Оглавление

Глава 2

Язык CSS

2.1. История появления CSS

Как вы знаете из предыдущей лекции, HTML — это язык разметки. То есть язык, который помогает ориентироваться браузеру в том, чем является данный текст.
Например, если поместить текст внутрь тега h1, он отобразится большим. Если поместить текст внутрь тега h2 — то чуть поменьше, и так далее. Но выразительности тегов стало не хватать для тех, кто разрабатывал сайты.
Пользователи хотели изменить цвет текста, или изменить цвет каждой второй буквы текста, или изменить шрифт текста, или же поместить текст в красную рамку. Для решения этой задачи в HTML был введен тег font.

```
<font size="+3" face="Helvetica" color="red">ПРЕВЕД</font>
```

Более показателен пример типичной разметки с использованием тега font:

```
<body bgcolor="#f1f1f1">
    <h1 align="center"><font size="16px" color="red"
     ↪  face="Tahoma">Основной заголовок</font></h1>
    <p><font size="8px" color="gray" face="Arial">Текст параграфа
     ↪  1</font></p>
    <h2><font size="10px" face="Tahoma">Подзаголовок 1</font></h2>
    <p><font size="8px" color="gray" face="Arial">Текст параграфа
     ↪  2</font></p>
    <h2><font size="10px" face="Tahoma">Подзаголовок 2</font></h2>
    <p><font size="8px" color="gray" face="Arial">Текст параграфа
     ↪  3</font></p>
</body>
```

Внутрь каждого элемента, внутрь каждого параграфа, нужно вставлять тег font с заданными атрибутами. Получилось, что разметка-то есть, но она используется не по назначению.
В 1994 году Хокон Виум Ли, автор и создатель браузера Opera, и Берт Бос сформулировали эту проблему и назвали решение проблемы CSS.
Причины появления CSS:

- Избыточность HTML. HTML стал нести слишком много информации, кроме как про разметку документа.
- Желание большего числа возможностей для оформления. Это еще больше раздувало HTML разметку.
- Упрощение оформление страницы.
- Желание избежать копирования одних и тех же тегов и атрибутов для создания одинакового оформления.
- Отделение контента страницы от его представления.

С появлением CSS упомянутый выше код можно переписать в следующем виде:

```
body {
    background: #f1f1f1;
}
h1 {
    font-size: 16px;
    color: red;
    font-family: Tahoma;
    text-align: center;
}
h2 {
    font-size: 10px;
    font-family: Tahoma;
}
p {
    font-size: 8px;
    color: gray;
    font-family: Arial;
}
```

```
<body>
    <h1>Основной заголовок</h1>
    <p>Текст параграфа 1</p>
    <h2>Подзаголовок 1</h2>
    <p>Текст параграфа 2</p>
    <h2>Подзаголовок 2</h2>
    <p>Текст параграфа 3</p>
</body>
```

На первый взгляд, кода стало еще больше. Но основное преимущество состоит в том, что мы можем добавить произвольное число параграфов без необходимости указывать стили оформления для каждого из них по отдельности.
Тем более не будет проблемы поменять сразу для всех параграфов цвет с одного на другой.
CSS — язык стилей, определяющий отображение HTML (и других) документов.
CSS работает со шрифтами, цветом, полями, строками, высотой, шириной, фоновыми изображениями, позиционированием элементов и многими другими вещами.
Основные этапы развития:

- CSS уровень 1 (1996, 1999) – параметры шрифтов, цвета, ...
- CSS уровень 2 (12 мая 1998) – блочная вёрстка, селекторы, ...

CSS уровней 1 и 2 рассматривали разные аспекты стилизации документов. К сожалению, после второй версии спецификации развитие CSS застыло на долгие года. Причиной стал известный верстальщикам браузер — Internet Explorer. Проблема состояла в том, что IE6 поставлялся вместе с операционной системой Windows XP и он не обновлялся автоматически. Получилась ситуация при которой пользователи были привязаны к конкретному браузеру, а веб-разработчики — внедрять современные стандарты CSS.
В конце 2000 годов первыми, кто начал внедрять новые стандарты, стал браузер Firefox от компании Mozilla. Позже к нему присоединился браузер Chrome от Google и браузер Opera. Это дало толчок для развития новых спецификаций.

- CSS уровень 2.1 (07 июня 2011)

Работа над спецификацией 2.1 велась более чем 10 лет, потому что браузеры не стремились активно применять эту спецификацию.

