Компьютерные языки разметки
ОИТ
ЭКЗАМЕН

HTML, CSS, JS, XML, HTML-5

Доц. каф. ИСиТ Жиляк Надежда Александровна

311-1

Основы CSS

http://code.mu/books

Введение в CSS.

Синтаксис CSS.

Способы внедрения CSS.

Работа с тегами через CSS.

Селекторы атрибутов.

Основные свойства стилей.

Вложенность и наследование в CSS.

Приоритеты стилей.

Псевдоклассы и псевдоэлементы.

Введение в CSS.

CSS Cascading Style Sheets

Каскадные таблицы стилей.

HTML лишь первый этап в процессе обучения созданию сайтов. Следующим шагом является изучение стилей или CSS . Этот язык отвечает за внешний вид HTML-страницы. Синтаксис языка достаточно прост: он состоит из селекторов и свойств.

Основная идея CSS в том, чтобы отделить дизайн документа от его содержимого. CSS отвечает за оформление и внешний вид, а HTML — за содержание и логическую структуру документа.

Содержимое страницы почти не связано с дизайном её внешнего вида. Изменив всего одну строку в css-стилях, дизайнер сайта может радикально изменить оформление многих тысяч страниц сайта, сделав все заголовки, скажем, зелёными, переместив блок новостей в угол или изменив фон страниц.

Таблицы CSS предлагают логический способ оформления документа. Т.е. в любом правильно составленном тексте можно отделить такие понятия, как заголовки, параграфы, термины, ссылки друг от друга. К тому же CSS охватывает и такие области, как оформление изображений, таблиц и прочих визуальных элементов.

Главные преимущества CSS:

1.Более чистый код

- Этот код легче поддерживать;
- Он быстрее загружается;
- Он лучше оптимизирован для поисковых систем.

2.Модульный код

- Правила стиля могут применяться ко множеству страниц;
- Единообразный дизайн;
- Код легче поддерживать.

3.Точность контроля (позиционирование, размер, поля и др.).

4.Разделение труда

- Задача разработчика — разрабатывать, задача дизайнера — создавать дизайн.

5.Лучшая доступность

- Теги больше не используются не по назначению ;
- Нет необходимости в позиционировании невидимых картинок
- Пользователи могут переписывать стилевые таблицы автора.

Синтаксис CSS.

Стилевые правила записываются в своём формате, отличном от HTML. Основным понятием выступает селектор — это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид:

Все CSS-правила состоят из селектора и блока объявлений (заключённого в фигурные скобки). Внутри блока объявлений (внутри фигурных скобок, проще говоря) может находиться одно или несколько объявлений, разделённых точкой с запятой. Объявление – это строка, составленная из css-свойства и его значения.

Примеры правил CSS:

```css
/* оформляем заголовки: */
h1 {
  color: red;
  background-color: yellow;
  font: Tahoma 2em;
}
/* оформляем абзацы текста: */
p {
  color: grey;
  line-height: 150%;
}
```

Селекторы.

Чтобы применить css-оформление к HTML-элементу или множеству элементов, обычно используются селекторы – специальные указатели на HTML-объекты, к которым мы планируем применить css-правило.

Три основных вида селекторов:

- HTML селекторы;
- Селекторы класса;
- ID селекторы (или идентификаторы).

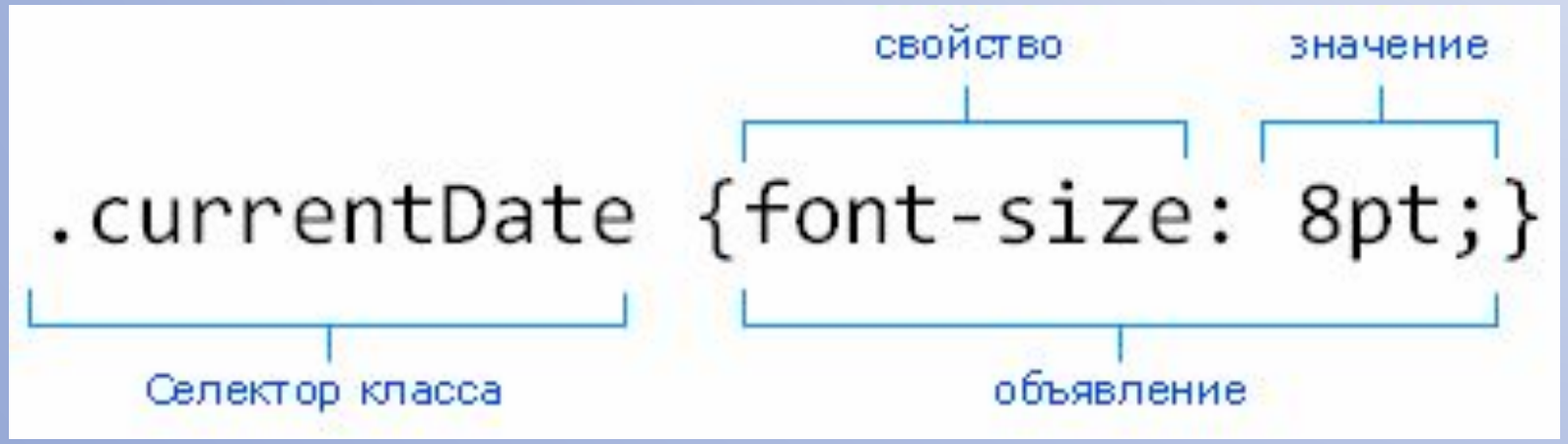
HTML селекторы.

