CSS basic

syntax
properties
inheritance
selectors priority
cascading

CSS selectors:
rules
pseudoclasses
pseudo elements
attributes selectors;

Inline, internal, external, import styles.;

Units: em, %, px, pt, vw, vh, vmin, vmax, rem

CSS (Cascading Style Sheets), или каскадные таблицы стилей, используются для описания внешнего вида документа, написанного языком разметки.

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Объявление стиля состоит из двух частей:

элемента веб-страницы — **селектора**

команды форматирования — **блока объявления**.

Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

```
Селектор {
        свойство1: значение1;
        свойство2: значение2;
        свойство3: значение3 значение4;
}
```

Правила записываются внутри фигурных скобок и отделяются друг от друга точкой с запятой.

Между свойствами и их значениями ставится двоеточие.

CSS, как и HTML, игнорирует пробелы. Можно добавлять комментарии, заключая их между /* и */.

http://diana-adrianne.com/

Таблицы стилей

Внешняя

```
<head>
        <link rel="stylesheet" href="css/style.css">
        <link rel="stylesheet" href="css/assets.css">
</head>
```

Встроенные стили

```
<p style="font-weight: bold; color:
red;">Some text.</p>
```

Внутренняя

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

https://codepen.io/lostsou41216364/pen/GXvMVz

@import

Правило @import позволяет импортировать содержимое CSS-файла в текущую стилевую таблицу. @import не разрешается вставлять после любых объявлений кроме @charset или другого @import.

```
@import url("имя файла") [типы носителей];
@import "имя файла" [типы носителей];
```

В качестве типа носителя выступают различные устройства, например, принтер, КПК, монитор и др.

В качестве значения используется путь к стилевому файлу, который указывается внутри необязательной конструкции url(). Путь к файлу при этом можно писать как в кавычках (двойных или одинарных), так и без них.

```
@import url("fineprint.css") print;
@import url("bluish.css") speech;
@import 'custom.css';
@import url("chrome://communicator/skin/");
@import "common.css" screen;
@import url('landscape.css') screen and (orientation:landscape);
```

Тип	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры.
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	Наладонные компьютеры и аналогичные им аппараты.
print	Печатающие устройства вроде принтера.
projection	Проектор.
screen	Экран монитора.
tv	Телевизор.

```
<style>
  @import "/style/main.css" screen; /* Стиль для вывода результата на монитор */
  @import "/style/palm.css" handheld, print; /* Стиль для печати */
</style>
```

Наследование

Когда никакого значения для **свойства**, которое наследуется, у элемента **не установлено**, элемент получает вычисленное значение этого **свойства от его родителя**.

Только корневой элемент документа получает начальное значение из описания свойства.

Типичный пример наследуемого свойства `color`. Стили:

```
p { color: green }
```

https://codepen.io/lostsou41216364/pen/ZMJrgp

разметка:

```
<p>В этом параграфе <em>подчёркнутый текст</em>.</p>
```

слова "подчёркнутый текст" станут зелёными, т.к. тег **em** унаследовал значение свойства `color` от элемента **p,** а не получают начальное значение свойства.

Ненаследуемые свойства

Значения свойства элемента, которое не наследуется, не указано, элемент получает начальное значение этого свойства (как указано в описании свойства).

```
p { border: medium solid }
```

разметка:

```
<p>В этом параграфе <em>подчёркнутый текст</em>.</p>
```

у слов "подчёркнутый текст" не будет рамки (т.к. начальное значение `border-style:none`).

https://codepen.io/lostsou41216364/pen/YOxayJ

Спецификацией CSS **предусмотрено наследование** свойств, относящихся к **текстовому содержимому** страницы, таких как color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space и word-spacing.

Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к **форматированию блоков, не наследуются**.

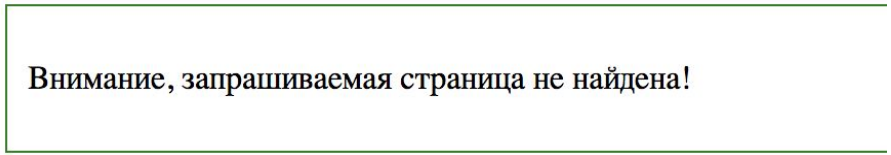
Это background, border, display, float и clear, height и width, margin, min-max-height и -width, outline, overflow, padding, position, text-decoration, vertical-align и z-index.