Начиная с этого времени размер спецификации стал слишком большой. И его сложно было поддерживать в виде единого документа. Логично было разнести эту спецификацию на несколько разных документов, каждый из которых затрагивает отдельный аспект стилизации элементов. Те спецификации, которые были развитием тех или иных аспектов, которые уже были затронуты в спецификации уровней 1 или 2 получили номер 3. Если спецификация новая и не была сформулирована ранее, ее нумерация начинается с 1.

- CSS уровней 3 находится в стадии разработки – трансформации, анимация, ...
- CSS уровней 4 разрабатывается с 29 сентября 2011 года.

2.2. Синтаксис CSS

Каждое правило CSS состоит из двух частей: селектора и блока объявлений:

```
селектор {
    правило
}
```

Селектор определяет то, на какие элементы страницы распространяется данное правило. Селекторов может быть несколько. В этом случае они перечисляются через запятую. Каждое объявление представляет собой пару свойство-значение, разделенные символом «:». Объявления разделяются между собой знаком «;» и находятся в блоке объявления стилей, расположенном между фигурными скобками.

```
селектор,
селектор,
селектор { /* блок объявления стилей */
    свойство: значение;
    свойство: значение;
    свойство: значение;
    свойство: значение;
    свойство: значение;
}
```

Комментарии в CSS задаются следующим образом:
Основные ошибки при написании CSS:

- пропущенная точка с запятой или отсутствующая закрывающая скобка.

  Например, в следующем фрагменте пропущена точка запятой после объявления «color: red».

```
body {
    background: #f1f1f1;
}
h1 {
    font-size: 34px;
    color: red;
    text-align: center;
}
h2 {
    font-size: 24px;
}
h1,
h2 {
    font-family: Tahoma;
}
p {
    font-size: 18px;
    color: gray;
    font-family: Arial;
}
```

  В результате парсер считает, что значение у свойства «color» это «red text align: center», что, конечно, неправильно.

  Если же забыта закрывающая фигурная скобка, парсер игнорирует все стили, которые расположены после забытой фигурной скобки.

2.3. Селекторы тегов, классов и идентификаторов

Селекторы тегов записываются как имена тегов. Поэтому селектор, который выделяет все заголовки первого уровня выглядит просто как «h1».
Более интересны селекторы классов. Пусть дан следующий элемент:

```
<div class="heading red">Красный заголовок</div>
```

```
div.heading {
    font-size: 50px;
}
div.red {
    color: red;
}
```

Этот элемент принадлежит двум классам: heading и red. Чтобы стилизовать этот элемент, в CSS следует записать:

```
<div class="heading red">Красный заголовок</div>
<div class="red">Красный заголовок</div>
```

```
div.heading {
    font-size: 50px;
}
div.red {
    color: red;
}
```

Данные правила устанавливают всем элементам div с классом heading размер шрифта в 50px, а всем div с классом red красный цвет текста. Таким образом, текст в данном элементе будет отображен шрифтом в 50px красным цветом.
Для другого элемента, который принадлежит только к классу red, правила heading применяться не будут.
Селекторы идентификаторов задаются с помощью символа решетки, после которого идет значение атрибута id. В отличие от селекторов класса, id может быть только один.
Так, например, следующий код не будет работать:

```
<div id="heading red">Красный заголовок</div>
```

```
div#heading {
    font-size: 50px;
}
div#red {
    color: red;
}
```

В данном случае идентификатор содержит в себе символ пробела, который следует экранировать. Вообще говоря, рекомендуется не использовать символ пробела в идентификаторах.
Рабочий пример:

```
<div id="red">Красный заголовок</div>
```

```
div#heading {
    font-size: 50px;
}
div#red {
    color: red;
}
```

Применится второй селектор, и текст будет красным.
Классическая дилемма начинающих верстальщиков: какой селектор использовать — селекторы классов или селекторы идентификаторов для задания стилей. Рекомендуется всегда использовать селекторы классов, так как нет гарантии, что изначально уникальный на странице элемент не перестанет быть уникальным. Идентификаторы используются для логических или функциональных элементов на странице, например в связке for-id у label-input).
Существует также универсальный селектор.

```
<h2>Красный заголовок</h2>
<div>Красный заголовок</div>
<p>Красный заголовок</p>
```

```
* {
    color: red;
}
```

Он позволяет выделить все элементы на странице.
Мощь CSS состоит не столько в селекторах, сколько в возможности их комбинировать, тем самым задавая сложные условия:

- Комбинирование селектора тега и селектора класса (или нескольких селекторов класса):

  ```
  div.class { color: red; }
  p.class.class2 { color: red; }
  ```

- Комбинирование селектора тега и селектора идентификатора

  ```
  div#id { color: red; }
  ```

- Комбинирование селектора тега, селектора идентификатора и селектора класса:

```
a#id.class { color: red; }
```

- Можно даже комбинировать два селектора идентификатора, но это не имеет смысла.

```
div#id#id2 { color: red; }
```

- Также можно комбинировать универсальный селектор с любым селектором, но такая запись считается избыточной: эти два примера дадут одинаковый результат, и * можно опустить.

```
*.class { color: red; }
.class { color: red; }
```

- и так далее...

Еще один вид селекторов — селекторы атрибутов. Например, можно выделить все элементы, которые содержат определенный атрибут:

```
/* Элементы содержащие атрибут */
[href] { color: red; }

/* Значение атрибута в точности равно заданному */
[href="http://ya.ru"] { color: red; }
/* Значение атрибута содержит заданное значение */
[href*="http://ya.ru"] { color: red; }

/* Значение атрибута начинается с заданного значения */
[href^="https://"] { color: red; }
/* Значение атрибута заканчивается заданным значением */
[href$="ya.ru"] { color: red; }
```

В CSS существует два вида псевдоселекторов: псевдоэлементы и псевдоклассы.

Динамические псевдоклассы Псевдоклассы — такие селекторы, которые позволяют выбрать элементы в зависимости от их состояния. Следующие псевдоклассы соответствуют состояниям ссылок:

```
a:link { color: blue }
a:active { color: red }
a:hover { color: green }
a:visited { color: purple }
a:focus { color: yellow }
```

Эти псевдоклассы появились еще в ранних спецификациях CSS.
В более поздних спецификациях появился целый ряд новых псевдоклассов:

- «:enabled» и «:disabled» позволяют выбирать элементы в зависимости от их доступности для взаимодействия с пользователем.
- «:checked» позволяет выбрать все элементы, в которых есть атрибут checked.
- «:indeterminate» соответствует неопределенному состоянию checkbox'ов. Его нет в HTML, но оно доступно из javascript. Чаще всего оно употребляется во вложенных списках с чекбоксами, в которых одним кликом по флажку можно выбрать всю категорию. Такой флажок принимает неопределенное состояние, если в соответствующей категории есть выбранные и не выбранные элементы.
- :read-only позволяет выбрать элементы с атрибутом "только для чтения".
- :valid позволяет выбрать валидные элементы форм.

Например:

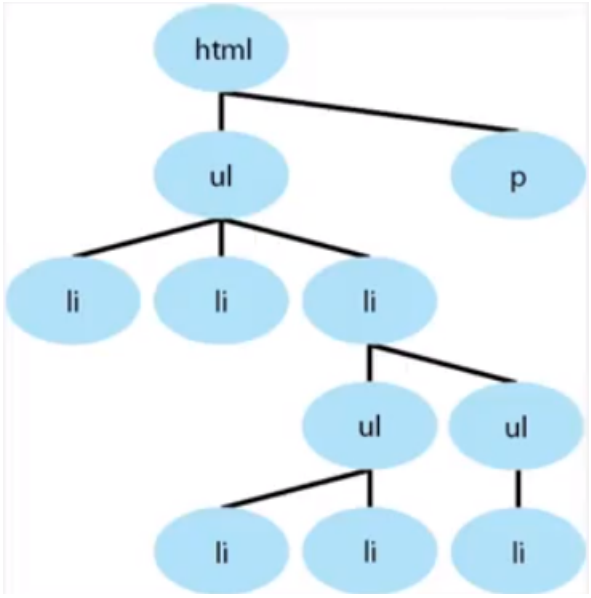
```
input.example { padding: 5px; }
.example:enabled { background: red; }
.example:disabled { background: yellow; }
.example:valid { background: green; }
```

Структурные псевдоклассы Следующее семейство псевдоклассов позволяет выбирать элементы в зависимости от их положения в структуре HTML документа.