Это простейший случай – в качестве селектора мы используем имя того html-элемента, который хотим изменить.

```
strong {font-weight: normal; color: red;}
h1 { font: bold 10pt verdana; }
```

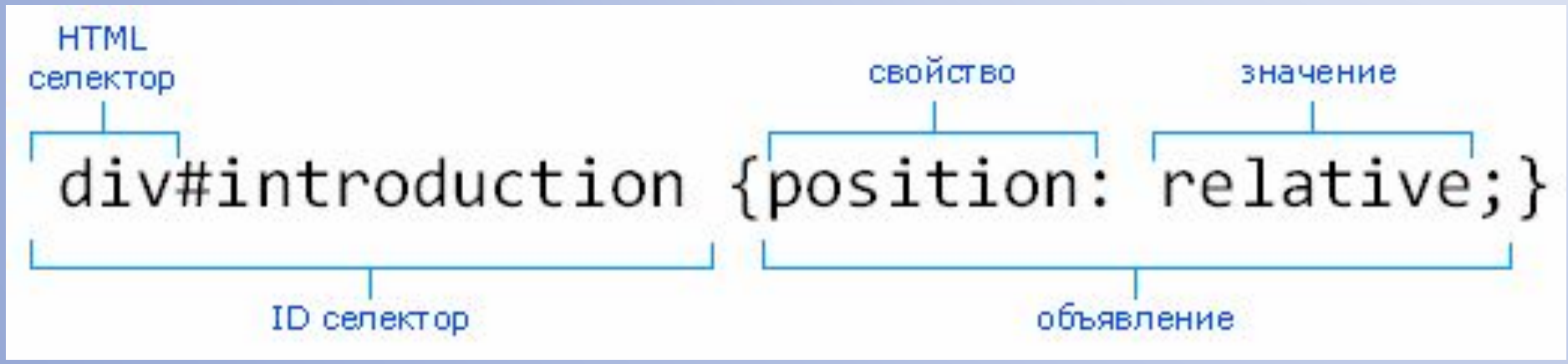
Селекторы класса.

«Класс» - это некое имя, строка, которое мы можем применить к любым HTML-тегам, чтобы впоследствии ссылаться на них по имени класса. В качестве имени класса вы можете использовать практически любую строку. Удобство таких селекторов в том, что можно присвоить одно имя класса множеству html-тегов в документе и затем управлять их внешним видом, обращаясь к ним по имени класса:



ID селекторы (или идентификаторы)

Любой идентификатор (ID) – это некое имя, которое вы, так же, как и в случае с классами, можете применить к любому HTML-тегу. Основное отличие – ID должен быть уникален в рамках html-документа:



CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.
Так, существует две разновидности оформления селекторов и их правил: **расширенная и компактная**.

Расширенная форма записи:

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Компактная форма записи:

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

Если для селектора вначале задаётся свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде установлено ниже.

```
p { color: green; }
p { color: red; }
```

В данном примере для селектора p цвет текста вначале установлен зелёным, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к тексту.

Как применить один стиль к нескольким селекторам.

Это делается просто– достаточно перечислить селекторы через запятую:

Комментарии.

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилевого свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Вместе с тем, добавление текста увеличивает объём документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают.

Содержимое комментариев, браузеры игнорируют.

Чтобы пометить, что текст является комментарием, применяют следующую конструкцию /* ... *

Примеры
:

```
div {
  width: 200px; /* Ширина блока */
  margin: 10px; /* Поля  вокруг элемента */
  float: left; /* Обтекание по правому краю */
}
```

```
/* Это комментарий */
p { margin: 0 1em; }

/* Это более длинный комментарий,
   который занимает более одной строки */

h1 { color: #900; } /* Красный заголовок */
h2 { color: #009; } /* Синий заголовок второго уровня */
/* h3 { padding: 0.5em 0} /* Стиль для заголовков h3 применён не будет,
потому что является частью комменария */
```

Способы внедрения CSS.

Для того, чтобы применить таблицу стилей к HTML-документу, мы можем избрать один из трёх способов, либо комбинировать их:

- Внешнее (в файле);
- Внутреннее (в теге head);
- Строковое (в нужном теге).

Применение внешних стилей (в виде отдельного текстового .css-файла) с помощью элемента link

Применяются с помощью элемента link, который должен располагаться внутри элемента head и нигде более.

```
<link rel="stylesheet" type="text/css" href="mystyle.css" media="all" />
```

Встретив в HTML-документе этот тег, браузер загрузит с сайта CSS-файл (в нашем случае это mystyle.css) и применит к документу содержащиеся в нём стили. Файл не должен содержать ничего, кроме CSS-инструкций.
Внешний файл со стилями удобен тем, что одни и те же стили можно применять ко множеству документов на сайте — в каждом из них достаточно лишь вписать одну строку с элементом link.

Пример

Создадим текстовый файл со следующим содержанием, сохраним как file**.CSS** (сохраняется в папке, в которой находится файл .html, но не с .html расширением, а с **.CSS**).

```
body {
background-color:#0000cc
}
p {
color:#222255
}
.forexample {
color:Orange
}
#ident {
color:#ffffff;
font-weight:bold
}
```

Пример (продолжение):

Сам html документ:

```
<head>
<title>О том, как вставить CSS в HTML страницу</title>
<link rel="stylesheet" type="text/css" href="file.css" />
</head>
<body>
<p>Текст параграфов этого документа темно-синего цвета</p>
<p>Текст</p>
<p class="forexample">А здесь текст оранжевый</p>
<p id="ident">Текст белого цвета, полужирный</p>
<h2 class="forexample">Заголовок</h2>
<p>Текст</p>
</body>
</html>
```

Внутреннее подключение.

Внедрение стилей непосредственно в HTML-документ (в виде блока css-текста) с помощью элемента style.