Принудительное наследование

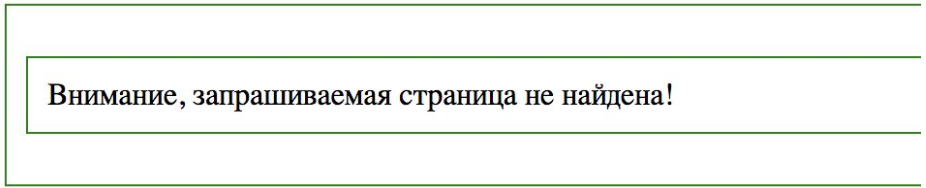
С помощью ключевого слова inherit можно принудить элемент наследовать любое значение свойства родительского элемента.

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

Без inherit



С inherit



https://codepen.io/lostsou41216364/pen/zJdWwp

@charset

Кодировка документа обычно задается в html-документе в теге `<meta charset="...">`. Как для html-документов, так и для таблиц стилей должна использоваться кодировка UTF-8.

```
<!-- HTTP запись -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<!-- краткая запись для HTML5 -->
<meta charset="utf-8">
```

```
@charset "UTF-8";
@charset 'iso-8859-15';
```

Команда @charset применяется для задания кодировки внешнего CSS-файла. Это имеет значение в том случае, если в CSS-файле используются символы национального алфавита.

Не допускается одновременное задание кодировки в таблице стилей с помощью правила `@charset` и на html-странице внутри элемента `<style>`, или как значение атрибута свойства `style`.

Поскольку существует несколько способов задания кодировки таблицы стилей, браузер проверяет эти способы в следующем порядке:

1) Тип кодировки символов, размещенных в начале документа.

2) Значение атрибута `charset` в `Content-Type`, указанное в http-заголовке (или аналогичном), передаваемом веб-сервером.

3) Тип кодировки, указанный в правиле `@charset`.
Если в результате проверки информация о кодировке не была получена, то браузер подразумевает, что документ в кодировке UTF-8.

@font-face

Стандартные шрифты
Arial, **Verdana**, **Times**, **Georgia**, **Courier New**

Правило @font-face позволяет подключать разнообразные пользовательские шрифты. Браузер загружает шрифты в кэш и использует их для оформления текста на странице.

Такой подход называется **встраиванием шрифтов**, а встроенные шрифты — **веб-шрифтами**.

Размещается перед всеми остальными правилами css, улучшает производительность страницы.

Загружаемые шрифты можно помещать в специальную папку `fonts`, созданную на сервере.

Чтобы подключить шрифт с помощью правила `@font-face`, нужно:

1) загрузить файл шрифта на сервер в нескольких форматах для поддержки всеми браузерами,
2) указать название шрифта, прописать ссылку на файл и задать описание шрифта,
3) добавить имя шрифта в свойство `font-family` элемента, который будет отображаться данным шрифтом.

```
@font-face {
    font-family: "WebFont";
    src: url(WebFont.eot?) format("eot"), /* IE8+, знак ? позволяет обойти баг в
         url(WebFont.woff) format("woff"), /* все современные браузеры, IE9+ */
         url(WebFont.ttf) format("truetype"); /* все современные браузеры */
}
p {font-family: "WebFont", Arial, sans-serif; }
```

Форматы веб-шрифтов

Формат WOFF (Web Open Font Format)

Форматы OTF/TTF (OpenType Font и TrueType Font)

Формат EOT (Embedded Open Type)

SVG/SVGZ (Scalabe Vector Graphics)

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.

Сразу рассказать!!! 👻

Вес правила можно задать с помощью ключевого слова !important, которое добавляется сразу после значения свойства, например,

```
span {
        font-weight: bold!important;
}.
```

Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Селекторы

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. Например,

```
* {
        margin: 0;
}
```

обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом:

```
*:after {
        CSS-стили
},
*:checked {
        CSS-стили
}.
```

Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта.

Например,

```
h1 {
        font-family: Lobster, cursive;
}
```

задаст общий стиль форматирования всех заголовков h1.

```
a {
        font-size: 12pt;
        text-decoration: none
}
```

```
table {
        border: black solid 1px
}
```

Первая строчка этого CSS-кода задает всем ссылкам 12-й размер шрифта и убирает подчеркивание. На второй стиле указывается, что у **всех** таблиц граница будет черного цвета, сплошной (solid) и шириной 1 пиксель.

