поле обязательно, где ошибка, какое значение введено верно. CSS умеет реагировать на такие состояния – без единой строки JavaScript – с помощью специальных псевдоклассов.

Псевдокласс	Пример	Применение
:required Поле обязательно к заполнению	input:required	Подсветка обязательных полей
:optional Поле не обязательно	input:optional	Подстветка необязательных полей (используется редко)
:valid Значение введено корректно	input:valid	Подтверждение правильного ввода
:invalid Ошибка в данных	input:invalid	Валидация без JS
:in-range Значение в диапазоне	input:in-range	Для числовых полей
:out-of-range Значение вне допустимого диапазона	input:out-of-range	Контроль диапазона
:disabled Поле отключено	input:disabled	Нерабочие поля
:enabled Поле активно	input:enabled	Обычные поля
:read-only Только для чтения	input:read-only	В полях без редактирования
:checked Отмеченные чекбоксы/радиокнопки	input:checked	Переключатели и опросы

Пример простой валидации без JavaScript, здесь поле само проверяет email.

```
<form>
  <input type="email" placeholder="Email" required />
  <button type="submit">Отправить</button>
```

```
</form>
input {
    border: 1px solid #aaa;
    padding: 8px;
    margin-bottom: 10px;
    display: block;
}
input:required {
    border-left: 3px solid crimson;
}
input:valid {
    border-color: green;
}
input:invalid {
    border-color: red;
}
```

Псевдокласс :not() и логические селекторы

Иногда нужно выбрать почти все элементы, но исключить один конкретный вариант. Например, стилизовать все ссылки, кроме тех, что ведут наружу, задать отступ всем элементам списка, кроме последнего. Для таких случаев есть псевдокласс :not() – он срабатывает на все, что не попадает под условие.

Пример	Применение
p:not(.lead)	Исключение всего абзаца, кроме тех, у кого класс .lead
a:not([href^="http"])	Все ссылки, кроме внешних
div:not(:first-of-type)	Все div, кроме первого
li:not(:last-child)	Все пункты списка, кроме последнего

Пример с отступами:

```
li {
   margin-bottom: 10px;
}
li:not(:last-child) {
   border-bottom: 1px solid #ddd;
}
```

4.12. CSS-СЧЕТЧИКИ

Иногда верстка требует автоматической нумерации: надо подписать главы документа, пронумеровать таблицы, добавить номера к пунктам FAQ или оформить список задач. Можно было бы написать все вручную, но это неудобно – если добавить новый пункт, вся нумерация «поедет». Именно для таких случаев в CSS есть счетчики – скрытый, но очень полезный инструмент.

CSS-счетчики работают как переменные: их можно создать, увеличивать и выводить прямо в стилях с помощью псевдоэлементов и специальных свойств. Главное преимущество – они позволяют генерировать нумерацию автоматически, без JavaScript и без дополнительной разметки в HTML.

Чтобы разобраться в CSS-счетчиках, нужно знать три ключевых свойства:

Свойство	Назначение
counter-reset	Создает/сбрасывает счетчик
counter-increment	Увеличивает значение счетчика
content: counter(name)	Выводит значение счетчика

Давайте начнем с наглядного случая – создадим список, который будет нумероваться не с помощью <ol>, а с помощью CSS-счетчиков:

```
<ul class="tasks">
  <li>Купить дом</li>
```

```
  <li>Посадить дерево</li>
  <li>Вырастить сына</li>
</ul>
.tasks {
  counter-reset: myCounter; /* создаем счетчик */
}
.tasks li {
  counter-increment: myCounter; /* увеличиваем */
}
.tasks li::before {
  content: counter(myCounter) ". "; /* выводим номер */
  font-weight: bold;
}
```

Результат:

1. Купить дом
2. Посадить дерево
3. Вырастить сына

CSS позволяет использовать несколько разных счетчиков одновременно. Например, можно сделать нумерацию глав и подглав как в учебнике.

```
<h2>Введение</h2>
<h2>Основы CSS</h2>
<h3>Селекторы</h3>
<h3>Свойства</h3>
<h2>Продвинутая часть</h2>
CSS:
body {
  counter-reset: chapter; /* создаем счетчик глав */
}
h2 {
```

```
  counter-increment: chapter; /* увеличиваем номер главы */
  counter-reset: sub; /* сбрасываем счетчик подглав при каждой новой главе */
}
h3 {
  counter-increment: sub; /* увеличиваем номер подглавы */
}
h2::before {
  content: "Глава " counter(chapter) ". ";
  font-weight: bold;
}
h3::before {
  content: counter(chapter) "." counter(sub) " ";
  color: gray;
}
```

Результат будет таким:

Глава 1 Введение

Глава 2 Основы CSS

2.1 Селекторы

2.2 Свойства

Глава 3 Продвинутая часть

Иногда одного счетчика мало. Например, нужно объединить все уровни списка в одну строку. Тут пригодится функция counters() (обратите внимание на букву s на конце – это уже другая функция!).

```
.plan li::before {
  content: counters(main, ".") " ";
}
```

Это соберет все уровни в одну цепочку, вроде:

1

1.1

1.1.1

CSS-счетчики отлично подходят в ситуациях, когда нужно: автоматически нумеровать пункты без изменений HTML; поддерживать структуру длинных документов (главы, разделы, списки); делать красивые списки и навигацию; оформлять учебные материалы, технические статьи, инструкции; избавиться от JavaScript там, где он не обязателен.

Несмотря на удобство, у счетчиков есть свои особенности: не работают со старыми браузерами IE; нельзя использовать в динамических списках без перерисовки; плохо подходят, если нужна интерактивность или сложная логика; читаемость стилей может пострадать, если переборщить с вложенностью.

Если требуется реагировать на действия пользователя или данные меняются через JS, то в этом случае лучше комбинировать CSS-счетчики с JavaScript.

4.13. АДАПТИВНЫЙ ДИЗАЙН В CSS

Еще десять лет назад сайты в основном открывали с компьютеров. Сегодня все наоборот – половина трафика идет со смартфонов. Одну и ту же страницу теперь просматривают на разных экранах: от широких мониторов до узких телефонов. И если сайт «ломается» на маленьком экране или приходится листать его по горизонтали, то пользователь просто закрывает вкладку.

Адаптивный дизайн решает эту проблему. Его задача – сделать так, чтобы страница выглядела аккуратно и читалась удобно на любом устройстве, без отдельной мобильной версии сайта. Этого можно добиться уже на уровне CSS – без тяжелых библиотек и сложных технологий.