Называются так потому, что располагаются непосредственно в HTML-документе и применяются только к нему. Иногда называются embedded style sheet (встроенный стиль).
CSS-стили и комментарии располагаются между открывающим и закрывающим тегами элемента style:

```
<style type="text/css">
...
</style>
```

Сам тег style (в отличие от link) может находиться в любой части документа, но обычно его размещают внутри элемента head.

Атрибут со значениями type="text/css" внутри тега <style> сообщает, встроенному в браузер интерпретатору, что применены стилевые описания, то есть CSS.

Синтаксис: первым делом назван селектор (p, body, .forexample, #ident), затем открыта фигурная скобка, прописан атрибут со значением, фигурная скобка закрыта. Атрибуты между собой разделяются точкой с запятой.

Пример

```
<html>
<head>
<title>Подключение CSS файла</title>
<style type="text/css">
p {color:#006633}
.forexample {color:Yellow}
#ident {color:#ffffff; font-weight:bold}
body {background-color:#66cc66}
</style>
</head>
<body>
<p>Текст параграфов этого документа темно-зеленого цвета</p>
<p>Текст</p>
<p class="forexample">А здесь текст желтый</p>
<p id="ident">Текст белого цвета, полужирный</p>
<h2 class="forexample">Заголовок</h2>
<p>Текст</p>
</body>
</html>
```

Строковое подключение

То есть назначение стиля конкретному HTML-элементу непосредственно в документе, с помощью HTML-атрибута style.

Воспользуемся атрибутом style (именно атрибутом элементов, а не элементом!):

```
<p style="color: red">Я абзац, выделенный красным цветом, других таких на сайте нет</p>
```

Атрибут style.

Каждый HTML элемент имеет атрибут style, который сообщает браузеру о том, что к данному элементу будет применено стилевое описание.
Пример использования атрибута style для тега <p>:

```
<p style="color: #086fa1"> Стиль параграфа </p>
```

Внутри атрибута style можно написать несколько CSS объявлений, разделённых точкой с запятой, фигурные скобки не используются.

Примеры :

```
<p style="color:red; font-weight:bold">Текст красного цвета, полужирный</p>
```

```
<p style="font-size:1.3em"> Абзац оформленный с помощью CSS.</p>
```

```
<H1 style="font-size: 120%; font-family: Verdana, Arial, Helvetica, sans-serif; color: #336">
Hello, world!</H1>
```

Работа с тегами через CSS.

Тег <style>

Тег <style> применяется для определения стилей элементов веб-страницы.

Тег <style> необходимо использовать внутри контейнера <head>.

Можно задавать более чем один тег <style>.

Синтаксис:

```
<head>
  <style type="text/css">
   ...
  </style>
</head>
```

Атрибуты:

- media

Определяет устройство вывода, для работы с которым предназначена таблица стилей.

- type

Сообщает браузеру, какой синтаксис использовать, чтобы правильно интерпретировать стили.

Тег

Тег предназначен для определения строчных элементов документа. С помощью тега можно выделить часть информации внутри других тегов и установить для нее свой стиль. Например, внутри абзаца (тега <p>) можно изменить цвет и размер первой буквы, если добавить начальный и конечный тег и определить для него стиль текста. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

Тег <p>

Определяет текстовый абзац. Тег <p> является блочным элементом, всегда начинается с новой строки, абзацы текста идущие друг за другом разделяются между собой отбивкой. Величиной отбивки можно управлять с помощью стилей.

Селекторы тегов.

В качестве селектора может выступать любой тег HTML, для которого определяются правила форматирования, такие как: цвет, фон, размер и др.

```
E { Описание правил стиля }
```

Здесь E имя произвольного тега. Следует понимать, что хотя стиль можно применить к любому тегу, результат будет заметен только для тегов, которые непосредственно отображаются в контейнере <body>.

Пример
.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Селекторы тегов</title>
  <style>
   p {
    text-align: justify; /* Выравнивание по ширине */
    color: green; /* Зелёный цвет текста */
   }
  </style>
 </head>
 <body>
  <p>Более эффективным способом ловли льва в пустыне
  является метод золотого сечения. При его использовании пустыня делится
  на две неравные части, размер которых подчиняется правилу золотого
  сечения.</p>
 </body>
</html>
```

Селекторы атрибутов.

Атрибут - устанавливает стиль для элемента, если задан специфичный атрибут тега. Его значение в данном случае не важно.

