Cascading Style Sheets

Данильченко Анна Александровна

Преподаватель кафедры программного обеспечения систем ЖГТУ

Cascading Style Sheets

- ***CSS (каскадные листы/таблицы стилей)*** — это язык для описания стилей, которые задают внешний вид документов, написанных при помощи языков разметки.
- ***CSS*** позволяет устанавливать цвета, шрифты, отступы, фоны, размеры, управлять местоположением (позиционированием) и обтеканием элементов, реализовывать различные оформительские решения.

Идея использования HTML совместно с CSS

- **Разделение структуры и оформления документа.**

HTML используется для описания **логической** структуры документа (выделения заголовков, подзаголовков, абзацев, таблиц, гиперссылок и других базовых логических элементов)

CSS описывает **физическое** форматирование документа (как должен выглядеть тот или иной логический элемент документа)

Разделение оформления и структуры документа дает такие преимущества:

- возможность параллельной разработки/модификации документа и его оформления/дизайна.
- расширенные возможности по сравнению с HTML;
- возможность одновременного изменения внешнего вида множества документов при помощи одной таблицы стилей;
- возможность установки различного форматирования для различных носителей информации (экран, печать и т. п.).

Уровни CSS

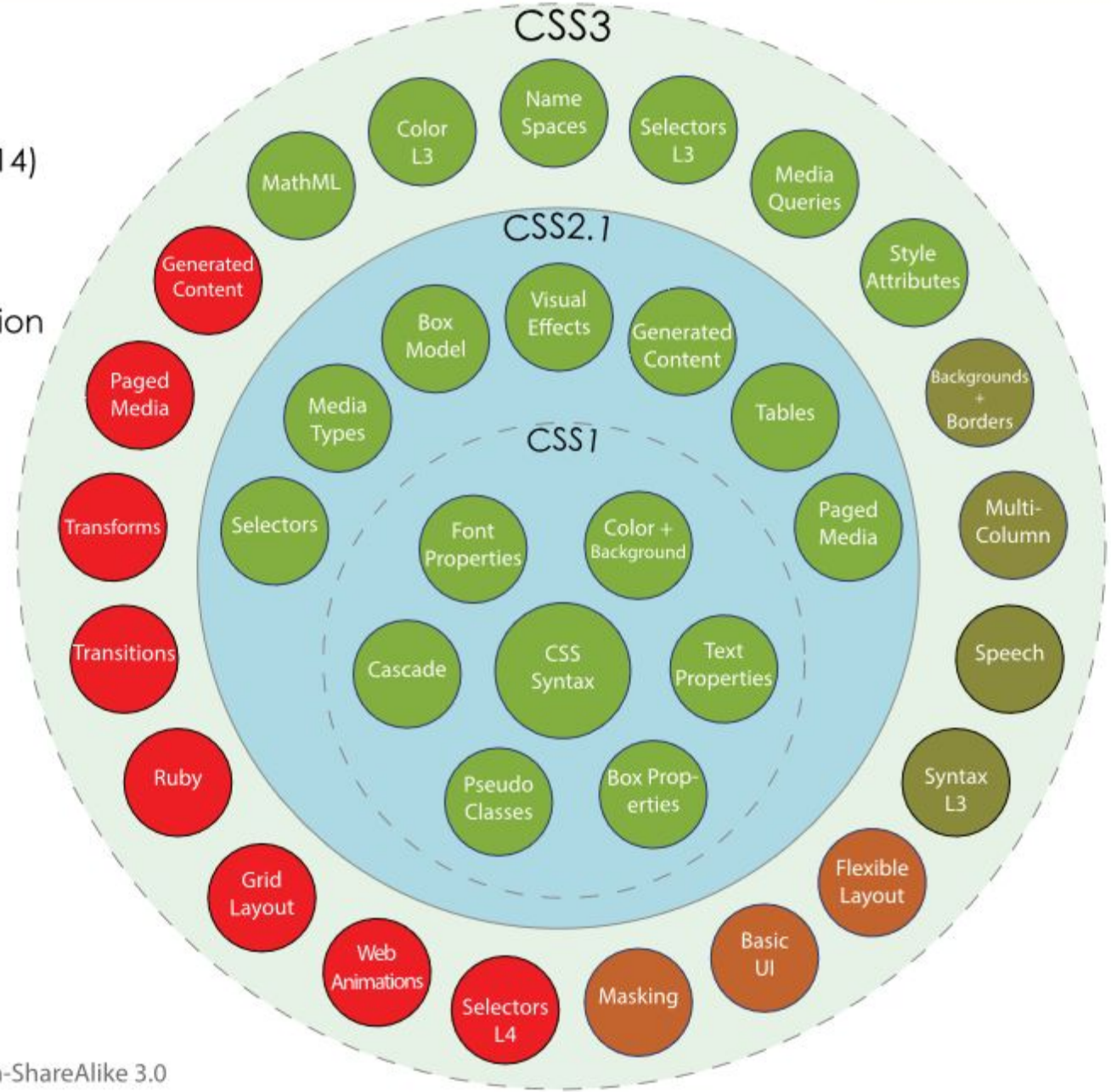
ПРИ РАЗРАБОТКЕ СТАНДАРТА CSS КОНСОРЦИУМ ВСЕМИРНОЙ ПАУТИНЫ ПРИНЯЛ РЕШЕНИЕ КЛАССИФИЦИРОВАТЬ НОВЫЕ СТАНДАРТЫ CSS НЕ ПО ВЕРСИЯМ, КАК ПРИНЯТО В РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, А ПО УРОВНЯМ.