Адаптивный дизайн строится на нескольких принципах:

1. Гибкая сетка

Страница должна растягиваться вместе с экраном. Вместо фиксированных ширин используются относительные единицы: %, vh, vw, fr.

```
.container {
  width: 90%;   /* вместо 1200px */
  margin: 0 auto;
}
```

2. Элементы, которые подстраиваются

Картинки, видео и блоки не должны «вылезать» за рамки экрана. Это делается буквально одной строкой:

```
img {
  max-width: 100%;
  height: auto;
}
```

3. Медиазапросы

Это сердце адаптивной верстки. С их помощью можно менять стили в зависимости от ширины экрана. Например, перестраивать меню или колонки.

```
@media (max-width: 768px) {
  .sidebar {
    display: none;
  }
}
```

Когда речь заходит об адаптивной верстке, просто медиазапросов уже недостаточно. Нужно уметь перестраивать макет, менять количество колонок, порядок блоков и выравнивание элементов. Для этого в CSS есть два мощных инструмента: Flexbox и Grid.

Flexbox управляет расположением, порядком, размерами и выравниванием элементов внутри «флекс-контейнера» и он идеально подходит для адаптивных рядов: меню, карточек, галерей, кнопок.

Простой пример:

```
.cards {
  display: flex;
  gap: 20px;
}
.card {
  flex: 1;
}
```

Так карточки автоматически распределятся по строке и займут доступное пространство. Но самое интересное – они легко становятся адаптивными:

```
@media (max-width: 768px) {
  .cards {
    flex-direction: column; /* карточки становятся вертикальными */
  }
}
```

То есть на большом экране – три карточки в ряд, на телефоне – каждая в своей строке.

Grid применяется для сложных макетов. Если нужно построить страницу с разными зонами (шапка, боковая панель, контент, подвал), Grid справится лучше Flexbox. Он позволяет буквально рисовать таблицу из областей.

```
.layout {
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-areas:
```

```
    "header header"
    "sidebar content"
    "footer footer";
}
.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.content { grid-area: content; }
.footer { grid-area: footer; }
```

На компьютере все выглядит как полноценный сайт, а на телефоне оформление можно перестроить:

```
@media (max-width: 600px) {
  .layout {
    grid-template-columns: 1fr;
    grid-template-areas:
      "header"
      "content"
      "sidebar"
      "footer";
  }
}
```

Боковая панель аккуратно переедет вниз, контент останется на первом месте – все логично и удобно пользователю.

Даже идеально сверстанный сайт будет выглядеть плохо, если текст слишком мелкий на телефоне или слишком огромный на ноутбуке. Поэтому шрифты, отступы и размеры элементов тоже должны подстраиваться под устройство. Для этого нужно правильно выбрать единицы измерения. Ни в коем случае не следует все задавать в px (пикселях), в адаптивной верстке лучше использовать относительные единицы:

Единица	Использование
% от ширины	Ширина блоков
em от размера шрифта	Внутренние отступы
rem от базового шрифта страницы	Шрифты
vw, vh от ширины/высоты окна	Крупные заголовки

Чтобы текст плавно подстраивался под ширину экрана, используют функцию clamp():

```
h1 {
  font-size: clamp(24px, 5vw, 40px);
}
```

Это значит: не меньше 24px, не больше 40px, а между ними шрифт растет плавно вместе с экраном.

Верстальщик может сделать гибкую сетку, аккуратные блоки и плавную типографику, но все это рухнет, если изображения на сайте не адаптивны. Огромные картинки, обрезанные фото, «поехавшие» баннеры – одна из самых частых проблем на мобильных устройствах.

Изображения должны подстраиваться под контейнер:

```
img {
  max-width: 100%;
  height: auto;
  display: block;
}
```

max-width: 100% – не дает изображению выходить за пределы блока

height: auto – сохраняет пропорции

display: block – убирает лишние отступы снизу

Если картинки должны заполнять фиксированные блоки (например, карточки товара), используют:

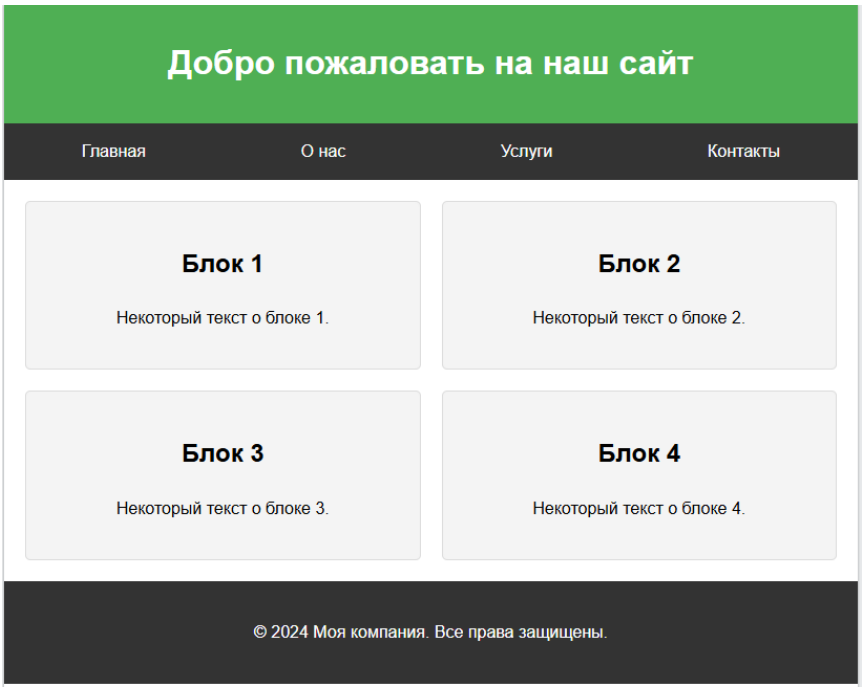
```
.card img {
  width: 100%;
  height: 200px;
  object-fit: cover; /* обрезает красиво */
}
```