```
[атрибут] { Описание правил стиля }
E[атрибут] { Описание правил стиля }
```

Стиль применяется к тем тегам, внутри которых добавлен указанный атрибут. Пробел между именем селектора и квадратными скобками не допускается.

Атрибут фона - background

background-attachment	Как ведет себя фон при прокручивании страницы
background-color	Цвет фона
background-image	Фоновый рисунок
background-position	Положение фонового рисунка
background-repeat	Повторение фона

Форматирование границ (бордюра) элемента – border.

border-collapse	Слияние границ таблицы
border-color	Цвет границ элемента
border-spacing	Расстояние между границами ячеек таблицы
border-style	Вид рамки элемента
border-width	Ширина линии рамки

Шрифт –font.

font-family	Семейство шрифта
font-size	Кегль (размер) шрифта
font-style	Начертание шрифта
font-variant	Малые прописные буквы
font-weight	Насыщенность шрифта

Текст – text.

text-align	Выравнивание по горизонтали
text-decoration	Оформление текста
text-indent	Красная строка
text-transform	Преобразование текста
text-shadow	Добавляет тень к тексту, а также устанавливает её параметры: цвет тени, смещение относительно надписи и радиус размытия.
text-overflow	Определяет параметры видимости текста в блоке, если текст целиком не помещается в заданную область.
text-alingn-last	Задает выравнивание последней строки текста, когда свойство text-align установлено как justify.

Таблица – table.

table-layout	Подгонка ячеек таблицы по высоте и ширине
tab-size	Используется для изменения ширины отступа, заданного с помощью символа табуляции (клавиша Tab).
width	Устанавливает ширину блочных или заменяемых элементов . Ширина не включает толщину границ вокруг элемента, значение отступов и полей.
height	Устанавливает высоту блочных или заменяемых элементов . Высота не включает толщину границ вокруг элемента, значение отступов и полей.

Visibility

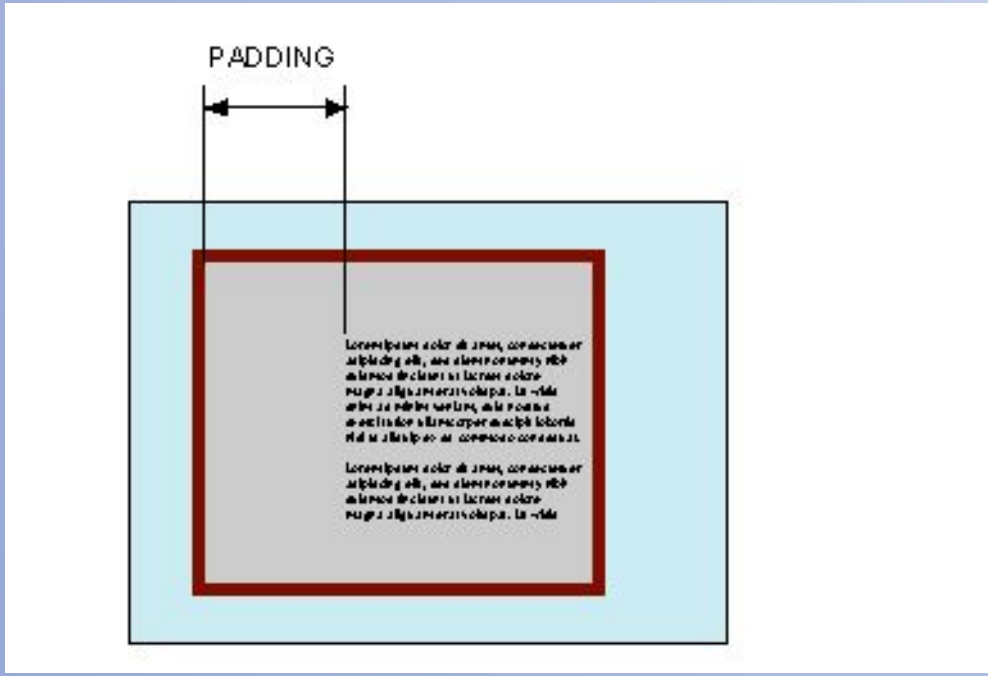
Предназначен для отображения или скрытия элемента, включая рамку вокруг него и фон. При скрытии элемента, хотя он и становится не виден, место, которое элемент занимает, остается за ним. Если предполагается вывод разных элементов в одно и то же место экрана, для обхода этой особенности следует использовать абсолютное позиционирование или воспользоваться свойством display.

```
visibility: visible | hidden | collapse | inherit
```

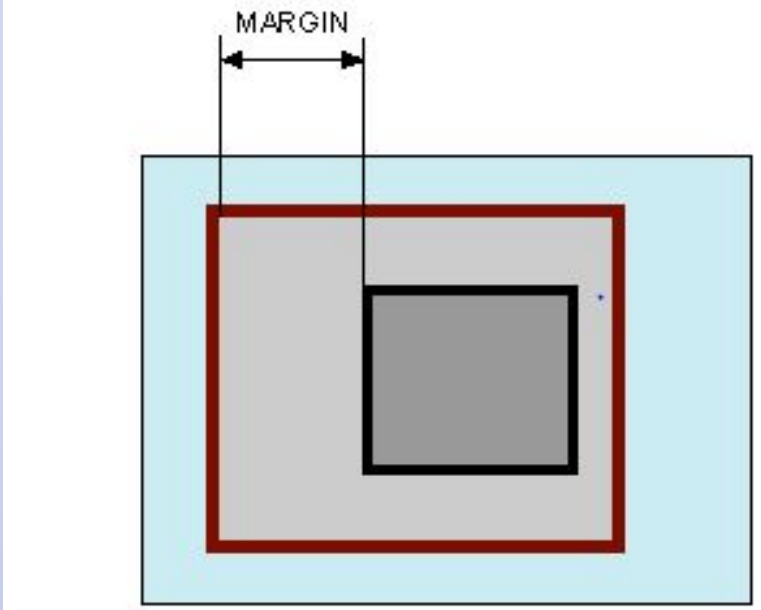
Padding

Устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое .

```
padding: [значение | проценты] {1, 4} | inherit
```



Margin



Устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента.

```
margin: [значение | проценты | auto] {1,4} | inherit
```

Если у элемента нет родителя, отступом будет расстояние от края элемента до края окна браузера с учетом того, что у самого окна по умолчанию тоже установлены отступы. Чтобы от них избавиться, следует устанавливать значение margin для селектора <body> равное нулю.

Свойство margin позволяет задать величину отступа сразу для всех сторон элемента или определить ее только для указанных

Position

Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

```
position: absolute | fixed | relative | static | inherit
```

Значения position:

absolute	Указывает, что элемент абсолютно позиционирован, при этом другие элементы отображаются на веб-странице словно абсолютно позиционированного элемента и нет. Положение элемента задается свойствами left, top, right и bottom, также на положение влияет значение свойства position родительского элемента. Так, если у родителя значение position установлено как static или родителя нет, то отсчет координат ведется от края окна браузера. Если у родителя значение position задано как fixed, relative или absolute, то отсчет координат ведется от края родительского элемента.
fixed	По своему действию это значение близко к absolute, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы.
relative	Положение элемента устанавливается относительно его исходного места. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения.
static	Элементы отображаются как обычно. Использование свойств left, top, right и bottom не приводит к каким-либо результатам.
inherit	Наследует значение родителя.

Font

Универсальное свойство, которое позволяет одновременно задать несколько характеристик шрифта и текста.