Уровень CSS	Статус документа W3C	Дата принятия документа
CSS1 (уровень1)	рекомендация	17 декабря 1996 г., отредактирована 11 апреля 2008 г.
CSS2 (уровень 2)	рекомендация	12 мая 1998 г.
CSS2.1 (уровень 2, редакция 1)	кандидат на рекомендацию	19 июля 2007 г.
CSS3 (уровень 3)	в стадии разработки	

CSS3

Taxonomy & Status (October 2014)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



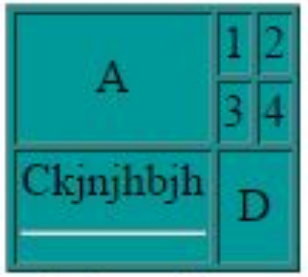
Способы включения CSS в HTML

1. Использование внешних таблиц стилей
2. Использование внутренних таблиц стилей
3. Использование встраиваемых стилей

Встраиваемые стили

Описание стиля располагается непосредственно внутри тега элемента, который описывается. Это делается с помощью параметра STYLE

```
<TABLE BORDER style="background:#099">
```

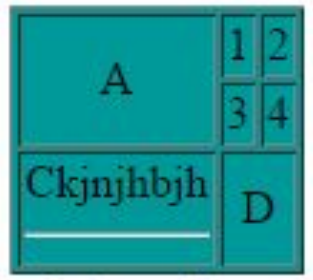


Этот метод нежелателен, он приводит к потере одного из основных преимуществ CSS – возможности отделения информации от описания оформления информации.

Внутренние таблицы стилей

Описание стилей располагается в коде Web-странички, внутри тега <STYLE type="text/css">... </STYLE>. Тег <STYLE> размещается внутри контейнера HEAD

```
<style>
table{background :#099;}
</style>
```

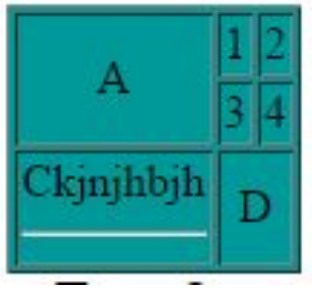


В этом случае описанные стили можно использовать для элементов, располагающихся в пределах странички.

Внешние таблицы стилей

Информация о стилях располагается в отдельном файле (*.css). Это имеет смысл в случае, если планируется применять эти стили к большему, чем одна, количеству страниц.

```
table{background :#099;}
```



Подключение файла *.css

<link rel="stylesheet" type="text/css" href="style2.css">

example.html

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="mystyles,css"/>
  </head>
  <body>
    содержание страницы HTML-документа
  </body>
</html>
```

Ссылка может быть как на «локальную» страницу стилей, созданную специально для этого документа, так и на «глобальную», хранящуюся в сети Интернет.

example1.html

```
<head>
  <link rel="stylesheet" type="text/css"
        href="http://www.google.com/uds/css/gsearch.css" />
</head>
```

Базовый синтаксис

- **Стиль** – это набор параметров, задающих представление некоторого элемента веб-страницы.
- **Селектор** – это имя стиля.

Стили описываются в такой форме:

```
селектор {
        свойство1: значение1;
        свойство2: значение2;
        …
        свойствоN: значениеN;
}
```

Пример:

```
.footer {
    background: #960869;
}
```

Синтаксис CSS

Типы селекторов

- В качестве селекторов (имен стилей) могут использоваться:

- универсальный селектор
- теги
- классы, определяемые пользователем
- идентификаторы, определяемые пользователем;

Типы селекторов

1. Универсальный селектор
2. Селектор тегов
3. Селектор атрибутов
4. Селектор по классу
5. Селектор по идентификатору
6. Контекстный селектор
7. Дочерний селектор
8. Соседний селектор

Универсальный селектор
один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста

- * { Описание правил стиля }
- * { font-family: Arial, Verdana, sans-serif; */* Рубленый шрифт для текста */*
- font-size: 96%; /* Размер текста */ }

Селектор тегов

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

Тег { свойство1: значение; свойство2: значение; ... }

Пример:

```
P { text-align: justify; /* Выравнивание по ширине */
 color: green; /* Зеленый цвет текста */
 }
```

Селектор атрибутов

Устанавливает стиль для элемента, если задан специфичный атрибут тега.

[атрибут] { Описание правил стиля }

Селектор[атрибут] { Описание правил стиля }

A[href^="http://"] - начинается с определённого текста

A[href$=".ru"] - оканчивается определённым текстом

A[href*="htmlbook"] - содержит указанный текст

[class~="block"] - Одно из нескольких значений атрибута

DIV[class|="block"] - Дефис в значении атрибута
(**<div** class="block-menu-therm"**>)**

[attr~=value]

➔ Пример задания правила для всех элементов на странице, имеющих атрибут `data-content`, значение которого содержит `news`, отделённое от других с помощью пробела (например будет выбран элемент, если у него атрибут `data-content` равен `hot-news news news-football`):

```
/* селектор [data-content~="news"] выберет все элементы на странице, имеющих атрибут data-content, значение которого содержит news, отделённое от других с помощью пробела */
[data-content~="news"] {
  background-color: brown;
}
```

CSS

Пример:

```
A[href$=".ru"] { /* Если ссылка заканчивается на .ru
  */
 background: url(images/ru.png) no-repeat 0 6px; /*
   Добавляем фоновый рисунок */
 padding-left: 12px; /* Смещаем текст вправо */ }
 A[href$=".com"] { /* Если ссылка заканчивается на
   .com */
background: url(images/com.png) no-repeat 0 6px;
 padding-left: 12px; }
```

HTML

```
<a href="http://www.yandex.com">Yandex.Com</a> |
<a href="http://www.yandex.ru">Yandex.Ru</a>
```

Задание

Изображения с атрибутом alt содержащим слово «Ничосе» обвести зеленой рамкой. (border:1px solid green;)

Создать меню из 3 ссылок подсветить красным ссылки с защищёнными протоколами

Селектор по классу

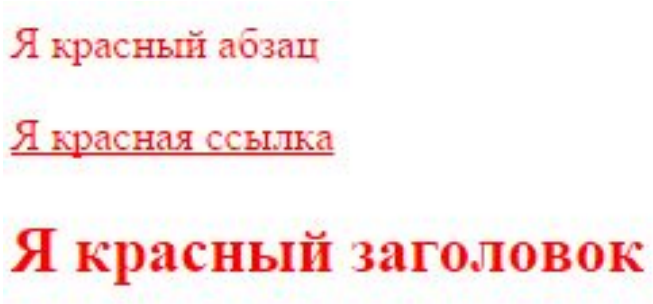
к элементам страницы добавляем слово class=“name” и задаем стили для класса

Пример HTML

```
<p class="red">Я красный абзац</p>
<a class="red" href="#">Я красная ссылка</a>
<h2 class="red">Я красный заголовок</h2>
```

CSS

```
.red{
   color:#F00;}
}
```



Селектор по идентификатору

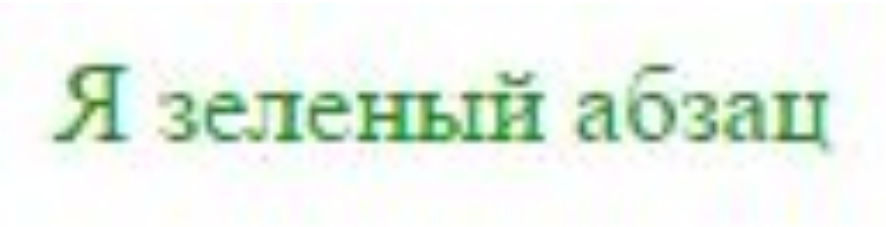
задаем элементу страницы уникальный идентификатор

Пример HTML