Приведем простой пример HTML-страницы с адаптивным дизайном с использованием CSS для стилизации. Эта страница будет состоять из заголовка, навигационного меню и нескольких блоков с контентом, а дизайн сайта будет адаптироваться под различные размеры экранов с использованием медиазапросов.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Адаптивная Страница</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            line-height: 1.6;
        }

        header {
            background: #4CAF50;
            color: white;
            padding: 10px 20px;
            text-align: center;
        }

        nav {
            display: flex;
            justify-content: space-around;
            background: #333;
        }

        nav a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }

        nav a:hover {
            background: #575757;
        }
```

```
        .container {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
            padding: 20px;
            gap: 20px;
        }
        .card {
            background: #f4f4f4;
            border: 1px solid #ddd;
            border-radius: 5px;
            padding: 20px;
            text-align: center;
        }
        footer {
            text-align: center;
            padding: 20px;
            background: #333;
            color: white;
        }
        @media (max-width: 768px) {
            nav {
                flex-direction: column;
            }

            nav a {
                padding: 10px;
            }
        }
    </style>
</head>
```

Результат



5. ПРИМЕРЫ ОФОРМЛЕНИЯ WEB-СТРАНИЦ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА СТИЛЕЙ CSS

5.1. ТРАНСФОРМАЦИИ, ПЕРЕХОДЫ И АНИМАЦИИ

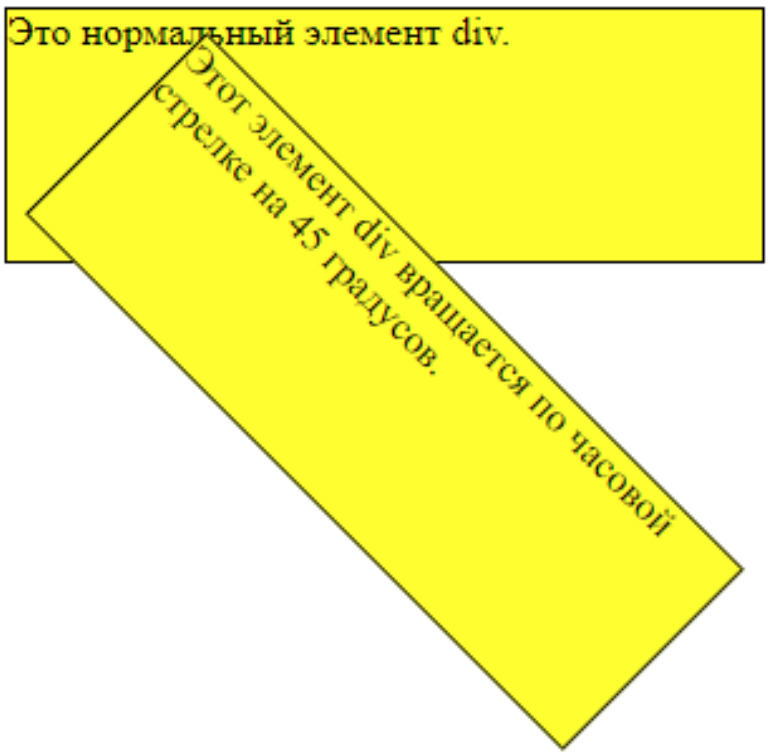
К трансформациям относятся такие действия, как вращение элемента, его масштабирование, наклон или перемещение по вертикали или горизонтали. Для создания трансформаций в CSS3 применяется свойство transform. На веб-страницах можно применить несколько видов CSS-трансформаций.

Название трансформации	Описание
Перемещение (translate)	Изменение положения элемента на странице
Изменение размера (scale)	Увеличение или уменьшение размера элемента
Вращение (rotate)	Поворот элемента на определенный угол
Наклон (skew)	Наклон элемента по горизонтали или вертикали
Изменение формы (transform)	Изменение формы элемента
Прозрачность (opacity)	Изменение прозрачности элемента
Анимация (animation)	Создание анимации элемента

Конечно, применение CSS-трансформаций может улучшить внешний вид веб-страницы, но, однако, не следует злоупотреблять этими эффектами, так как это может привести к замедлению загрузки страницы.

Пример. Поворот элемента на 45 градусов.

```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

div#myDiv {
  -ms-transform: rotate(45deg); /* IE 9 */
  -webkit-transform: rotate(45deg); /* Safari */
  transform: rotate(45deg); /* Стандартный синтаксис */
}
</style>
</head>
<body>

<div>
Это нормальный элемент div.
</div>

<div id="myDiv">
Этот элемент div вращается по часовой стрелке на 45 градусов.
</div>
```



Переход (transition) представляет анимацию от одного стиля к другому в течение определенного периода времени. Для создания перехода необходимы, прежде всего, два набора свойств CSS: начальный стиль, который будет иметь элемент в начале перехода, и конечный стиль – результат перехода. Рассмотрим простейший переход:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Переход в CSS3</title>
        <style>
            div{
                width: 100px;
                height: 100px;
                margin: 40px 30px;
                border: 1px solid #333;

                background-color: #ccc;
                transition-property: background-color;
                transition-duration: 2s;
            }
            div:hover{
                background-color: red;
            }
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

В этом примере анимируется свойство background-color элемента div. При наведении указателя мыши на элемент он будет менять цвет с серого на красный.

Чтобы включить переход, нужно указать хотя бы свойство и длительность. Но на самом деле возможностей больше – можно управлять скоростью, задержкой и даже типом плавности.

Свойство	Назначение	Пример
transition-property	Указывает, какое свойство будет анимироваться	transition-property: color;
transition-duration	Длительность перехода	transition-duration: 0.3s;
transition-timing-function	Стиль плавности (ускорение/замедление)	transition-timing-function: ease-in;
transition-delay	Задержка перед стартом	transition-delay: 0.2s;
transition	Сокращенная запись всех параметров	transition: color 0.3s ease;

Анимации в CSS предназначены для «оживления» интерфейса без единой строки JavaScript. Они помогают смягчить резкие изменения на странице: появление меню, смену цвета кнопки при наведении, плавный въезд блока при прокрутке. В итоге сайт выглядит чище и профессиональнее, а взаимодействие с ним становится приятнее.

CSS-анимации строятся вокруг двух вещей: ключевых кадров и свойства animation. Ключевые кадры описывают, что должно измениться, а animation – как долго и каким образом это произойдет.

Для начала нужен объект, которому мы хотим задать анимацию:

```
<div class="box"></div>
.box {
   width: 80px;
   height: 80px;
   background: coral;
   animation: move 2s;
}
```

Теперь опишем ключевые кадры – движение слева направо:

```
@keyframes move {
   from {
```

```
    transform: translateX(0);
  }
  to {
    transform: translateX(200px);
  }
}
```

В данном примере:

@keyframes move – объявление анимации с именем move;

from → начальное состояние;

to → конечное состояние;

animation: move 2s – привязали анимацию и указали ее длительность.

Иногда «от точки А до точки Б» мало. Можно разбить процесс на этапы с помощью процентов:

```
@keyframes colorRun {
  0% {
    background: coral;
  }
  50% {
    background: gold;
  }
  100% {
    background: limegreen;
  }
}
.box {
  animation: colorRun 3s;
}
```

Так элемент плавно меняет цвета – от кораллового до зеленого.

Чтобы управлять поведением анимации, используется следующий набор свойств.

Свойство	Назначение	Пример
animation-name	Имя анимации	animation-name: move;
animation-duration	Длительность	animation-duration: 2s;
animation-timing-function	Скорость анимации	animation-timing-function: ease;
animation-delay	Задержка перед началом	animation-delay: 1s;
animation-iteration-count	Количество повторов	animation-iteration-count: infinite;
animation-direction	Направление	animation-direction: alternate;
animation-fill-mode	Сохранение состояния	animation-fill-mode: forwards;
animation-play-state	Пауза/запуск	animation-play-state: paused;

Пример.