```
font: [font-style||font-variant||font-weight] font-size [/line-height] font-family | inherit
```

Значения

В качестве обязательных значений свойства font указывается размер шрифта и его семейство. Остальные значения являются опциональными и задаются при желании.

inherit	Наследует значение родителя.
caption	Шрифт для текста элементов форм вроде кнопок.
icon	Шрифт для текста под иконками.
menu	Шрифт применяемый в меню.
message-box	Шрифт для диалоговых окон.
small-caption	Шрифт для подписей к небольшим элементам управления.
status-bar	Шрифт для строки состояния окон.

Основные свойства стилей.

Свойства шрифта.

font-family - определяет используемый элементом шрифт. Если указать URL(file), то шрифт автоматически установится на компьютер пользователя;
font-style - стиль шрифта (normal, italic);
font-variant - варианты отображения шрифта (normal, small-caps);
font-weight - жирность шрифта (normal, bold, bolder, lighter, значение от 100 до 900);
font-size - размер шрифта (размер, xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger);
font - обобщает вышеперечисленные свойства (любая комбинация из вышеперечисленных значений).

Свойства текста.

word-spacing - расстояние между словами (значение, normal);
text-decoration - декорация текста (none, underline, overline, line-through, blink);
letter-spacing - расстояние между буквами (значение, normal);
vertical-align - позиционирование по отношению к другим элементам стоящим в одном ряду (baseline, sub, super, top-text, top, middle, bottom, bottom-text, %);
text-transform - изменение текста (none, Capitalize, UPPERCASE, lowercase);
text-align - положение текста (left, right, center, justify);
text-indent - отступ (значение, %);
line-height - отступ сверху (normal, значение, %).

Свойства фон и цвет.

color - цвет элемента (значение);
backgroung-color - цвет фона элемента (значение);
background-image - изображение фон (none, URL);
background-repeat - варианты повторения фонового изображения (repeat, repeat-x, repeat-y, no-repeat);
background-attachment - возможность прокрутки фонового изображения (scroll, fixed);
background-position - положение фонового изображения (%ширины, %высоты, top, middle, bottom, left, center, right);
background - обобщает вышеперечисленные свойства (любая комбинация из вышеперечисленных значений).

Свойства блока

margin-top - определяет отступ сверху (значение, %, auto);
margin-right - определяет отступ справа (значение, %, auto);
margin-bottom - определяет отступ снизу (значение, %, auto);
margin-left - определяет отступ слева (значение, %, auto);
margin - обобщает все вышеперечисленные свойства;
padding-top - отступ от верхнего border'а (значение, %);
padding-right - отступ от правого border'а (значение, %);
padding-bottom - отступ от нижнего border'а (значение, %);
padding-left - отступ от левого border'а (значение, %);
padding - обобщает все вышеперечисленные свойства;
border-top-width - толщина верхнего border'а (значение, thin, medium, thick);
border-right-width - толщина правого border'а (значение, thin, medium, thick);
border-bottom-width - толщина нижнего border'а (значение, thin, medium, thick);

border-left-width-толщина левого border'а значение, thin, medium, thick);
border-width - обобщает все вышеперечисленные свойства;
border-color - Цвет border'а. (значение);
border-style - стиль border'ов (none, dotted, dashed, solid, double, groove, ridge, inset, outset);
border-top - обобщает вышеперечисленные свойства для верхнего border'а;
border-right -обобщает вышеперечисленные свойства для правого border'а;
border-left - обобщает вышеперечисленные свойства для левого border'а;
border-bottom - обобщает вышеперечисленные свойства для нижнего border'а;
border - обобщает все вышеперечисленные свойства;
width - ширина элемента (значение, %);
height - высота элемента (значение, %);
float - расположение элемента (left, right, none);
clear - расположение других элементов вокруг данного (left, right,

Классификационные свойства.

display - определяет, как будет отображаться элемент (none, block, inline, list-item);
white-space - определяет, как будут отображаться пробелы между элементами (normal, pre, nowrap);
list-style-type - определяет вид list-item маркера в списках (disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none);
list-style-image - задает вид list-item маркера из картинки (none, URL);
list-style-position - определяет положение маркера в зависимости от list item элемента (inside, outside);
list-style - обобщает вышеперечисленные свойства.

Свойства элемента

position - определяет, как будет отображаться элемент по отношению к другим элементам документа (relative, absolute);
top - определяет позицию элемента TOP относительно элемента родителя (значение, %);
left - определяет позицию элемента LEFT относительно элемента родителя (значение,%);
width - определяет ширину элемента (значение, %, auto);
height - определяет высоту элемента (значение, %, auto);
overflow - режим отображения содержимого элемента, при несоответствии размера элемента, размеру содержимого (non, clip, scroll);
visibility - управление видимостью элемента в документе (hidden, " ").

Вложенность и наследование в CSS.

Вложенность

В хорошо структурированной каскадной таблицей стилей нет необходимости применять множество классов или id селекторов. Это возможно благодаря подробному изложению свойств селекторов внутри других селекторов.

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут вложенные селекторы. Например, задать стиль для тега <b> только когда он располагается внутри контейнера <p>. Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

Синтаксис:

```
E F { Описание правил стиля }
```

Здесь E это родительский тег, а F — дочерний тег, расположенный в контейнере <E>.

В этом случае стиль будет применяться к тегу <F>, когда соблюдается следующий код <E><F></F></E>.

Не обязательно должно быть два тега, допускается произвольный уровень вложения. При этом конструкция может записываться так: div div ul li {...}.

Пример :

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Селекторы</title>
  <style>
   p b {
    font-family: Times, serif; /* Семейство шрифта */
    font-weight: bold; /* Жирное начертание */
    color: navy; /* Синий цвет текста */
   }
  </style>
 </head>
 <body>
  <div><b>Жирное начертание текста</b></div>
  <p><b>Одновременно жирное начертание текста
  и выделенное цветом</b></p>
 </body>
</html>
```

В примере ниже, один стиль указан для всех элементов p, другой стиль указан для всех элементов с class="marked", и третий стиль указан только для элементов p с class="marked":

```
p
{
color:blue;
text-align:center;
}
.marked
{
background-color:red;
}
.marked p
{
color:white;
}
```

Наследование.

Наследованием называется перенос правил форматирования для элементов, находящихся внутри других.
Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются.

Например, все элементы, расположенные внутри элемента <body>, являются его дочерними элементами и потомками. Если в стиле для <body> задать с помощью CSS свойства color красный цвет текста, то цвет текста всех его дочерних элементов и потомков

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Название документа</title>
    <style>
      body { color: red; }
    </style>
  </head>
  <body>

    <h1>Цвет текста заголовка красный</h1>
    <p>Цвет текста абзаца тоже красный.</p>

  </body>
</html>
```

Наиболее удачным примером, на котором можно наглядно исследовать нюансы наследования CSS, является, на мой скромный взгляд, таблица html, которая создается с помощью тегов table, tr и td.

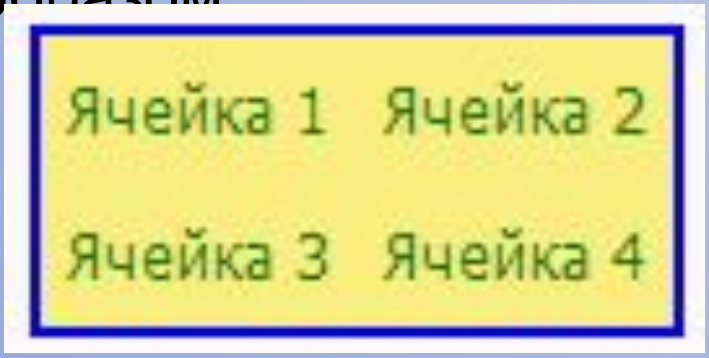
Допустим, заданы свойства оформления для тега table:

```
table {
    color: green; /* Цвет текста */
    background: #faee81; /* Песочный цвет фона таблицы */
    border: blue solid 2px; /* Рамка вокруг таблицы */
    padding: 5px; /* Поля вокруг текста */
}
```

Теперь составим простенькую таблицу из 4 ячеек:

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

На вебстранице она будет выглядеть следующим образом:



Для данной таблицы установлен зеленый цвет текста, поэтому в ячейках слова приняли этот оттенок. Это следствие того, что дочерний элемент тег td наследует свойства своего родителя тега table. Но нужно понимать, что не все стилевые свойства подвержены наследованию.

Например, border определяет рамку вокруг таблицы, но не вокруг ячеек, поэтому эти ячейки не выделены рамкой внутри таблицы. Также не наследуется свойство background. Однако, в этом случае возникает вопрос: почему же цвет фона ячеек приобрел песочный цвет, который указан в качестве значения родительского тега table, если он не наследуется?

Здесь все дело в том, что у свойства background в качестве значения по умолчанию для тега td выступает transparent, то есть прозрачность. Таким образом, цвет фона родительского элемента “просматривается” сквозь фон дочернего элемента, который является прозрачным. Отмечу, что во многих случаях для большинства свойств CSS предусмотрены значения по умолчанию. Поэтому, если для какого-то свойства явно не заданы параметры, до вступает в силу предусмотренное значение по умолчанию.

Наследование позволяет определять значения один раз, задавая их для родительского элемента верхнего уровня.

Приоритеты стилей.

Ниже приведены приоритеты браузеров, которыми они руководствуются при обработке стилевых правил. Чем выше в списке находится пункт, тем ниже его приоритет, и наоборот.

- Стиль браузера.
- Стиль автора.
- Стиль пользователя.
- Стиль автора с добавлением !important.
- Стиль пользователя с добавлением !important.

Самым низким приоритетом обладает стиль браузера — оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

!important

Ключевое слово !important играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей. Если возникает противоречие, когда стиль автора страницы и пользователя для одного и того же элемента не совпадает, то !important позволяет повысить приоритет стиля или его важность, иными словами.

Синтаксис применения !important следующий.