Селектор класса

Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта.

```
<h1 class=«headline">
        Header1
</h1>
```

https://codepen.io/lostsou41216364/pen/bxobWm

Селектор идентификатора

Селектор идентификатора позволяет форматировать **один** конкретный элемент.

Идентификатор id должен быть уникальным и на одной странице может встречаться только один раз.

```
<div id="sidebar"></div>
```

https://codepen.io/lostsou41216364/pen/QVqLqo

Селектор потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера.

Например,

```
ul li {
        text-transform: uppercase;
}
```

выберет все элементы li, являющиеся потомками всех элементов ul.

https://codepen.io/lostsou41216364/pen/ZMXzoG

Дочерний селектор
p > strong — выберет все элементы strong, являющиеся дочерними по отношению к элементу p.

Сестринский селектор
h1 + p — выберет все первые абзацы, идущие непосредственно за любым тегом <h1>, не затрагивая остальные абзацы

h1 ~ p — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него

Селектор атрибута
[атрибут] — все элементы, содержащие указанный атрибут, [alt] — все элементы, для которых задан атрибут alt;

селектор[атрибут]
селектор[атрибут=«значение"]

https://codepen.io/lostsou41216364/pen/mGBbjL

При наведении

https://codepen.io/anon/pen/JxavoW

Динамические псевдоклассы

Псевдокласс в CSS — это ключевое слово, добавленное к селектору, которое определяет его особое состояние. Например, :hover может быть использован для изменения цвета кнопки при наведении курсора на неё.

Выбирают ссылки на странице, которые имеют атрибут `href` и находятся в определенном состоянии, а также некоторые другие элементы:

`:link` — не посещенная ссылка;

`:visited` — посещенная ссылка;

`:focus` — ссылки, а также элементы форм, которые активированы посредством курсора мыши или на которые перешли с помощью клавиатуры (кнопка TAB);

`:hover` — ссылки, а также другие элементы, стили применяются при наведении пользователем на элемент;

`:active` — выбирает элемент, активированный пользователем с помощью клика мышки. Обычно применяется для ссылок, но может отбирать и другие элементы на странице.

Псевдоклассы пользовательского интерфейса

Относятся к элементам форм.
`:disabled` — используется для отбора и стилизации заблокированных для выбора и изменения элементов форм;
`:enabled` — отбирает не заблокированные для выбора и изменения элементы форм;
`:checked` — применяется для выбора и стилизации элементов `<input type="radio">`, `<input type="checkbox">`, а также элементов `<option></option>`, находящихся внутри элемента `<select></select>`;
`:indeterminate` — элементы радио и чекбокс могут быть включены или выключены пользователем. Некоторые из них могут находиться в неопределенном состоянии, к которым и применяется данный псевдокласс.

Структурные псевдоклассы

Представляют концепцию структурных псевдоклассов, которая позволяет отбирать элементы на основании информации, которая отражена в дереве документа и не может быть получена при помощи простых селекторов или их комбинации.

Отсчёт начинается с элемента с индексом 1.

`:nth-child()` — элементы на основе их индекса (в порядке очереди) внутри их родительского контейнера. Варианты:
`li:nth-child(even)` — каждый элемент списка с индексом 2, 4, 6, 8, и т.д.
`li:nth-child(odd)` — каждый элемент списка с индексом 1, 3, 5, 7, и т.д.
`li:nth-child(3)` — только элемент с индексом 3.
`li:nth-child(an+b)` — например, `li:nth-child(3n+1)` выберет первый (3*0 +1 = 1), четвёртый (3*1 +1 = 4), седьмой (3*2 +1 = 7) элементы и т.д., причём значение `b` может быть равно нулю;

https://codepen.io/lostsou41216364/pen/QVqLXL

`:nth-last-child()` — дочерние элементы на основе их индекса внутри контейнера, при этом отсчёт идёт в обратном порядке, т.е. начиная с последнего элемента. Значениями аргумента могут быть положительные, отрицательные значения, а также ключевые слова `even` `odd`;

`:nth-of-type()` — элементы одного типа на основе их индекса внутри контейнера, например, `img:nth-of-type(2n) {float: left;}` установит обтекание по левому краю каждой четной картинке;