```
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  -webkit-animation-name: example; /* Safari 4.0 - 8.0 */
  -webkit-animation-duration: 4s; /* Safari 4.0 - 8.0 */
  animation-name: example;
  animation-duration: 4s;
}

/* Safari 4.0 - 8.0 */
@-webkit-keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* Standard syntax */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
<body>

   <div></div>

<p><b>Примечание:</b> Когда анимация закончена, она возвращается к исходному стилю.</p>
```

5.2. ЗАКРУГЛЕННЫЕ УГЛЫ

Иногда элементу на странице не хватает мягкости. Все слишком прямое, жесткое, угловатое. Закругленные углы решают эту проблему буквально одной строчкой кода – свойством border-radius. Это не только эстетика, но и важная деталь интерфейса: с его помощью делают кнопки, карточки, уведомления, аватары и даже круги.

Самый простой способ применить скругление – указать одно значение. Тогда радиус закругления применяется сразу ко всем четырем углам.

```
.box {
   border: 2px solid #3498db;
   border-radius: 10px;
   padding: 15px;
   width: 200px;
}
```

Чем больше число – тем сильнее закругление. Укажем 50px – и блок станет почти «пузатым»:

```
border-radius: 50px;
```

CSS дает полную свободу: можно скруглить только одну сторону или сделать каждый угол своим. Значения указываются в порядке по часовой стрелке, начиная с верхнего левого угла:

```
border-radius: 10px 20px 30px 40px;
```

Можно еще короче – скруглить только один угол:

```
border-top-left-radius: 15px;
border-bottom-right-radius: 25px.
```

Круг получается легко: если высота и ширина равны, а border-radius – 50%, элемент станет идеальным кругом.

```
.avatar {
   width: 120px;
```

height: 120px;

background: #f39c12;

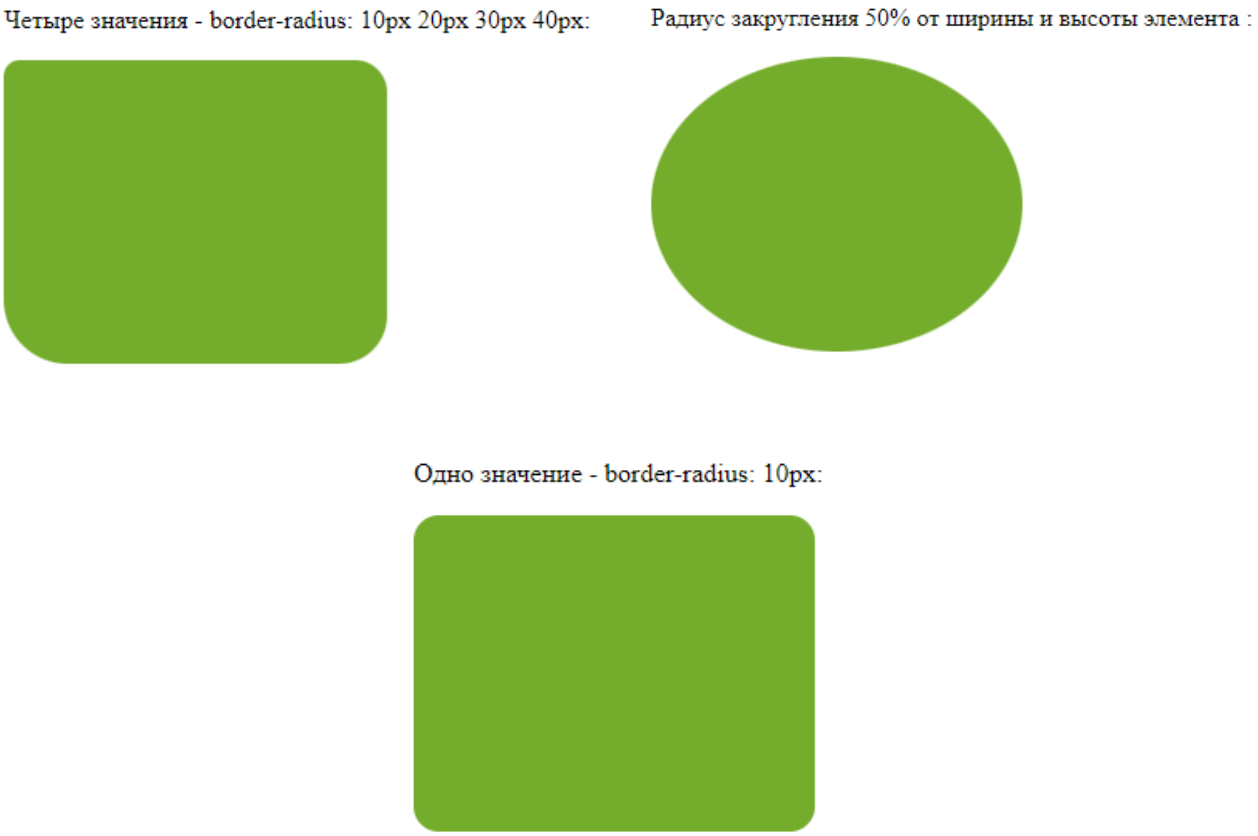
border-radius: 50%;

}

Используется часто – для кнопок в стиле социальных сетей, фото профиля и иконок.

Пример.

```
<style>
#rcorners1 {
  border-radius: 10px 20px 30px 40px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}

#rcorners2 {
  border-radius: 50%;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}

#rcorners3 {
  border-radius: 15px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
</style>
</head>
<body>

<p>Четыре значения - border-radius: 10px 20px 30px 40px:</p>
<p id="rcorners1"></p>

<p> Радиус закругления 50% от ширины и высоты элемента :</p>
<p id="rcorners2"></p>

  <p>Одно значение - border-radius: 10px:</p>
<p id="rcorners3"></p>
```



5.3. CSS-ТЕНЕВЫЕ ЭФФЕКТЫ

Тень в веб-дизайне помогает выделить важные блоки, создать глубину и визуальную иерархию. Благодаря свойству box-shadow в CSS можно сделать мягкую тень, плотную, многослойную или динамическую при наведении. И все это – без графических редакторов и лишних изображений.

У тени есть несколько параметров, которые задаются последовательно в свойстве box-shadow. Это смещение по горизонтали (смещение X), смещение по вертикали (смещение Y), размытие, растяжение и цвет.