```
Свойство: значение !important
```

При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же селектору, браузер руководствуется следующим алгоритмом.

!important добавлен в авторский стиль — будет применяться стиль автора.
!important добавлен в пользовательский стиль — будет применяться стиль пользователя.
!important нет как в авторском стиле, так и стиле пользователя — будет применяться стиль пользователя.
!important содержится в авторском стиле и стиле пользователя — будет применяться стиль пользователя.

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Специфичность

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше. Специфичность это некоторая условная величина, вычисляемая следующим образом.

За каждый идентификатор (в дальнейшем будем обозначать их количество через a) начисляется 100,

за каждый класс и псевдокласс (b) начисляется 10,

за каждый селектор тега и псевдоэлемент (c) начисляется 1. Складывая указанные значения в определённом порядке, получим значение специфичности для данного селектора.

Пример

```
*               {} /* a=0 b=0 c=0 -> специфичность = 0   */
li              {} /* a=0 b=0 c=1 -> специфичность = 1   */
li:first-line   {} /* a=0 b=0 c=2 -> специфичность = 2   */
ul li           {} /* a=0 b=0 c=2 -> специфичность = 2   */
ul ol+li        {} /* a=0 b=0 c=3 -> специфичность = 3   */
ul li.red       {} /* a=0 b=1 c=2 -> специфичность = 12  */
li.red.level    {} /* a=0 b=2 c=1 -> специфичность = 21  */
#t34            {} /* a=1 b=0 c=0 -> специфичность = 100 */
#content #wrap  {} /* a=2 b=0 c=0 -> специфичность = 200 */
```

Встроенный стиль, добавляемый к тегу через атрибут style, имеет специфичность 1000, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление !important перекрывает в том числе и встроенные стили.

Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что указан в коде ниже.

В примере показано, как влияет специфичность на стиль элементов списка.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Список</title>
  <style>
   #menu ul li {
    color: green;
   }
   .two {
    color: red;
   }
  </style>
 </head>
 <body>
  <div id="menu">
   <ul>
    <li>Первый</li>
    <li class="two">Второй</li>
    <li>Третий</li>
   </ul>
  </div>
 </body>
</html>
```

В данном примере цвет текста списка задан зелёным, а второй пункт списка с помощью класса two выделен красным цветом. Вычисляем специфичность селектора #menu ul li — один идентификатор (100) и два тега (2) в сумме дают значение 102, а селектор .two будет иметь значение специфичности 10, что явно меньше. Поэтому текст окрашиваться красным цветом не будет. Чтобы исправить ситуацию, необходимо либо понизить специфичность первого селектора, либо повысить

```
/* Понижаем специфичность первого селектора */
 ul li {...} /* Убираем идентификатор */
 .two  {...}

/* Повышаем специфичность второго селектора */
 #menu ul li {...}
 #menu  .two {...} /* Добавляем  идентификатор */

 #menu ul li {...}
 .two { color:  red !important; } /* Добавляем !important */
```

Добавление идентификатора используется не только для изменения специфичности селектора, но и для применения стиля только к указанному списку.
Поэтому понижение специфичности за счёт убирания идентификатора применяется редко, в основном, повышается специфичность нужного селектора.

Псевдоклассы и псевдоэлементы.

Псевдоэлементы.

Псевдоэлементы позволяют задать стиль элементов не определённых в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

Синтаксис использования псевдоэлементов следующий:

```
Селектор:Псевдоэлемент { Описание правил стиля }
```

Вначале следует имя селектора, затем пишется двоеточие, после которого идёт имя псевдоэлемента. Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности.

Пример

.

```
.foo:first-letter { color: red }
.foo:first-line {font-style: italic}
```

Псевдоэлементы не могут применяться к внутренним стилям, только к таблице связанных или глобальных стилей.

:after

Применяется для вставки назначенного контента после содержимого элемента. Этот псевдоэлемент работает совместно со стилевым свойством content, которое определяет содержимое для вставки. В примере показано использование псевдоэлемента :after для добавления текста в конец абзаца.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   P.new:after {
    content: " - Новьё!"; /* Добавляем после текста абзаца */
   }
  </style>
 </head>
 <body>
  <p class="new">Ловля льва в пустыне с помощью метода золотого сечения.</p>
  <p>Метод ловли льва простым перебором.</p>
 </body>
</html>
```

Результат:

Ловля льва в пустыне с помощью метода золотого сечения. - Новьё!

Метод ловли льва простым перебором.

:before

По своему действию :before аналогичен псевдоэлементу :after, но вставляет контент до содержимого элемента. В примере показано добавление маркеров своего типа к элементам списка посредством скрытия стандартных маркеров и применения псевдоэлемента :before.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   UL {
    padding-left: 0; /* Убираем отступ слева */
    list-style-type: none; /* Прячем маркеры списка */
   }
   LI:before {
    content: "\20aa "; /* Добавляем перед элементом списка символ в юникоде */
   }
  </style>
 </head>
 <body>
  <ul>
   <li>Чебурашка</li>
   <li>Крокодил Гена</li>
   <li>Шапокляк</li>
   <li>Крыса Лариса</li>
  </ul>
 </body>
</html>
```

Результат:

- Чебурашка
- Крокодил Гена
- Шапокляк
- Крыса Лариса

:first-letter

Определяет стиль первого символа в тексте элемента, к которому добавляется. Это позволяет создавать в тексте буквицу и выступающий инициал.

Рассмотрим пример создания выступающего инициала. Для этого требуется добавить к селектору P псевдоэлемент :first-letter и установить желаемый стиль инициала. В частности, увеличить размер текста и поменять цвет текста (пример).