`:nth-last-of-type()` — элементы одного типа на основе их индекса внутри контейнера, начиная с последнего элемента к первому;
`:first-child` — элемент, который является первым дочерним элементом некоторого другого элемента;

`:last-child` — последний дочерний элемент элемента-контейнера;

`:first-of-type` — первый элемент данного типа среди дочерних элементов родительского элемента-контейнера;

`:last-of-type` — последний элемент данного типа среди дочерних элементов родительского элемента-контейнера;

`:only-child` — дочерний элемент, который является единственным дочерним элементов. Работает аналогично с `:first-child:last-child` или `:nth-child(1):nth-last-child(1)`, но с меньшей специфичностью;

`:only-of-type` — элемент, который является единственным элементом данного типа в родительском элементе;

`:empty` — элемент, который не содержит ни одного дочернего элемента.

Языковой псевдокласс `:lang()`

Данный псевдокласс используется когда в документе содержатся абзацы текста на разных языках. Чтобы браузер различал их, элементу с текстом добавляется атрибут `lang` с кодом языка, например, `lang="fr"`. В результате чего этот элемент может быть стилизован при помощи селектора `p:lang(fr) {css-стили}`.

https://codepen.io/lostsou41216364/pen/YOrzyj

Если на странице будет динамика, заполнять контент по атрибутам

Псевдокласс отрицания `:not()`

Функциональный селектор псевдокласса, который принимает в качестве значения простой селектор, а затем отбирает элементы, которые не содержат указанное значение аргумента.

Значениями аргументов могут быть только следующие:

селекторы элемента, например, `body :not(strong)`
селекторы класса и идентификатора, например, `p:not(.text)`
селекторы псевдокласса, например, `ul:not(:first-child)`
селекторы атрибута, например, `input:not([type="checkbox"])`

https://codepen.io/lostsou41216364/pen/bxoGeb

Комбинация псевдоклассов

При стилизации элементов возможна комбинация псевдоклассов, например:

`tr:nth-last-child(even):hover` — добавит стили при наведении каждой чётной строке таблицы (отсчёт в обратном порядке);

`h2:not(:first-of-type):not(:last-of-type)` — добавит стили для всех элементов данного типа, кроме первого и последнего элемента данного типа.

Псевдоэлемент

Псевдоэлементы позволяют задать стиль элементов не определённых в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста. Псевдоэлементы начинаются с ::, чтобы отличить их от псевдоклассов.

::after

Псевдоэлемент, который используется для вывода контента после содержимого элемента, к которому он добавляется. Псевдоэлемент ::after работает совместно со свойством content.

По умолчанию ::after создаёт строчный элемент.

```
Селектор::after {
    content: "текст"
}
```

https://codepen.io/lostsou41216364/pen/dqVOmR

::before

псевдоэлемент ::before применяется для отображения контента до содержимого элемента, к которому он добавляется. Работает совместно со свойством content.
По умолчанию ::before создаёт строчный элемент.

```
Селектор::before { content: "текст" }
```

https://codepen.io/lostsou41216364/pen/eLGBPp

::backdrop

Псевдоэлемент отображается ниже самого верхнего элемента в стеке по оси Z, но выше всех остальных элементов на странице, если они имеются.

Обычно ::backdrop применяется для затемнения страницы, чтобы акцентировать внимание на фотографии или диалоговом окне, которые выводятся поверх такого затемнения.

```
Селектор::backdrop { ... }
```

https://codepen.io/lostsou41216364/pen/gdGLEd

::first-letter

Псевдоэлемент ::first-letter определяет стиль первого символа в тексте элемента, к которому добавляется. К этому псевдоэлементу могут применяться только стилевые свойства, связанные со свойствами шрифта, полями, отступами, границами, цветом и фоном.

```
Селектор::first-letter { ... }
```

https://codepen.io/lostsou41216364/pen/KxXNOK

::first-line

Псевдоэлемент ::first-line задаёт стиль первой строки форматированного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т. д. В правилах стиля допустимо использовать только свойства, относящиеся к шрифту, изменению цвета текста и фона.

```
Селектор::first-line { ... }
```