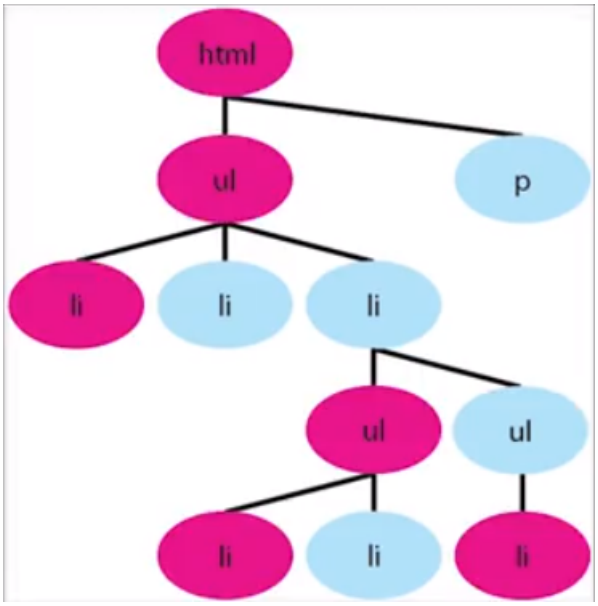
- :root выбирает корневой элемент HTML-документа
- :first-child/:last-child выбирают первый или последний дочерний элемент.
- :nth-child/:nth-last-child(2n+1) позволяют выбрать n-ый элемент или n-ый элемент с конца. Здесь дополнительно указано, как строится номер элемента. В выражении $2n + 1$, n принимает значение от нуля до бесконечности. Таким образом, выбираются все нечетные элементы.
- :nth-of-type/:nth-last-of-type(-n+4)

- :only-child/:only-of-type выбирает элемент, который является единственным дочерним для своего родительского элемента, или единственным данного типа.
- :empty выбирает пустые элементы.

Для примера можно рассмотреть следующую структуру HTML документа:



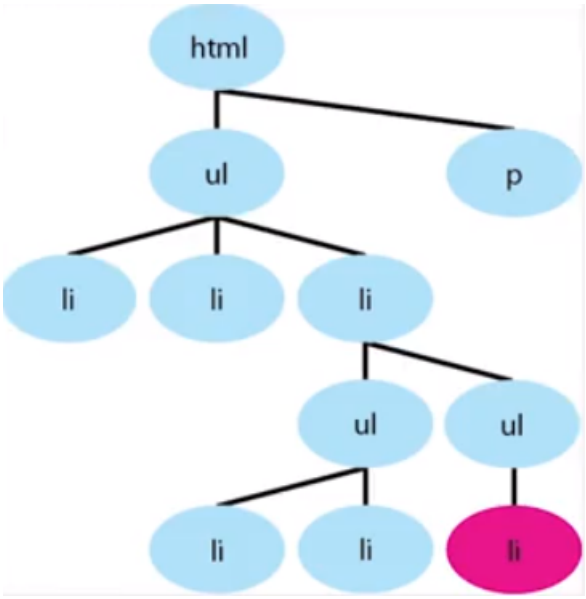
Так она схематично изображена в виде дерева. В результате применения селектора :first-child будут выделены следующие элементы:



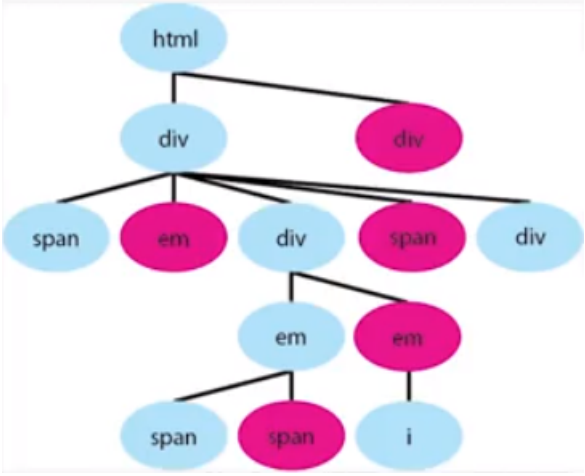
А в результате применения :last-child

tex/lastchild.png

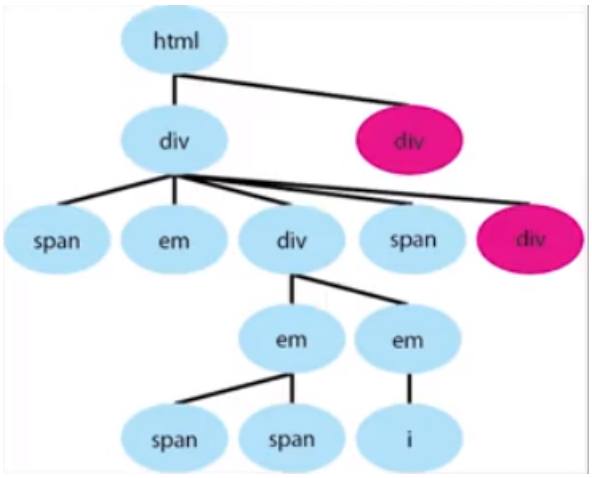
А в результате применения :only-child



Рассмотрим немного другое дерево. Результат применения селектора :nth-child(2n), который выбирает четные дочерние элементы, имеет вид:



При применении селектора :nth-of-type типом считается часть селектора до двоеточия. Результат применения div:nth-child(2n):



Что происходит внутри браузера? Для начала браузер выбирает все div'ы на всех уровнях. А потом среди выбранных div'ов на каждом из уровней (только среди соседних div'ов) он выбирает четные.
Если же мы не будем указывать тип, то эту операцию браузер будет проделывать с каждым тегом, то есть он выберет сначала все div'ы и среди них

выберет четные, потом все span'ы, все em'ы и так далее. Вот так работает селектор nth-of-type и аналогичные ему.

Псевдоэлементы Следующий класс псевдоселекторов — псевдоэлементы. Они называются так, поскольку не содержатся в HTML структуре документа. К таким селекторам относятся:

- ::first-letter который выделяет первую букву текста:

```
::first-letter { color: red }
```

- ::first-line который выделяет первую строку текста:

```
::first-line { color: blue }
```

- ::before, ::after выделяют создаваемые в структуре документа элементы сразу после открывающего тега и сразу до закрывающего соответственно.

```
::before,
::after {
    color: red;
    font-weight: bold;
    content: '»';
}
::before { content: '«';}
```

2.4. Отношения элементов в CSS

В CSS выделяют следующие виды отношений элементов:

- Потомок. Например, следующий селектор выделяет все div которые являются потомками (другого) div.

```
div div
div >> div /* CSS Selectors Level 4 */
```

- Непосредственный потомок.

```
div > div
```

- Элемент, следующий за данным элементом (брат):

```
div + div
```

- Элементы, следующие за данным элементом (братья):

```
div ~ div
```

Примеры:

2.5. Специфичность

В этом разделе речь пойдет об одном из самых важных и базовых понятий в CSS, о специфичности.
Пусть дан следующий CSS файл:

```
1 div {
2     color: red;
3 }
4
5 div {
6     color: blue;
7 }
```

Для одного и того же элемента сначала задается красный цвет, а потом синий. В результате цвет будет выбран синий.
Пусть теперь дан другой CSS код:

```
1 html div {
2     color: red;
3 }
4
5 div {
6     color: blue;
7 }
```

Оба селектора здесь также в любом html документе выбирают одинаковый набор элементов. Но в данном случае в результат будет выбран красный цвет. Дело в том, что каждому из селекторов CSS сопоставляется трехчисловое значение, которое называется специфичность. Считается специфичность следующим образом:

- Первое число — количество селекторов по идентификатору.
- Второе число — количество селекторов классов, псевдоклассов (кроме :not) или по атрибуту.
- Третье число — количество селекторов элементов (тегов) или псевдоэлементов

Правила CSS сортируются по специфичности в лексикографическом порядке. Например, пусть дан следующий CSS-код:

```
div {
    color: blue;
    font-weight: bold;
    font-size: 12px;
}

html div {
    color: red;
}

div {
    font-size: 15px;
}
```

Первое, что нужно сделать — подсчитать специфичность каждого селектора:

```
div {
    color: blue; /* 0 0 1 */
    font-weight: bold; /* 0 0 1 */
    font-size: 12px; /* 0 0 1 */
}

div {
    font-size: 15px; /* 0 0 1 */
}
```

```
10
11 html div {
12     color: red; /* 0 0 2 */
13 }
```

Все три селектора выбирают один и тот же набор элементов — все div'ы.
Чтобы применить к конкретному элементу div заданные правила, отсортируем их в лексикографическом порядке по специфичности, максимально сохраняя порядок.

```
1 color: blue; /* 0 0 1 */
2 font-weight: bold; /* 0 0 1 */
3 font-size: 12px; /* 0 0 1 */
4 font-size: 15px; /* 0 0 1 */
5 color: red; /* 0 0 2 */
```