Пример

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   P {
    font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта основного текста */
    font-size: 90%; /* Размер шрифта */
    color: black; /* Черный цвет текста */
   }
   P:first-letter {
    font-family: 'Times New Roman', Times, serif; /* Гарнитура шрифта первой буквы */
    font-size: 200%; /* Размер шрифта первого символа */
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
    <p>Луч фонарика высветил старые скрипучие ступени, по которым не далее
    как пять минут назад в дом поднялся Паша. Оля осторожно приоткрыла дверь
    и посветила внутрь дома. Луч света, словно нехотя, пробивался сквозь тугую
    завесу из мрака и пыли. </p>
    <p>Взгляд Оли опустился на пол, и она вскрикнула. В пустом помещении никого
    не было, и лишь на полу валялась порванная туфля Паши.</p>
 </body>
</html>
```

Результат:

Луч фонарика высветил старые скрипучие ступени, по которым не далее как пять минут назад в дом поднялся Паша. Оля осторожно приоткрыла дверь и посветила внутрь дома. Луч света, словно нехотя, пробивался сквозь тугую завесу из мрака и пыли.

Взгляд Оли опустился на пол, и она вскрикнула. В пустом помещении никого не было, и лишь на полу валялась порванная туфля Паши.

:first-line

Определяет стиль первой строки блочного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т.д.

Пример
.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоэлементы</title>
  <style>
   P:first-line {
    color: red; /* Красный цвет текста */
    font-style: italic; /* Курсивное начертание */
   }
  </style>
 </head>
 <body>
  <p>Интересно, а существует ли способ действительно практичного применения
  свойства first-line? Нет, не такого, чтобы можно было бы показать, что это
  возможно, а чтобы воистину захватило дух от красоты решения, загорелись глаза от
  скрытых перспектив, после чего остается только сказать себе, что вот это вот, это
  самое сделать по-другому, также изящно и эффектно просто невозможно.</p>
 </body>
</html>
```

Результат:

Интересно, а существует ли способ действительно практичного применения свойства first-line? Нет, не такого, чтобы можно было бы показать, что это возможно, а чтобы воистину захватило дух от красоты решения, загорелись глаза от скрытых перспектив, после чего остается только сказать себе, что вот это вот, это самое сделать по-другому, также изящно и эффектно просто невозможно.

Псевдоклассы.

Псевдокласс :invalid
Применяется к полям формы, содержимое которых не соответствует указанному типу.
Псевдокласс :read-only
Применяется к полям формы, у которых задан атрибут readonly.
Псевдокласс :active
Определяет стиль активной ссылки.
Псевдокласс :default
Применяет стиль к элементам форм, которые установлены по умолчанию в группе похожих элементов.
Псевдокласс :disabled
Применяет стиль к заблокированным элементам форм.
Псевдокласс :empty
Представляет пустые элементы, т.е. те, которые не содержат дочерних элементов, текста или пробелов.
Псевдокласс :enabled
Используется для применения стиля к доступным (не заблокированным) элементам форм.
Псевдокласс :first-child
Применяет стилевое оформление к первому дочернему элементу своего родителя.

Псевдокласс :first-of-type
Задает правила стилей для первого элемента в списке дочерних элементов своего родителя.
Псевдокласс :hover
Определяет стиль элемента при наведении на него курсора мыши, но при этом элемент еще не активирован.
Псевдокласс :lang
Определяет язык, который используется в документе или его фрагменте.
Псевдокласс :last-child
Задает стилевое оформление последнего элемента своего родителя.
Псевдокласс :last-of-type
Задает правила стилей для последнего элемента в списке дочерних элементов своего родителя.
Псевдокласс :link
Применяется к ссылкам, которые еще не посещались пользователем.
Псевдокласс :not
Задает правила стилей для элементов, которые не содержат указанный селектор.
Псевдокласс :nth-child
Используется для добавления стиля к элементам на основе нумерации в дереве элементов.
Псевдокласс :nth-last-child
Используется для добавления стиля к элементам на основе нумерации в дереве элементов.

Псевдокласс :nth-last-of-type
Используется для добавления стиля к элементам указанного типа на основе нумерации в дереве элементов.
Псевдокласс :nth-of-type
Используется для добавления стиля к элементам указанного типа на основе нумерации в дереве элементов.
Псевдокласс :only-child
Применяется к дочерним элементам, только если он единственный у родителя.
Псевдокласс :only-of-type
Применяется к дочерним элементам указанного типа, только если он единственный у родителя.
Псевдокласс :read-write
Применяется к полям формы, доступных для изменения.
Псевдокласс :root
Определяет корневой элемент документа. В HTML этот селектор всегда соответствует элементу <html>.
Псевдокласс :target
Применяется к целевому элементу, иными словами, к идентификатору, который указан в адресной строке браузера.
Псевдокласс :visited
Применяется к ссылкам, уже посещённым пользователем, и задает для них стилевое оформление.

Пример
.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>link</title>
  <style>
   a:link {
    color: #0000d0; /* Цвет ссылок */
   }
   a:visited {
    color: #900060; /* Цвет посещенных ссылок */
   }
  </style>
 </head>
 <body>
   <p><a href="task2.html">Выяснить вес нестандартного груза (камень)</a></p>
   <p>Дается: стул, палка, веревка, безмен с ограничением по весу.</p>
 </body>
</html>
```

Результат:

Выяснить вес нестандартного груза (камень)

Дается: стул, палка, веревка, безмен с ограничением по весу.

Пример:

```
<html>
 <head>
  <meta charset="utf-8">
  <title>last-child</title>
  <style>
   .block {
    background: #7da7d9; /* Цвет фона */
    color: #fff; /* Цвет текста */
   }
   .block :first-child {
    padding: 10px; /*  Поля вокруг текста */
   }
   .block :last-child {
    background: #dda458 url(images/line.png) repeat-x; /* Параметры фона */
    height: 5px; /* Высота блока */
   }
  </style>
 </head>
 <body>
  <div class="block">
   <div>
    При истечении возможности понимания вышеизложенной информации вы
    имеете объективную возможность подать официальный запрос главному
    субординатору локальной виртуальной вселенной.
   </div>
   <div></div>
  </div>
 </body>
</html>
```

Результат:

При истечении возможности понимания вышеизложенной информации вы имеете объективную возможность подать официальный запрос главному субординатору локальной виртуальной вселенной.

Спасибо за внимание!