https://codepen.io/lostsou41216364/pen/rZGjNM

::placeholder

Псевдоэлемент, с помощью которого задаётся стилевое оформление подсказывающего текста, созданного атрибутом placeholder.
Допускается использовать свойства для изменения вида текста, например, задать шрифт и цвет.

```
Селектор::placeholder { ... }
```

https://codepen.io/lostsou41216364/pen/XPepbp

::selection

Псевдоэлемент ::selection применяет стиль к выделенному пользователем тексту. В правилах стилей допускается использовать следующие свойства: color, background, background-color, cursor, outline и text-shadow.

```
Селектор::selection { ... }
```

https://codepen.io/lostsou41216364/pen/aaLpZy

Селекторы атрибутов

Селекторы атрибутов отбирают элементы по наличию атрибута или его значению.

[*attr*]
Обозначает элемент с атрибутом по имени attr.

[*attr=value*]
Обозначает элемент с именем атрибута attr и значением в точности сопадающим с value.

[*attr~=value*]
Обозначает элемент с именем атрибута attr значением которого является набор слов разделенных пробелами, одно из которых в точности равно value

[*attr|=value*]
Обозначает элемент с именем атрибута attr. Его значение при этом может быть или в точности равно "value" или может начинаться с "value" со сразу же следующим "-" (U+002D). Это может быть использовано когда язык описывается с подкодом.

[*attr^=value*]
Обозначает элемент с именем атрибута attr значение которого начинается с "value"

[*attr$=value*]
Обозначает элемент с именем атрибута attr чье значение заканчивается на "value"

[*attr*=value*]
Обозначает элемент с именем атрибута attr чье значение содержит по крайней мере одно вхождение строки "value" как подстроки.

```
/* Все span с атрибутом "lang" будут жирными */
span[lang] {font-weight:bold;}

/* Все span в Португалии будут зелеными */
span[lang="pt"] {color:green;}

/* Все span с американским английским будут синими */
span[lang~="en-us"] {color: blue;}

/* Любые span на китайском языке будут красными, как на упрощенном китайском (zh-CN) так и на традиционном (zh-TW) */
span[lang|="zh"] {color: red;}

/* Все внутренние ссылки будут иметь золотой фон */
a[href^="#"] {background-color:gold}

/* Все ссылки с url заканчивающимся на .cn будут красными */
a[href$=".cn"] {color: red;}

/* Все ссылки содержащие "example" в url будут иметь серый фон */
a[href*="example"] {background-color: #CCCCCC;}
```

Специфичность

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

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

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

LI 0,0,0,1 — селектор по тегу

UL LI 0,0,0,2 — селектор с двумя тегами весит больше, чем с одним.

.orange 0,0,1,0 — селектор с классом весит больше, чем селектор с тегом.

.orange A SPAN 0,0,1,2 — селектор перевесит предыдущий, потому что помимо класса содержит два тега.

#page .orange 0,1,1,0 — селектор с ID перевесит всё, кроме inline-стилей.

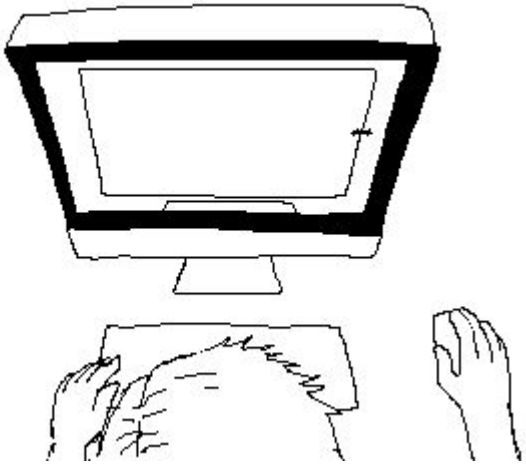
#container A 0,1,0,1

.list A 0,0,1,1

Какая будет специфичность каждого?

```
*                /* a=0 b=0 c=0 -> специфичность =   0 */
li               /* a=0 b=0 c=1 -> специфичность =   1 */
ul li            /* a=0 b=0 c=2 -> специфичность =   2 */
ul ol+li         /* a=0 b=0 c=3 -> специфичность =   3 */
h1 + *[rel=up]   /* a=0 b=1 c=1 -> специфичность =  11 */
ul ol li.red     /* a=0 b=1 c=3 -> специфичность =  13 */
li.red.level     /* a=0 b=2 c=1 -> специфичность =  21 */
#x34y            /* a=1 b=0 c=0 -> специфичность = 100 */
#s12:not(p)      /* a=1 b=0 c=1 -> специфичность = 101 */
```

Единицы измерения



Единицы измерения: "px", "em", "rem" и другие

Пиксель px – это самая базовая, абсолютная и окончательная единица измерения.