Теперь нужно последовательно применять каждое правило, перетирая, если необходимо, уже установленные значения.
В CSS есть еще два способа переопределять приоритет правил: инлайн и !important.
Инлайн — это способ указать стили непосредственно внутри HTML элемента:

```
1 <div style="color: blue"></div>
```

Ключевое свойство !important используется внутри значения объявления, которое повышает приоритет данного объявления:

```
1 div {
2     color: green !important;
3 }
```

Также можно указать !important внутри инлайн стилей:

```
1 <div style="color: white !important"></div>
```

Инлайн стили по умолчанию приоритетнее стилей в CSS. Стили в CSS с !important приоритетнее инлайн стилей. Инлайн стили с !important приоритетнее всего.
Также в CSS существует механизм наследования стилей. А именно: некоторые стили применяются не только к целевому элементу, но и к его потомкам. Унаследованные стили не имеют специфичности, то есть их всегда перебивает любой селектор.
Именно поэтому не рекомендуют пользоваться селектором *, так как его специфичность будет «0 0 0». Пусть даны следующий CSS код и HTML разметка:

```html
<div>
    Привет,
    <abbr title="Уральский федеральный университет">
        УрФУ
    </abbr>
</div>
```

```css
div {
    color: green; /* 0 0 1 */
    /* abbr { color : green } - нет специфичности */
}
```

В результате зеленым будет отображен не только текст непосредственно внутри div, но и текст внутри элемента abbr, который унаследовал цвет от элемента div.
Другой пример. Кроме селектора элемента есть универсальный селектор.

```css
* {
    color: red; /* 0 0 0 */
}
div {
    color: green; /* 0 0 1 */
    /* abbr { color : green } - нет специфичности */
}
```

В данном случае универсальный селектор будет приоритетнее унаследованных стилей.
Не все стили наследуются. Например, наследуются:

```
color                list-style-position
cursor               list-style-image
direction            list-style
empty-cells          line-height
font-family          quotes
font-size            text-align
font-weight          text-indent
font-style           text-transform
font-variant         visibility
font                 white-space
letter-spacing       word-spacing
list-style-type
```

Полный список доступен по ссылке.
Стили, которые указывает верстальщик — не единственный источник стилей для страницы. Причем порядок применения стилей CSS следующий:

- Стили браузера
- Стили пользователя и/или плагинов браузера
- Стили страницы
- Стили страницы с !important
- Стили пользователя и/или плагинов браузера с !important

Именно такая последовательность наложения стилей и называется каскад.
Числовые значения в CSS:

- Целые числа (1, 2, 3...)
- Дробные числа (.5, 1.5)
- Процентные значения (50%)

Абсолютные единицы измерения длины:

- Миллиметры (mm)
- Сантиметры (cm)
- Дюймы (in)
- Пункты – используется в типографиях, 1 дюйм = 72 пункта (pt)
- Пики – 12 пунктов (pc)

Абсолютные единицы удобны при определении CSS для печати. В вебе же практически не используются
Относительные единицы измерения длины:

- пикселы (px) До сих пор не утихают споры, считать ли пиксели относительной единицей измерения. Мы будем считать, что является, так как в последнее время пиксели как они задаются для браузера не совпадают, например, с физическими пикселями на экране дисплея.

- em – зависит от размера шрифта (1 em = размер шрифта)
- rem – зависит от размера шрифта корневого элемента (1 rem = размер шрифта корневого элемента)
- ex – зависит от высоты символа x в данном шрифте
- ch – зависит от ширины символа 0 в данном шрифте
- vh/vw – 1/100 высоты и ширины vieport'a соответственно
- vmax – 1/100 от максимума между высотой и шириной vieport'a
- vmin – 1/100 от минимума между высотой и шириной vieport'a

Цвета Цвет в CSS задается (на примере красного цвета):

- С помощью именованных констант: red.
- Функционально: rgb(255, 0, 0) или rgba(255,0,0,1)
- Задание в шеснадцатиричном виде: #ff0000 или #f00 (краткая форма)

В CSS существуют другие типы значений:

- Именованные слова: bold, underline
- Комбинации: 1px solid red
- Функциональные: url(), attr(), rgb()

Существуют следующие способы вставить CSS код в документ:

- Инлайн стили: Такой способ не далеко ушел от использования тега font.
- Используя тег style:
- Подключить внешний файл с помощью директивы import внутри тега style:
- Подключить внешний файл с помощью тега link: