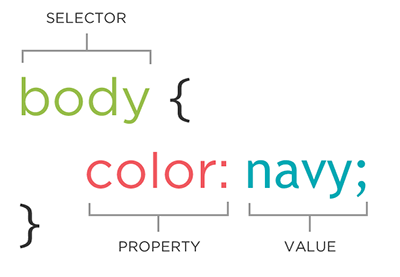
Технология CSS

**CSS (Cascading Style Sheets)** - каскадные таблицы стилей, это язык, используемый для описания оформления документа, написанного языком разметки, таким как HTML.

В то время как HTML используется для определения структуры и семантики содержимого документа, CSS используется для его стилизации и позиционирования. Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Синтаксис CSS

Синтаксис CSS очень простой, объявление стиля состоит из двух частей: элемента веб-страницы — селектора и команды форматирования — блока объявления. Вначале пишется имя селектора, например body, это означает, что все стилевые параметры будут применяться к тегу **body**, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия.



**CSS правило** - это конструкция, которая связывает 1 или более селекторов с одним или более свойств CSS.

**Селектор (selector)** - указывая его, мы говорим браузеру, к чему именно хотим применить css правило (html-тег, id, класс и т.д.).

**Блок объявлений (declaration block)** - так называется всё содержимое, находящееся между фигурными скобками после селектора.

**Свойство (property)** - идентификатор действия, которое будет применено к элементу (например, вы хотите изменить начертание шрифта font-weight, цвет color, размер шрифта font-size и т. д.). После имени свойства необходимо ставить двоеточие.

**Значение (value)** - описывает, как именно свойство будет обработано браузером. Каждое свойство имеет набор допустимых значений, определенных формальными правилами, а также семантический смысл, реализованный движком браузера. Записывается после двоеточия.

Виды таблиц стилей

Встроенные стили

Стили прописываются непосредственно в атрибуте **style** самого тега элемента, к которому мы хотим их применить.

Внутренние стили

Стили прописываются в теге **style**, который размещается в **head**.

Внешняя таблица стилей

Внешняя таблица стилей описывается во внешнем файле с расширением **.css**, после чего он подключается к документу, используя тег **link**. Внутри файла могут содержаться только стили. К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов **link**. На сегодняшний день это стандарт подключения стилей к документу.

* На одном уровне с index.html создается папка css
* Внутри папки css создается файл стилей, к примеру, styles.css
* В теге head, используя тег link, подключаем созданный ранее файл стилей. В атрибут href тега link значением записываем путь к файлу стилей, относительно index.html

<head>

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

</head>

Виды селекторов

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

**Универсальный селектор** - соответствует любому HTML-элементу

**Селектор элемента** - описывает, к какому типу элемента будет применено css правило, после чего свойства будут применены ко всем элемента данного типа.

**Селектор класса** - тегам можно задать специальный атрибут **class**. После чего мы можем использовать селектор класса, чтобы задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы. В CSS перед именем класса ставится точка .

**Селектор идентификатора** - как и с классом, тегам можно задавать атрибут **id**. Отличие в том, что id должен быть уникальным на странице, это значит, что селетор id позволяет форматировать один конкретный элемент. В CSS перед идентификатором ставится решетка #

**Селектор потомка** - применяет стили к элементам, расположенным внутри элемента-контейнера. Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс.

**Дочерний селектор** - дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только, если дочерний элемент идёт сразу за родительским элементом, и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

**Сестринский селектор** - сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня.

**Селектор атрибута** - выбирает элементы на основе имени атрибута или значения атрибута.

**Селектор псевдокласса** - псевдоклассы это состояния элемента. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Группировка селекторов

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо перечислить через запятую нужные селекторы.

Наследование, каскад, специфичность

Наследование и каскад — два важных понятия в CSS, которые тесно связаны. Наследование заключается в том, что элементы наследуют свойства от содержащего их элемента (родителя). Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

Наследование

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как color, font, text-align и т.д. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

* Стили могут наследоваться от родительского элемента.
* Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.
* К одному элементу могут применяться стили из разных источников. Проверить, какие стили применяются, можно в инструментах разработчика.
* При определении стиля можно использовать любую комбинацию селекторов — селектор элемента, псевдокласса элемента, класса или идентификатора элемента.

Свойства, относящиеся к форматированию блоков, не наследуются. Такие как background, border, display, width и т.д.

Каскад

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

* К одному и тому же элементу может примениться несколько CSS-правил. В этом случае свойства правил комбинируются.
* Свойства элемента можно комбинировать и явно с помощью множественных классов.
* Когда в разных CSS-правилах есть одинаковые свойства с разными значениями, то они конфликтуют.

Правило !important

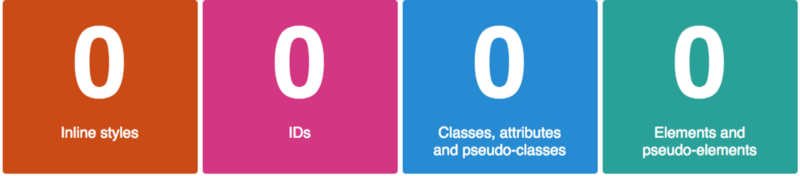
Вес правила можно задать с помощью ключевого слова **!important**, которое добавляется сразу после значения свойства. Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить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

Крайне не рекомендуется использовать **!important** в современной разработке.

Специфичность

Для каждого правила браузер вычисляет специфичность селектора, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: 0, 0, 0, 0. В результате к элементу применятся те правила, специфичность которых больше.

* Специфичность определяется по селектору.
* Если специфичность одинаковая, то побеждает то правило, которое расположено ниже в коде.
* Чем проще селекторы, тем меньше проблем с расчётом специфичности.



Специфичность селектора определяется следующим образом:

* каждый **встроеннй стиль**, добавляет единицу в первый разряд — **1, 0, 0, 0**
* каждый **id** добавляет единицу во второй разряд - **0, 1, 0, 0**
* каждый **class, атрибут, псевдокласс** добавляет единицу в третий разряд - **0, 0, 1, 0**
* каждый **элемент и псевдоэлемент** добавляет единицу в четвертый разряд - **0, 0, 0, 1**

В материалах по ссылкам ниже вы можете более детально ознакомиться с правилами вычисления веса селектора.

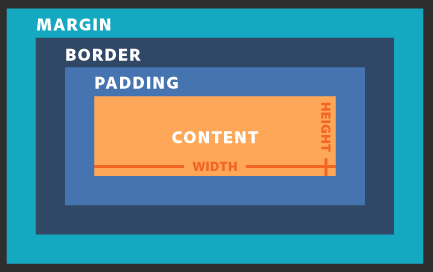
* [Взвешиваем селекторы CSS](https://habrahabr.ru/post/137588/)
* [Why doesn’t my CSS rules apply?, CSS specificity](https://medium.com/@millergonzalez/why-doesnt-my-css-rules-apply-css-specificity-bd07c1118b24)
* [CSS Specificity: What You May Not Have Known](https://webdevstudios.com/2015/05/26/css-specificity-may-not-known/)
* [Онлайн калькулятор специфичности](https://specificity.keegan.st/)

Модель визуального форматирования

Модель визуального форматирования CSS представляет собой алгоритм, который обрабатывает HTML-документ и выводит его на экран устройства. Данная модель преобразует каждый элемент документа таким образом, что он генерирует ноль или более прямоугольных боксов в соответствии с блочной моделью CSS

Блочная модель

В блочной модели элемент рассматривается как прямоугольный контейнер, имеющий область содержимого и необязательные поля, рамки и отступы. Свойство **display** определяет тип контейнера элемента. Для каждого элемента существует значение браузера по умолчанию.



**Область содержимого (content)** — это содержимое элемента, например, текст или изображение.

**Внутреннее поле (padding)** — это расстояние между основным содержимым и его границей (border). Если для элемента задать фон, то он распространится также и на поля элемента. Внутренний отступ не может принимать отрицательных значений.

**Внешний отступ (margin)** - добавляет отступ снаружи элемента от внешней границы рамки (border) до соседних элементов, тем самым разделяя элементы на странице. Внешние отступы всегда остаются прозрачными и через них виден фон родительского элемента. Может принимать отрицательное значение. Не обязателен, по умолчанию значение равно нулю.

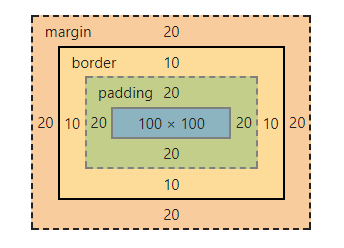
**Граница, или рамка (border)** - если цвет рамки не задан, она принимает цвет основного содержимого элемента, например, текста. Если рамка имеет разрывы, то сквозь них будет проступать фон элемента. Не обязателен, по умолчанию значение равно нулю.

Подвиды блочной модели

Существует несколько блочных моделей. Они отличаются расчетом финального размера блока. Для изменения модели используется свойство **box-sizing**.

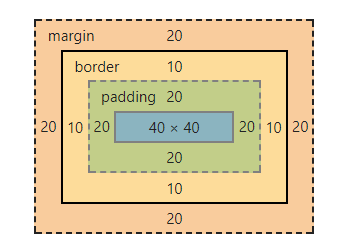
Content box

content-box включена в браузерах по умолчанию, контринтуитивна. Поставив ширину 100px, padding 20px и border 10px, мы получим 100px ширину content area плюс по 20px padding(левый и правый) и по 10px border (левый и правый), итого полная ширина будет равна 100+20+20+10+10 = 160px. Хотя ширину блока мы указали 100px. Аналогично рассчитывается высота.

В результате получится блок cо следующими размерами

Border box

В отличие от content-box, при border-box width и height описывают финальные, желаемые размеры блока, а не контентной области. Контентная область как-бы сжимается чтобы "внутри" блока еще вместился padding и border.

В результате получится блок cо следующими размерами

Блочные и строчные элементы

Блочные элементы

**Блочные элементы (display: block)** — элементы, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя (если для элемента не задано значение width). Это элементы div, h1-h6, p, ol, ul, li и многие другие.

Строчные элементы

**Строчные (display: inline)** - элементы, которые генерируют внутристрочные контейнеры. Они не формируют новые блоки контента.

Они игнорируют верхние и нижние margin и padding, но если для элемента задан фон, он будет распространяться на верхний и нижний padding, заходя на соседние строки текста.

Ширина и высота строчного элемента зависит только от его содержимого, задать размеры с помощью CSS нельзя. Можно увеличить расстояние между соседними элементами по горизонтали с помощью горизонтальных полей и отступов.

Строчные элементы могут содержать только данные и другие строчные элементы. Исключение составляет элемент <a>, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

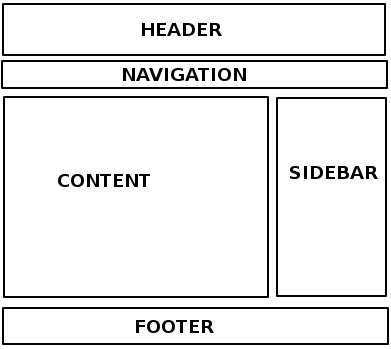
Строчно-блочные элементы

**Cтрочно-блочные (display: inline-block)** - такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.

Для того чтобы верхние и нижние поля и отступы работали для строчного элемента, нужно использовать display: inline-block. Элемент останется строчным, но к нему можно будет полноценно применить поля, отступы, задать ширину и высоту.

Преобразование типов элементов

В некоторых случаях бывает необходимо, чтобы строчный элемент вел себя как блочный и наоборот. Значение свойства display у элемента можно менять по необходимости на любое другое. Можно превратить строчный в блочный или строчно-блочный или наоборот. Это стандартная практика при верстке.

Допустим нам надо сверстать такую структуру документа

У нас все элементы блочные (div), но content и sidebar надо поставить в одну строку, по умолчанию с блочными элементами так сделать не получится. Поэтому мы нашим content и sidebar меняем display на inline-block. После чего задаем им какие-то ширины (в примере ширины в % от общей ширины, для красоты), и вот они уже стоят в одной строке, так как теперь они строчно-блочные и если их общая ширина помещается в одну строку, мы получим желаемый результат.

По [**ссылке**](https://css-tricks.com/fighting-the-space-between-inline-block-elements/)ознакомьтесь с особенностями inline-block элементов, в частности, почему у них всегда есть странный отступ.

Особенности margin

Итак, margin, внешний отступ блока, используется для визуальной реализации вертикальных\горизонтальных зазоров между блоками. Может быть у всех элементов, кроме строчных, которым можно задать только горизонтальный margin.

Схлопывание вертикальных отступов

Одно из самых важных правил, которые нужно запомнить. Соприкасающиеся вертикальные отступы margin объединяются. При этом ширина общего отступа равна ширине большего из исходных отступов. Слияние выполняется только для блочных элементов в нормальном потоке документа.

Дадим верхнему блоку margin-bottom в 30px, нижнему блоку margin-top в 50px, результирующий margin будет не 80px, из-за схлопывания вертикальных отступов, будет выбран больший - 50px.

Выпадение вертикальных отступов

Если внутри одного блока расположить другой блок и задать ему margin-top, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху, т.е. внутренний блок «выпадет» из родительского блока. Если у родительского элемента также был задан верхний отступ, то выберется наибольшее из значений.

В примере мы задали margin-top в 50px для inner-box, а в результате отодвинулся не он сам от верха outer-box, а весь outer-box.

Верхний margin на стыке с блоком-родителем как будто пробивает его и выпадает наружу, это одна из самых частых ошибок при верстке! Рекомендуем быть очень внимательными и использовать верхний и нижний margin для блока-ребенка в крайнем случае, если он вложен в блок-родитель.

Выравнивание контента

В этой секции мы познакомимся с базовыми приемами выравнивания блочного и строчного контента внутри блока-родителя.

Горизонтальное

text-align

Свойство **text-align** используется для выравнивания строки текста или строчных и строчно-блочных элементов внутри блочного элемента по ширине относительно его границ. Применяется только к блочным элементам, например, абзацам. Наследуется.

Возможные значения: **left|right|center|justify|inherit|initial**.

margin

Блочный элемент можно выровнять внутри родителя, задав ему горизонтальный margin. Центрировать можно, указав для **margin-left** и **margin-right** значение **auto** (а также прописав для него конкретный **width**, иначе он займёт всю ширину родительского контейнера, и не будет нуждаться в центрировании). Прижать к правому краю родителя можно задав ему **margin-left: auto**.

Вертикальное

Vertical-align

Применяется только к строчным и строчно-блочным элементам, не выравнивает содержимое блочного элемента, не наследуется. Отлично подходит если стоит задача выровнять элементы в одной строке по вертикали.

В таблице приведены самые часто используемые значния свойства **vertical-align**, полный список значений есть по ссылкам в дополнительных материалах.

|  |  |
| --- | --- |
| **Значение** | **Описание** |
| top | Верхний край элемента совмещается с верхним краем самого высокого элемента в линии. |
| middle | Средняя линия элемента (обычно изображения) совмещается с линией, проходящей через середину родительского элемента. |
| bottom | Нижний край элемента совмещается с нижним краем самого низкого элемента в линии. |
| super | Выравнивает элемент так, как если бы он был надстрочным. |
| sub | Выравнивает элемент так, как если бы он был подстрочным. |

Оформление текста

Частой задачей при верстке является стилизация текстового контента, поэтому рассмотрим полезные свойства для оформления текста и их значения. Они могут быть применены к любому элементу, который содержит текстовый контент.

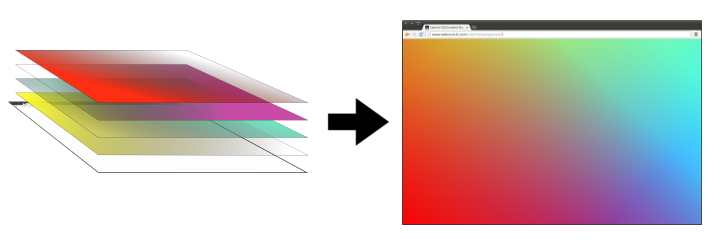
|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| font-size | указывает размер шрифта |
| font-weight | указывает "вес" (жирность начертания) шрифта |
| font-style | указывает стиль начертания шрифта |
| font-family | указывает семейство шрифтов |
| text-decoration | указывает декорирование шрифта |
| color | указывает цвет текста |
| text-indent | указывает отступ первой строки в абзаце |
| line-height | управляет межстрочным интервалом — дополнительным расстоянием между строками над и под текстом. При равных значениях line-height и height выравнивает текст по высоте. |

Фон

Фон в визуальном послании - будь это журнал, картина или веб-страница, всегда преследует две цели. Первая цель - это помощь в распознавании, выделении контента. Если текст светлый, то фон должен быть темным (и наоборот). В противном случае чтение становится трудным или невозможным. Вторая цель использования фона - это помощь в восприятии смысла, подсказка о содержимом контента. Текст с названием компании на фоне пальм и морского пляжа без лишних слов подскажет сферу деятельности и поможет принять решение о том, продолжать ли знакомство с контентом.

В браузере для работы с фоном используется свойство CSS **background** (и ряд других свойств, начинающихся с этого слова). Этих свойств не много, но они отличаются особой моделью поведения и правилами расчета положения на экране.

Фон родителя и потомков. Наследование.

Если элемент находится в потоке документа (а это происходит по умолчанию со всеми элементами), то фон потомка будет над фоном родительского элемента.По умолчанию все элементы имеют белый цвет фона, и свойство **background** не наследуется. Это означает, что если фон отличается от белого, то его всегда нужно задавать.

Фон для элементов, в том числе **body**, связан с их высотой. По умолчанию высота этих элементов **auto**, т.е. определяется контентом. Для того чтобы заполнить экран фоном полностью, и при этом не зависеть от количества контента, потребуется установить для **body** свойство **min-height: 100vh**. Таким образом высота **body** будет минимум по высоте вьюпорта, или больше если того потребует контент.

Существует возможность сделать цвет фона прозрачным или полупрозрачным. Это создает т.н. новый контекст, т.е. вынуждает браузер учитывать дополнительные факторы при рендеринге страницы - цвет фона, border, padding и контента, находящегося под таким фоном. Это усложняет расчеты и требует дополнительных ресурсов устройства, заставляет разработчика тратить больше времени на создание страницы и проверку отображения. Поэтому если есть выбор, то стоит рассмотреть альтернативный вариант верстки.

Свойство background

Это краткая запись, которая допускает описание до 5 различных свойств фона одновременно. Рассмотрим пример этой краткой записи и затем перейдем к знакомству с этими свойствами в полной записи.

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| background-color | задает цвет фону, либо делает фон прозрачным с помощью специального значения **transparent** |
| background-image | позволяет задать фону изображение по url. Через запятую можно задать несколько изображений. Слой image находится над color, т.е. если размер изображения меньше или изображение не загрузится, то будет виден цвет фона, заданный background-color |
| background-repeat | по умолчанию заполняет фон копиями background-image. Возможно запретить повторение или задать направление (горизонталь, вертикаль) для повторения. |
| background-position | задает положение изображения относительно границ блока. В качестве значения этого свойства используют абсолютные(px), относительные единицы(%) и ключевые слова (top, right). |
| background-attachment | определяет поведение фонового изображения при прокрутке содержимого с помощью scroll |
| background-size | определяет размеры фонового изображения в контейнере, использует абсолютные(px), относительные единицы(%) и ключевые слова (cover, contain) |
| background-clip | Изменяет границу распространения фона для box-model. Обрезает фон до заданной области. По умолчанию фон включает content-area, padding и border. Можно уменьшить, вплоть до content-area. |
| background-origin | Изменяет границу распространения фона для box-model. Изменяет размер фона до заданной области. По умолчанию фон включает content-area, padding и border. Можно уменьшить, вплоть до content-area. |

background-repeat

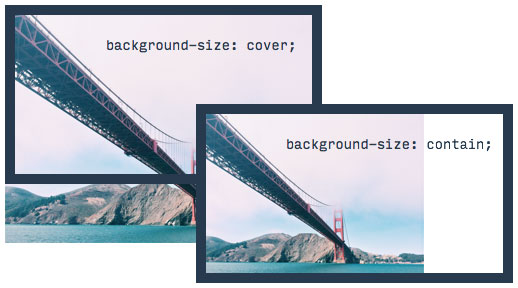
Пример применения значений **no-repeat**, **repeat-x**, **repeat-y** и **repeat** (по умолчанию).

background-position

Имеет два значения, первое описывает горизонтальное положение, а второе вертикальное. Если используется только одно значение, то это означает что второе по умолчанию **center** (или 50%).

Обратите внимание, что в отличие от **box-model** разместить фон по центру блока достаточно легко - достаточно задать 50% по вертикали и горизонтали. Это возможно из-за того, что проценты применяются к точке отсчета, которая в свою очередь также смещается на те же проценты по площади изображения. Легче всего разобраться с этим на примере (точка отсчета - белый круг, а фоновое изображение - красный квадрат).

background-size

У этого правила наибольшее количество значений. Но наиболее важно понимать, как работают два из них - **cover** и **contain**. Оба эти свойства стремятся вместить в контейнер фоновое изображение по максимуму, только одно за счет обрезания лишнего, а другое за счет появления пустого пространства в контейнере - см. рисунок.Можно задавать числовые значения. Если будет задано одно значение, то браузер будет считать его шириной фона и высоту вычислит сам, сохраняя пропорции. Если будет задано два значения - то первое будет как всегда отвечать за горизонталь, т.е. это будет ширина, а второе - высота. И самое интересное, если фону было задано несколько изображений, то также, через запятую, можно задать несколько наборов размеров для этих изображений. Пример.

background-attachment

Пример применения значений **local**, **fixed** и **scroll** (по умолчанию). Различие в поведении проявляется при прокрутке содержимого блока - фон либо привязан к содержимому, либо зафиксирован относительно границ страницы, либо зафиксирован относительно границ блока.

Параллакс-эффект на CSS

Параллакс(Parallax, греч. смена, чередование) – это изменение видимого положения объекта по отношению к удаленному фону в зависимости от нахождения наблюдателя. Первостепенно этот термин употреблялся для природных явлений, в астрономии и геодезии.

В веб-дизайне параллакс-эффект или параллакс-скроллинг – это специальная техника, когда фоновое изображение в перспективе двигается медленнее, чем элементы переднего плана. Эта технология применяется все чаще, так как выглядит действительно эффектно. Достигается такой эффект трехмерного пространства с помощью нескольких слоев, которые накладываются друг на друга и при прокручивании движутся с различной скоростью.

Основной минус параллакса – это проблемы с производительностью сайта. Выглядит все красиво и стильно, но применение JavaScript, с помощью которого и создается эффект параллакса, в значительной степени утяжеляет страницу и очень снижает скорость ее загрузки. Это происходит потому, что в его основе лежат сложные вычисления: JavaScript приходится контролировать положение каждого пикселя на экране.

С появлением CSS3 задача немного упростилась. С его помощью можно создать очень похожий эффект, который будет намного экономичнее в плане ресурсозатрат.

Ближе к концу курса вернитесь и ознакомьтесь со статьей [Pure CSS Parallax Websites](https://keithclark.co.uk/articles/pure-css-parallax-websites/), где в деталях описано создание сложного параллакса на чистом CSS.

Дополнительные материалы

* [Словарь терминов](https://github.com/web-standards-ru/dictionary/blob/master/dictionary.md)
* [Хорошая статья о азах CSS](http://learn.shayhowe.com/html-css/getting-to-know-css/)
* [Словарь имен CSS классов](https://github.com/yoksel/common-words)
* [CSS селекторы](https://code.tutsplus.com/ru/tutorials/the-30-css-selectors-you-must-memorize--net-16048)
* [CSS валидатор](https://jigsaw.w3.org/css-validator/)
* [Учим inline-block](http://learnlayout.com/inline-block.html)
* [Все о vertical-align](http://christopheraue.net/2014/03/05/vertical-align/)
* [CSS альманах](https://css-tricks.com/almanac/)
* [Cheat sheet для Emmet](https://docs.emmet.io/cheat-sheet/)
* [CSS-tricks background-size](https://css-tricks.com/almanac/properties/b/background-size/)
* [MDN background-origin](https://developer.mozilla.org/en-US/docs/Web/CSS/background-origin)