```
.box {
    width: 200px;
    padding: 20px;
    background: white;
    box-shadow: 4px 4px 10px rgba(0,0,0,0.2);
}
```

Разберем каждый параметр подробно.

Параметр	Действие	Пример
Смещение X	Тень вправо/влево	5px – вправо, -5px – влево
Смещение Y	Тень вниз/вверх	5px – вниз, -5px – вверх
Размытие	Делает тень мягче	15px
Растяжение	Увеличивает площадь тени	10px
Цвет	Любой цвет + rgba	rgba(0,0,0,0.3)
inset	Внутренняя тень	inset 0 0 5px #000

3. Цвет тени: зачем использовать rgba

Можно указать цвет просто:

box-shadow: 3px 3px 8px black;

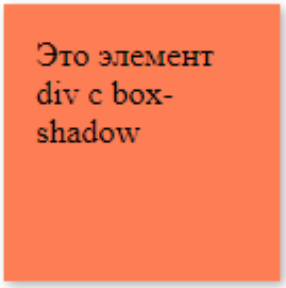
Но лучше так:

box-shadow: 3px 3px 8px rgba(0,0,0,0.25);

В этом случае rgba дает полупрозрачность, а значит тень будет мягкой и ближе к реалистичности.

Пример.

```
 <style>
 div {
   width: 100px;
   height: 100px;
   padding: 15px;
   background-color: coral;
   box-shadow: 2px 2px 5px rgba(0,0,0,0.3);
 }
 </style>
</head>
<body>
   <div>Это элемент div с box-shadow</div>
</body>
```

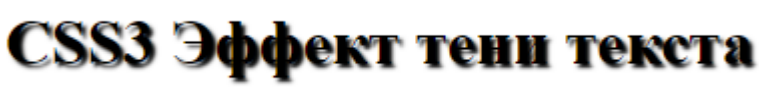


5.4. CSS-ЭФФЕКТЫ ТЕКСТА

С помощью текста на странице передают смысл, эмоцию и настроение. Если его чуть улучшить визуально, сайт смотрится аккуратнее, профессиональнее и приятнее для чтения. CSS дает немало простых, но полезных инструментов для оформления текста – от теней до эффектов при наведении. Приведем примеры наиболее часто используемых эффектов текста.

1. Тень текста

```
  <style>
  h1 {
    text-shadow: 2px 2px 2px #000;
  }
  </style>
 </head>
 <body>

  <h1>CSS3 Эффект тени текста</h1>

   </body>
</html>
```

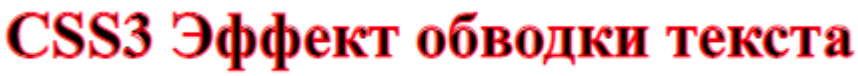


2. Текст на градиентном фоне

```
  <style>
  h1 {
    background-clip: text; background-image: linear-gradient(to right, #f00, #00f);
  }
  </style>
 </head>
 <body>

  <h1>CSS3 Эффект градиента текста</h1>

   </body>
</html>
```

3. Обводка текста

```
<style>
h1 {
 -webkit-text-stroke: 1px #FF0000;
}
</style>
</head>
<body>

<h1>CSS3 Эффект обводки текста</h1>
```



4. Градиентный текст

```
<style>
h1 {
background: linear-gradient(to right, #f00, #0f0, #00f);
-webkit-background-clip: text;
-webkit-text-fill-color: transparent;
color: #0B2349;
display: table;
margin: 20px auto;
}
</style>
</head>
<body>

<h1>CSS3 Градиентный текст</h1>

</body>
</html>
```

5. Текст с эффектом размытия

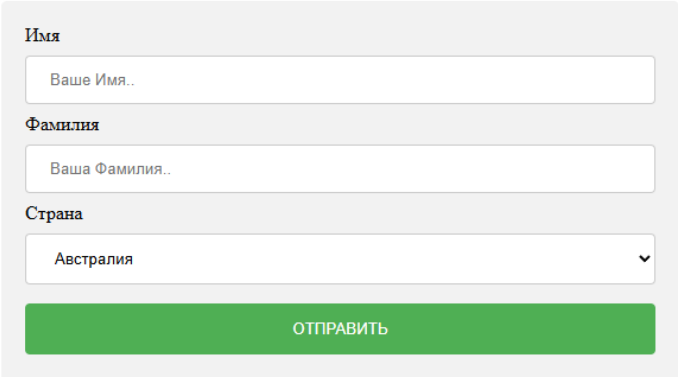
```
  <style>
  h1 {
      filter: blur(2px);
  }
  </style>
 </head>
 <body>

  <h1>CSS3 Размытый текст</h1>

   </body>
</html>
```

5.5. CSS-ФОРМЫ

Без форм не обходится ни один сайт: регистрация, поиск, подписка – все это формы. Но по умолчанию поля ввода выглядят скучно и по-разному в различных браузерах. CSS помогает привести их к единому стилю и сделать приятными для пользователя. Приведем пример простейшей формы, где мы задаем цвет фона, отступы, радиус границы и другие свойства для ее элементов.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
  input[type=text], select {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
  }
```

```
  input[type=submit] {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }

  input[type=submit]:hover {
    background-color: #45a049;
  }
```



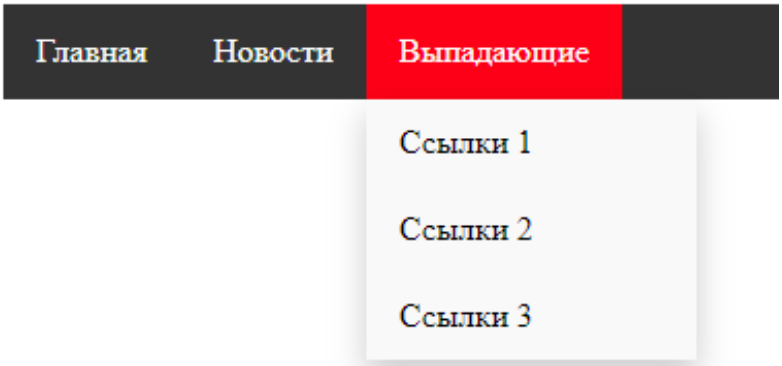

```
 div {
   border-radius: 5px;
   background-color: #f2f2f2;
   padding: 20px;
 }
 </style>
</head>
<body>

  <div>
   <form action="/action_page.php">
     <label for="fname">Имя</label>
     <input type="text" id="fname" name="firstname" placeholder="Ваше Имя..">

     <label for="lname">Фамилия</label>
     <input type="text" id="lname" name="lastname" placeholder="Ваша Фамилия..">

     <label for="country">Страна</label>
     <select id="country" name="country">
       <option value="australia">Австралия</option>
       <option value="canada">Канада</option>
       <option value="usa">США</option>
     </select>

     <input type="submit" value="ОТПРАВИТЬ">
   </form>
 </div>

</body>
</html>
```

5.6. CSS ВЫПАДАЮЩЕЕ МЕНЮ

Выпадающее меню используют для экономии места на странице и организации навигации. Такое меню можно сделать даже без JavaScript – достаточно средств CSS. Приведем пример выпадающего меню, где мы создаем список ссылок, псевдокласс :hover для элементов списка, чтобы показывать выпадающее меню при наведении курсора мыши на элемент, а также будем использовать позиционирование и задавать цвет фона для стилизации выпадающего меню.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
  }

  li {
    float: left;
  }

  li a, .dropbtn {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
  }

  li a:hover, .dropdown:hover .dropbtn {
    background-color: red;
  }

  li.dropdown {
    display: inline-block;
  }
```

```
  .dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
  }

  .dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
  }

  .dropdown-content a:hover {background-color: #f1f1f1}

  .dropdown:hover .dropdown-content {
    display: block;
  }
  </style>
 </head>
 <body>
  <ul>
    <li><a href="#home">Главная</a></li>
    <li><a href="#news">Новости</a></li>
    <li class="dropdown">
      <a href="javascript:void(0)" class="dropbtn">Выпадающие</a>
      <div class="dropdown-content">
        <a href="#">Ссылки 1</a>
        <a href="#">Ссылки 2</a>
        <a href="#">Ссылки 3</a>
      </div>
    </li>
  </ul>

 </body>
</html>
```



5.7. CSS-ГАЛЕРЕЯ ИЗОБРАЖЕНИЙ

Для того чтобы показать несколько фотографий в аккуратной сетке, используют галерею изображений. Такой блок часто встречается в портфолио, блогах или интернет-магазинах. Приведем пример простой адаптивной галереи на чистом CSS без лишнего кода.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
  div.gallery {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 180px;
  }
  div.gallery:hover {
    border: 1px solid #777;
  }
  div.gallery img {
    width: 100%;
    height: auto;
  }
  div.desc {
    padding: 15px;
    text-align: center;
  }
  </style>
```

```
</head>
<body>
 <h1>CSS Галерея изображений</h1>
 <div class="gallery">
   <a target="_blank" href="img_5terre.jpg">
     <img src="img_5terre.jpg" alt="Пять земли" width="600" height="400">
   </a>
   <div class="desc"> Фото 1</div>
 </div>
 <div class="gallery">
   <a target="_blank" href="img_forest.jpg">
     <img src="img_forest.jpg" alt="Лес" width="600" height="400">
   </a>
   <div class="desc">Фото 2</div>
 </div>
 <div class="gallery">
   <a target="_blank" href="img_lights.jpg">
     <img src="img_lights.jpg" alt="Северное сияние" width="600"
height="400">
   </a>
   <div class="desc">Фото 3</div>
 </div>
  </body>
</html>
```

CSS Галерея изображений

5.8. CSS-ТАБЛИЦЫ

Чтобы таблица не выглядела скучно и с ней было удобно работать, к ней применяют стили оформления. Приведем пример, как с помощью CSS превратить ее в аккуратный и читабельный элемент страницы.

1. Базовый HTML

```
<table class="price-table">
  <tr>
    <th>Услуга</th>
    <th>Цена</th>
    <th>Срок</th>
  </tr>
  <tr>
    <td>Верстка HTML-страницы</td>
    <td>3000 ₽</td>
    <td>2 дня</td>
  </tr>
  <tr>
    <td>Настройка CSS</td>
    <td>1500 ₽</td>
    <td>1 день</td>
  </tr>
  <tr>
    <td>Адаптивная верстка</td>
    <td>2000 ₽</td>
    <td>1 день</td>
  </tr>
</table>
```

2. Простая стилизация таблицы

```
.price-table {
    width: 100%;
    border-collapse: collapse; /* убираем двойные границы */
    margin-top: 10px;
    text-align: left;
}
```

3. Границы и отступы

```
.price-table th,
.price-table td {
    border: 1px solid #ccc;
    padding: 8px 12px;
}
```

4. Заголовок таблицы

```
.price-table th {
    background-color: #f2f2f2;
    font-weight: bold;
}
```

5. Полосатые строки (эффект зебры)

```
.price-table tr:nth-child(even) {
    background-color: #fafafa;
}
```

nth-child(even) выделяет каждую вторую строку.

6. Подсветка строки при наведении

```
.price-table tr:hover {
    background-color: #e8f4ff;
}
```

7. Адаптивность (скролл на телефонах)

Таблицы плохо сжимаются на маленьких экранах, поэтому добавим прокрутку:

```
.price-table {
  display: block;
  overflow-x: auto;
  white-space: nowrap;
}
```

Услуга	Цена	Срок
Верстка HTML-страницы	3000 ₽	2 дня
Настройка CSS	1500 ₽	1 день
Адаптивная верстка	2000 ₽	1 день
Оптимизация сайта	2500 ₽	3 дня

5.9. CSS-СПИСКИ

С помощью CSS-списки можно превратить из обычного текста с точками в полноценный элемент интерфейса. Можно создать списки с различными маркерами, установить в качестве маркера изображение, применить цвет фона для элементов списка. Рассмотрим примеры:

1) Различные маркеры элементов списка с использованием свойства list-style-type

```
<style>
ul.a {
  list-style-type: circle;
}

ul.b {
  list-style-type: square;
}

ol.c {
  list-style-type: upper-roman;
}

ol.d {
  list-style-type: lower-alpha;
}
</style>
```

CSS Разные маркеры списка

Пример неупорядоченных списков:

- Элемент 1
- Элемент 2
- Элемент 3

- Элемент 1
- Элемент 2
- Элемент 3

Пример упорядоченных списков:

I. Элемент 1
II. Элемент 2
III. Элемент 3

a. Элемент 1
b. Элемент 2
c. Элемент 3

2) Использование изображения в качестве маркера списка с помощью свойства list-style-image.

```
<style>
ul {
  list-style-image: url('sqpurple.gif');
}
</style>
```

CSS Изображение как маркер списка

- Элемент 1
- Элемент 2
- Элемент 3

3) Установка цвета элементов списка с использованием свойства background-color.

```
<style>
ol {
  background: #ff9999;
  padding: 20px;
}

ul {
  background: #3399ff;
  padding: 20px;
}

ol li {
  background: #ffe5e5;
  padding: 5px;
  margin-left: 35px;
}

ul li {
  background: #cce5ff;
  margin: 5px;
}
</style>
```

CSS Стили списка с цветом

1. Элемент 1
2. Элемент 2
3. Элемент 3

- Элемент 1
- Элемент 2
- Элемент 3

5.10. CSS-КНОПКИ

CSS позволяет сделать кнопки выразительными и интерактивными. Рассмотрим примеры стилизации кнопок.

1) Задание цвета кнопок с использованием свойства background-color.

```
<style>
  .button {
    background-color: #4CAF50; /* Зеленый */
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
  }

  .button2 {background-color: #008CBA;} /* Синий */
  .button3 {background-color: #f44336;} /* Красный */
  .button4 {background-color: #e7e7e7; color: black;} /* Серый */
  .button5 {background-color: #555555;} /* Черный */
  </style>
```

CSS Кнопки цветные

Изменение цвета фона кнопки со свойством background-color:

2) Изменение стиля кнопки при наведении мыши с использованием селектора :hover.

```
<style>
.button {
  background-color: #4CAF50; /* Зеленый */
  border: none;
  color: white;
  padding: 16px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
```

```
  -webkit-transition-duration: 0.4s;
  transition-duration: 0.4s;
  cursor: pointer;
}

.button1 {
  background-color: white;
  color: black;
  border: 2px solid #4CAF50;
}

.button1:hover {
  background-color: #4CAF50;
  color: white;
}
</style>
```

CSS Кнопка при наведении

Используйте селектор :hover для изменения стиля кнопки при наведении на нее курсора мыши.

Совет: Используйте свойство transition-duration для определения скорости эффекта "hover":

Зеленый

5.11. CSS-СТИЛЬ ССЫЛКИ

Изменение состояния ссылок при использовании псевдокласса.

Псевдокласс	Когда срабатывает
:hover	При наведении
:active	В момент клика
:visited	После посещения ссылки
:focus	При фокусе с клавиатуры

Пример.

```
<style>
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
```

CSS Разные ссылки

Наведите мышь на ссылки и смотрите на изменение макета

Эта ссылка меняет цвет

Эта ссылка меняет размер шрифта

Эта ссылка меняет семейство шрифтов

Эта ссылка меняет оформление текста

5.12. CSS-МАКЕТ WEB-САЙТА

В процессе написания CSS следует придерживаться некоторых принципов, которые позволяют сократить код CSS, сделать его более удобным, наглядным и читабельным. Читабельность в данном случае означает, что другой разработчик (или вы сами через некоторое время) может легко понять и модифицировать стиль. Рассмотрим основные принципы читаемого CSS-кода.

Размещайте каскадные таблицы стилей в отдельном файле

Размещение стилей в отдельном файле позволяет ускорить загрузку веб-страниц за счет уменьшения их кода, а также кэширования файла с описанием стиля.

Удаляйте неиспользуемые селекторы

Большое количество селекторов создает путаницу в вопросе о том, кто из них за что отвечает, да и просто увеличивает объем документа. Чтобы этого не произошло, удаляйте селекторы, которые никак не применяются на сайте. К сожалению, определить точно, какой селектор используется, а какой нет, довольно сложно, поэтому добавляйте комментарий в код. Это поможет хотя бы не запутаться в большом объеме текста.

Применяйте группирование

Достоинство и удобство группирования состоит в описании одинаковых свойств в одном месте. Тем самым, значение свойства пишется только один раз, а не повторяется многократно.

Используйте универсальные свойства

Вместо того, чтобы указывать значения отступа на каждой стороне элемента через свойства margin-left, margin-right, margin-top и margin-bottom, это можно одновременно задать через универсальное свойство margin. Перечисление значений через пробел позволяет установить индивидуальные отступы для каждой стороны. Кроме margin, к универсальным свойствам относятся background, border, font, padding. Применение этих свойств сокращает объем кода и повышает его читабельность.

Разработка CSS-макета web-сайта состоит из следующей последовательности шагов:

Этап	Действие
Анализ структуры страницы	На этом этапе определяются основные блоки интерфейса: шапка (header), меню навигации, основной контент, sidebar (если нужен), подвал (footer). Создается каркас в HTML – без стилей, только структура. Это называют скелетом страницы

Создание базовой сетки	Когда разметка готова, проектируют схему расположения элементов
Базовые стили и типографика	На этом этапе задаются общие правила оформления, которые будут использоваться по всему проекту. Это своего рода фундамент внешнего вида. Сюда входят: семейство шрифтов, размеры текста, отступы между параграфами, базовые цвета, настройки по умолчанию для ссылок и списков
Работа с отступами и выравниванием	Частая проблема новичков – элементы «липнут» друг к другу или плавают как попало. Решается это с помощью системы отступов (margin, padding) и грамотной иерархии заголовков
Подготовка адаптивности	Макет должен корректно отображаться не только на ноутбуках, но и на телефонах
Оформление блоков и компонентов	Когда базовые стили готовы, переходят к оформлению отдельных частей сайта – компонентов. Это могут быть: карточки товаров, кнопки, формы, таблицы, галереи, выпадающие меню. На этом этапе важно соблюдать единый стиль
Переиспользуемые классы и модульность	Хороший CSS строится по принципу – не повторяйся. Если ты видишь одинаковые стили в нескольких местах, лучше сделать один общий класс
Организация CSS-файлов	Небольшой сайт можно стилизовать одним CSS-файлом. Но если проект растет, он превращается в кашу. Поэтому стили разбивают на логические модули
Оптимизация и проверка макета	Когда макет готов и все выглядит красиво, наступает этап, о котором часто забывают – проверка и оптимизация. Здесь важно убедиться, что стили не только работают, но и правильно написаны. Что проверяем: ✓ нет ли лишних повторяющихся стилей ✓ не остались ли «мертвые» классы, которые нигде не используются ✓ все ли читается на мобильных устройствах ✓ нет ли горизонтальной прокрутки ✓ не разъезжаются ли блоки при масштабировании ✓ все ли шрифты подключены правильно ✓ корректно ли работают hover-эффекты и ссылки

Пример.

```
<!DOCTYPE html>
<html>
<head>
<title>Мой блог</title>
<style>
/* CSS-код из предыдущего примера */
body {
  font-family: sans-serif;
  margin: 0;
  padding: 0;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

header {
  background-color: #333;
  color: white;
  padding: 1rem 0;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  display: flex;
}

nav li {
  margin-left: 2rem;
}
```

```
nav a {
  text-decoration: none;
  color: white;
}
main {
  flex: 1;
  padding: 2rem;
}
.post {
  background-color: #f9f9f9;
  padding: 1rem;
  margin-bottom: 1rem;
  border-radius: 5px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.post h2 {
  margin-top: 0;
}

footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 1rem 0;
}

@media (max-width: 768px) {
  nav ul {
    flex-direction: column;
  }
  nav li {
    margin-left: 0;
    margin-bottom: 1rem;
  }
}
</style>
```

```
</head>
<body>
<header>
  <h1>Мой Блог</h1>
  <nav>
    <ul>
      <li><a href="#">Главная</a></li>
      <li><a href="#">О блоге</a></li>
      <li><a href="#">Контакты</a></li>
    </ul>
  </nav>
</header>

<main>
  <article class="post">
    <h2>Заголовок поста 1</h2>
    <p>Текст первого поста.</p>
  </article>
  <article class="post">
    <h2>Заголовок поста 2</h2>
    <p>Текст второго поста.</p>
  </article>
</main>

<footer>
  <p>&copy; 2023 Мой Блог</p>
</footer>

</body>
</html>
```

Данный код состоит из двух частей: HTML и встроенного CSS.

1) HTML-код

<head> – содержит метаданные страницы, такие как заголовок (`<title>`), внешние ссылки на CSS-файлы (в этом примере CSS встроен непосредственно в `<head>`) и другие метаданные, которые не отображаются непосредственно на странице.

<title>Мой блог</title> – задает заголовок, отображающийся в закладке браузера.

<style>... </style> – здесь находится встроенный CSS-код. В больших проектах обычно CSS помещается в отдельный файл и подключается с помощью <link> тега.

<body> – содержит видимый контент страницы.

<header> – заголовок страницы, содержащий заголовок блога (<h1>) и навигационное меню (<nav>).

<nav> – элемент навигации, содержащий неупорядоченный список (<ul>) ссылок (
<a>) на разные страницы блога.

<main> – основная часть страницы, содержащая посты (<article class="post">).

<article class="post"> – каждый пост представлен как отдельный <article> элемент с классом "post". Класс используется для применения CSS-стилей. Каждый пост содержит заголовок (<h2>) и текст (<p>).

<footer> – подвал страницы, содержащий информацию об авторских правах.

2) CSS-код стилизует элементы HTML.

body – определяет стили для всего тела документа, включая шрифт, отступы, и использование flexbox для создания вертикального макета. Для того чтобы страница занимала всю высоту окна браузера, используется min-height: 100vh.

header, nav, main, footer – определяют стили для соответствующих частей страницы – заголовка, навигации, основного контента и подвала. Используются цвета, отступы и стили для текста.

.post – определяет стили для элементов с классом post, т.е. для каждого поста блога. Здесь указан цвет фона, отступы, тени и т.д.

@media (max-width: 768px) – медиазапрос, который применяется, когда ширина экрана меньше 768 пикселей. Это позволяет адаптировать макет для меньших экранов, в данном случае, делая навигационное меню вертикальным.

ЗАКЛЮЧЕНИЕ

Искусственный интеллект и цифровые технологии все активнее входят во все сферы деятельности человека. Результативность маркетинга и менеджмента во многом зависят от оптимального функционирования цифровых инструментов, их наглядности и эргономичности. Веб-страницы и сайты становятся ключевыми источниками информации для принятия решений, и от их качественного и интуитивно понятного интерфейса зависят скорость и рациональный характер принимаемого решения.

Востребованность продукции деятельности веб-дизайнеров и программистов предопределяет значимость освоения компетенций в области создания сайтов и дизайна цифрового пространства.

Оценка сформированности компетенций разработчика веб-приложений определяется в ходе демонстрационного экзамена. Для выполнения задачи по созданию пользовательских интерфейсов Вам будет необходимо продумать и построить диаграмму прецедентов (вариантов использования) и диаграмму классов, на основе изучения предметной области разработать прототипы пользовательских интерфейсов, и, используя все имеющиеся навыки в дизайне, сверстать все страницы сайта. Целесообразно использовать анимацию для привлечения внимания посетителя к акцентам и основным объектам разрабатываемого Вами приложения. Также будет необходимо разработать базу данных с учетом особенностей предметной области сайта и реализовать механизм внесения в нее записей.

Создавая программный продукт, необходимо учитывать основные пожелания заказчика: чтобы сайт был современным и энергичным, удобным, простым и не менял свои качества при различных разрешениях экрана.

Для подготовки к демонстрационному экзамену и дальнейшей профессиональной деятельности Вам целесообразно изучить материал пособия и самостоятельно выполнить задание, аналогичное будущей проверочной работе.

СПИСОК ЛИТЕРАТУРЫ

1. **Северова, Т. С.** Основы веб-дизайна и проектирования пользовательских интерфейсов : учебное пособие / Т. С. Северова. – М. : МПГУ, 2024. – 120 с.

2. **Проектирование** и разработка интерфейсов пользователя. Основы веб-дизайна : учебное пособие / Ю. Ю. Володина, А. В. Скрыпников, О. С. Никульчева и др. – Воронеж : ВГУИТ, 2024. – 183 с.

3. **Шитов, В. Н.** Проектирование и разработка интерфейсов пользователя : учебное пособие / В. Н. Шитов, К. Е. Успенский. – М. : ООО «Издательство КноРус», 2023. – 296 с.

4. **Баркович, А. А.** Веб-проектирование : учебное пособие / А. А. Барковичч, Т .А. Филимонова. – М. : ООО «Научно-издательский центр «ИНФРА-М», 2025. – 231 с.

5. **Климова, Г. Л.** Основы дизайн-мышления для IT-специалистов : учебное пособие / Г. Л. Климова. – М. : ООО «Издательство КноРус», 2024. – 224 с.

ОГЛАВЛЕНИЕ

Учебное электронное издание

ТОЛСТЫХ Светлана Германовна
ПОПОВ Андрей Иванович

ОСНОВЫ WEB-ДИЗАЙНА

Учебное пособие

Редактирование И. В. Калистратовой
Графический и мультимедийный дизайнер Т. Ю. Зотова
Обложка, упаковка, тиражирование И. В. Калистратовой

ISBN 978-5-8265-2963-8

Подписано к использованию 18.11.2025.
Тираж 50 шт. Заказ № 121

Издательский центр ФГБОУ ВО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106/5,
помещение 2, к. 14
Тел. 8(4752) 63-81-08.
E-mail: izdatelstvo@tstu.ru