**Знакомство с CSS**

|  |
| --- |
| [Введение в CSS](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/1) |
| [CSS-правила](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/2) |
| [Продвинутые селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/3) |
| [Свойства для оформления текста](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/4) |
| [Свойства для задания размеров и отступов](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/5) |
| [Позиционирование элементов](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/6) |
| [Создание сетки страницы](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/7) |
| [Декоративные свойства](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/8) |
| [Каскадность](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/9) |
| [Каскадность. Переопределение стилей](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/10) |
| [Каскадность и приоритеты](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/11) |
| [Каскадность. Коктейль из классов](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/12) |
| [Наследование](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/13) |
| [Ненаследуемые свойства](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/14) |
| [Испытание: макет-прототип](https://npoed.htmlacademy.ru/htmlcss110/course/7/run/15) |

# Введение в CSS [1/15]

В предыдущих разделах вы уже немного познакомились с CSS. В этом и последующих разделах мы разберём его детально.

Напомним, что CSS расшифровывается как «каскадные таблицы стилей». Этот язык отвечает за внешний вид HTML-страницы. Синтаксис языка достаточно прост: он состоит из селекторов и свойств.

С помощью селекторов можно сказать браузеру какие именно элементы мы хотим оформить. Свойства описывают как именно мы хотим оформить эти элементы. То есть селекторы — это снайперский прицел, а свойства — это кисть, рубанок, скальпель и перфоратор.

В разделе «Структура HTML-документа» были описаны различные способы подключения стилей. В этом разделе мы будем пользоваться редактором **CSS**.

# CSS-правила [2/15]

Весь CSS-код состоит из повторяющихся блоков следующего вида:

селектор {

свойство1: значение;

свойство2: значение;

}

Такой блок называется «CSS-правило». Каждое CSS-правило содержит хотя бы один селектор и свойство.

Простейшие селекторы — это селекторы по именам тегов. С их помощью можно задать стили для всех абзацев на странице, для всех ссылок, заголовков первого уровня и так далее. Такие селекторы содержат имя тега без символов < и >. Например:

p {

/\* стили для абзацев \*/

}

h1 {

/\* стили для заголовков \*/

}

# Продвинутые селекторы [3/15]

К более сложным селекторам можно отнести селекторы с использованием классов и псевдоклассов.

Класс позволяет объединять разные элементы в смысловые группы и применять к ним одинаковое оформление. Например, можно создать класс «элементы с ошибкой» и задать ему красный цвет текста. Затем можно добавлять этот класс к любому HTML-тегу: абзацу, заголовку, элементу списка и так далее.

Класс тега можно задать с помощью атрибута class, который содержит имя класса (или имена классов через пробел). Пример:

<p class="help"></p>

<p class="help error"></p>

В примере у первого абзаца задан класс help, у второго абзаца заданы классы help и error.

Селектор с использованием класса задаётся так: .имя\_класса. Например:

.help { ... }

.error { ... }

Более подробно об использовании селекторов, рассказано в разделе «Селекторы».

# Свойства для оформления текста [4/15]

В CSS существует огромное количество свойств. Их можно разбить на следующие группы:

* оформление текста;
* работа с размерами и отступами;
* позиционирование элементов;
* создание сеток;
* декоративные: цвета, фон, тени;
* другие.

В этом задании мы познакомимся с несколькими свойствами для оформления текста, а более подробно эти свойства рассмотрим в разделе «Оформление текста с помощью CSS».

# Свойства для задания размеров и отступов [5/15]

С помощью CSS можно задавать ширину, высоту, внешние и внутренние отступы элементов, минимальную и максимальную ширину и высоту и так далее.

Все свойства, которые так или иначе влияют на размеры и отступы элементов, описываются в так называемой «Блочной модели документа». Помимо свойств для отступов и размеров в блочную

модель входят свойства для описания границ и очень важное свойство display, которое определяет тип элемента (блочный, строчный и другие).

В этом задании мы познакомимся с некоторыми свойствами из блочной модели, а более подробно и полно они будут рассмотрены в разделе «Блочная модель документа».

# Позиционирование элементов [6/15]

С помощью позиционирования можно очень точно и гибко управлять расположением элементов. Позиционирование применяют для создания сложных «многослойных» интерфейсов, таких как всплывающие окна или галереи, а также для вёрстки мелких декоративных элементов.

Главным свойством для работы с позиционированием является position, которое переключает режимы позиционирования элемента. Ещё четыре свойства: top, right, bottom, left, управляют расположением элемента. И последнее свойство z-index управляет порядком слоёв.

В этом задании мы изменим стандартный режим позиционирования элемента на абсолютный и поперемещаем элемент.

Более подробно теория и практика позиционирования элементов будет рассмотрена в разделе «Позиционирование».

# Создание сетки страницы [7/15]

Типичный веб-сайт состоит из шапки, главного меню, блока с основным содержанием, боковых колонок, подвала. Эти блоки могут быть расположены друг под другом, в несколько колонок или ещё сложнее. Такое взаимное расположение основных блоков сайта и называют «сеткой» или «раскладкой».

Сетка может быть фиксированной или тянущейся, когда ширина блоков изменяется в зависимости от ширины браузера. Чтобы быстро и легко создавать сетки, нужно хорошо разбираться в блочной модели и позиционировании.

В этом задании мы создадим простейшую раскладку из двух колонок. А в разделе «Сетки» рассмотрим приёмы создания сеток и потренируемся строить более сложные раскладки страниц.

# Декоративные свойства [8/15]

Чтобы сделать страницу яркой и красивой можно использовать множество свойств: цвет текста, фон, тени, рамки, закругления и многие другие.

Работа с декоративными свойствами будет подробно рассмотрена в последующих разделах, например, в разделе «Фоны» и «Оформление текста с помощью CSS».

В этом задании мы попрактикуемся использовать некоторые декоративные свойства.

# Каскадность [9/15]

Напомним, что CSS расшифровывается как «каскадные таблицы стилей». Почему именно каскадные?

Всё дело в том, что стили для элемента могут быть определены в нескольких местах: внутри одного файла стилей и в разных файлах стилей.

Браузер находит все CSS-правила, затрагивающие данный элемент, а затем комбинирует их и получает итоговый список свойств для этого элемента. Комбинирование свойств производится по чётким правилам, которые опираются на приоритетность и специфичность.

Название «каскадные» появилось из-за описанного механизма комбинирования стилей из разных CSS-правил.

В этом задании мы скомбинируем стили для одного из абзацев.

# Каскадность. Переопределение стилей [10/15]

В предыдущем примере мы с помощью класса добавили ко второму абзацу дополнительные стили. CSS-правило для абзацев и CSS-правило для класса содержали по одному свойству. Когда мы добавили класс ко второму абзацу, в его наборе стилей стало два свойства. Вот так:

p {

padding: 10px;

}

+

.truth {

background-color: #dff0d8;

}

=

стили второго абзаца {

padding: 10px;

background-color: #dff0d8;

}

Свойства в CSS-правилах были разными. А что произойдёт, если в разных CSS-правилах будут совпадающие свойства? Сейчас и проверим.

# Каскадность и приоритеты [11/15]

Когда в предыдущем задании мы задали цвет фона для правила с классом truth, одно из свойств второго абзаца переопределилось:

p {

padding: 10px;

background-color: #dff0d8;

}

+

.truth {

background-color: #aaddff;

}

=

стили второго абзаца {

padding: 10px;

background-color: #aaddff;

}

Когда для одного и того же элемента есть несколько CSS-правил с одинаковыми свойствами, браузер использует понятия приоритетов и специфичности, чтобы выбрать значение свойства из нескольких возможных. Упрощённо, можно сказать что:

1. CSS-правила в значении атрибута style самые приоритетные,
2. за ними идёт селектор с id,
3. затем селектор с классом,
4. затем селектор с именем тега.

На самом деле, механизм определения приоритетов и специфичности более сложный. Подробно он описан в разделе «Наследование и каскадирование».

# Каскадность. Коктейль из классов [12/15]

Похоже, вы решили головоломку! Поздравляем!

Для решения нужно было лишь переместить CSS-правило для класса green ниже CSS-правила для класса blue. Когда к одному элементу применяются несколько CSS-правил, то приоритетность этих правил определяется по их селектору. Если селекторы однотипные, как в нашем случае, то тогда более приоритетным является CSS-правило, которое расположено ниже в коде.

Резюмируем. Одному и тому же элементу можно назначать несколько классов. Благодаря механизму каскадности, CSS-правила этих классов будут комбинироваться, а при конфликте свойств будет применяться механизм приоритетов.

Назначение нескольких классов одному элементу — очень гибкий и мощный приём в арсенале веб-разработчика. Он позволяет упрощать и значительно сокращать CSS-код.

Представьте, что на макете очень много блоков с одинаковым фоном, цветом текста и отступами. Вместо того, чтобы всё время повторять CSS-свойства для этих блоков, можно создать один общий класс и использовать его в HTML-коде. А если понадобится изменить внешний вид этих блоков, то нужно будет исправить всего один класс в CSS.

# Наследование [13/15]

Другой важный механизм CSS — это наследование. Он заключается в том, что часть стилей может передаваться от родительского элемента к дочерним (вложенным в него).

Например, все элементы внутри тега body являются дочерними по отношению к нему. Если для body в стилях задать цвет текста красным, то цвет всех остальных элементов тоже станет красным.

Ещё пример: тег ul является родительским по отношению к вложенным в него тегам li. Если задать для ul шрифт курсивом, то и внутри всех li шрифт станет курсивным.

# Ненаследуемые свойства [14/15]

Наследование работает не для всех свойств. Некоторые свойства применяются только к самому элементу и не переходят к его потомкам.

К таким ненаследуемым свойствам относятся: ширина, высота, отступы, режим позиционирования и другие. Согласитесь, было бы странно задать отступы для body и обнаружить, что у всех вложенных элементов тоже появились отступы.

Более подробно наследование описано в разделе «Наследование и каскадирование».

# Испытание: макет-прототип [15/15]

В этом испытании вам нужно привести в порядок простой прототип макета. Код HTML-редактора заблокирован, поэтому вам придётся использовать только стили.

Чтобы исправить макет, нужно добавить недостающие свойства, а некоторые исправить. Все свойства, которые «испорчены» или «потеряны» в прототипе, вы использовали в этом разделе:

* ширина и внешние отступы,
* цвета фона и текста,
* начертание и толщина шрифта,
* построение сеток.

Подсказки:

* Использованные цвета: #ffffff и #333333.
* Для одного элемента нужно будет изменить размер шрифта, для этого используйте свойство font-size.
* Значения отступов, ширины блоков, а также неизвестных размеров шрифта кратны 10,
* Свойства для позиционирования position, top, left и другие не использовались.