Количество пикселей задаётся в настройках разрешения экрана, один px – это как раз один такой пиксель на экране. Все значения браузер в итоге пересчитает в пиксели.

Достоинства

Главное достоинство пикселя – чёткость и понятность

Недостатки

Другие единицы измерения – в некотором смысле «мощнее», они являются относительными и позволяют устанавливать соотношения между различными размерами

Существуют также «производные» от пикселя единицы измерения:
mm, cm, pt и pc, но они давно отправились на **свалку истории**.

1mm (мм) = 3.8px
1cm (см) = 38px
1pt (типографский пункт) = 4/3 px
1pc (типографская пика) = 16px

Так как браузер пересчитывает эти значения в пиксели, то смысла в их употреблении нет.

em

1em – текущий размер шрифта

Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п.

Размеры в em – *относительные*, они определяются по текущему контексту.

https://codepen.io/lostsou41216364/pen/MqEJGp

https://www.w3.org/TR/CSS2/visudet.html

Проценты %, как и em – относительные единицы.

«Процент от чего?»

Как правило, процент будет от значения свойства родителя с тем же названием, но не всегда.

https://codepen.io/lostsou41216364/pen/BOwpvN

px – абсолютные, чёткие, понятные, не зависящие ни от чего.

em – относительно размера шрифта.

% – относительно такого же свойства родителя.

Проценты и em это разные единицы измерения. Для каких-то свойств они ведут себя одинаково (в частности font-size), для каких-то - по-разному. Например, при задании line-height процент высчитывается от размера шрифта текущего элемента, а em от размера шрифта родителя.

В CSS3 были добавлены новые относительные единицы измерения `vh`, `vw`, `vmin`, `vmax`.

Эти единицы вычисляются относительно размеров окна браузера.

Для настольных компьютеров ширина окна браузера больше ширины области просмотра (добавляется ширина скроллбара), поэтому если для элемента установить ширину `100vw`, то он выйдет за пределы html.

Единица	Описание
`vh`	Эквивалентно 1% высоты окна браузера.
`vw`	Эквивалентно 1% ширины окна браузера.
`vmin`	Эквивалентно 1% меньшего размера окна браузера по высоте или ширине.
`vmax`	Эквивалентно 1% большего размера окна браузера по высоте или ширине.

Единицы vh и vw

https://codepen.io/SitePoint/pen/NqEqXd/

https://codepen.io/daspete/pen/zvrzvz

Проценты не лучшее решение для каждого случая, они вычисляются относительно размеров ближайшего родительского элемента.

Использовать высоту и ширину окна браузера - `vh` и `vw`.

Например, если высота окна браузера равна `900px`, то `1vh` будет равен `9px`. Аналогично, если ширина окна браузера равна `1600px`, то `1vw` будет равен `16px`.

Адаптивное полноэкранное фоновое изображение

Ширина элемента, указанная с помощью `100vw` больше ширины области просмотра, то для создания полноэкранных фоновых изображений лучше использовать ширину `100%`, которая будет равна ширине корневого элемента html.

```
.fullscreen-bg {
  background: url(image.jpg);
  background-position: center;
  background-size: cover;
  width: 100%;
  height: 100vh;
}
```

https://codepen.io/lostsou41216364/pen/ZMXeBo

Эффект полностраничного слайда

```
html, body {
  height: 100%;
}
section {
  height: 100%;
}
```

Процентные размеры вычисляются относительно значений родительских элементов

необходимо установить соответствующие значения для каждого элемента DOM.

Единица измерения `vh` не требует установки значений по цепочке, так как её значение вычисляется напрямую относительно окна браузера

```
section {
  height: 100vh;
}
```

Единицы vmin и vmax

`vh` и `vw` всегда вычисляются относительно соответствующих размеров окна браузера

`vmin` и `vmax` определяются минимальным или максимальным значением высоты или ширины.

Например, если ширина окна браузера равна `1200px`, а высота `700px`, то `vmin` будет равен `7px`, а `vmax` — `12px`.