```
<p id="green">Я зеленый абзац</p>
```

CSS

```
#green{
   color:#090;
}
```



Идентификаторы или классы?????

Идентификаторы	Классы
В коде документа каждый идентификатор уникален и должен быть включён лишь один раз. Имя идентификатора чувствительно к регистру. Через метод getElementById можно получить доступ к элементу по его идентификатору и изменить свойства элемента. Стиль для идентификатора имеет приоритет выше, чем у классов.	Классы могут использоваться в коде неоднократно. Имена классов чувствительны к регистру. Классы можно комбинировать между собой, добавляя несколько классов к одному тегу.

Контекстный селектор

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определённом контексте. Например, задать стиль для тега ***<b>*** *только когда он располагается внутри контейнера* ***<p>****.*

Контекстный селектор состоит из простых селекторов разделенных пробелом.

Тег1 Тег2 { … }

Контекстный селектор

Пример HTML

```
<div>
<b>Жирное начертание текста</b>
</div>
 <p><b>Одновременно жирное начертание текста и выделенное цветом</b>
</p>
```

Жирное начертание текста

Одновременно жирное начертание текста и выделенное цветом

CSS

```
P B { font-family: Times, serif; /* Семейство шрифта */
color: navy; /* Синий цвет текста */ }
```

Контекстный селектор

HTML

```
<div class="menu">
 <a href="http://htmlbook.ru/1.html">Русская кухня</a>
   | <a href="http://htmlbook.ru/2.html">Украинская
   кухня</a> |
   <a href="http://htmlbook.ru/3.html">Кавказская
   кухня</a>
</div>
 <p>
<a href="http://htmlbook.ru/text.html">Другие
   материалы по теме</a>
</p>
```

css

```
A { color: green; /* Зеленый цвет текста для всех ссылок */ }

 .menu { padding: 7px; /* Поля вокруг текста */
border: 1px solid #333; /* Параметры рамки */
 background: #fc0; /* Цвет фона */ }

 .menu A { color: navy; /* Темно-синий цвет ссылок */ }
```

Дочерний селектор

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента

body

<div class="main"**>**

** **

****Lorem ipsum dolor sit amet****, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat.** **

** **

****Ut wisis enim ad minim veniam****

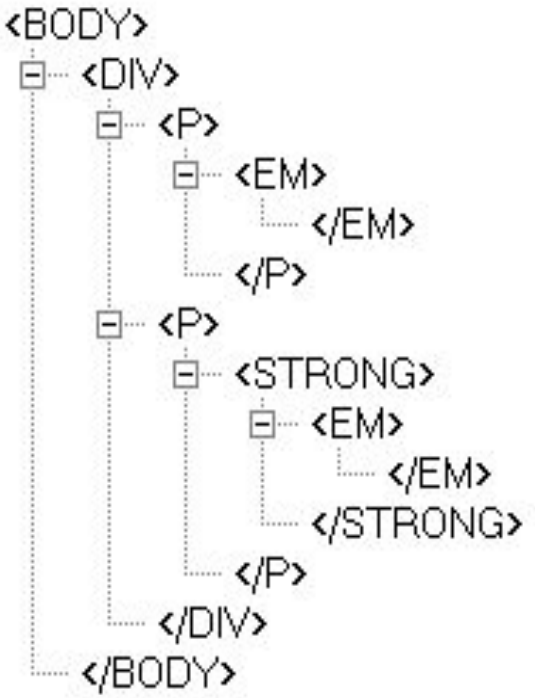
****

, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

** **

** **

</body>



Дочерний селектор

Селектор 1 > Селектор 2 { Описание правил стиля }

HTML

** **

<p>

****Lorem ipsum dolor sit amet****

, consectetuer adipiscing elit, sed diem nonummy nib… …
ut lacreet

aliguam erat volutpat. **> **dolore magna****

</p

** **

CSS

DIV I { /* Контекстный селектор */ color: green; /* Зеленый цвет текста */ }

P > **I** { /* Дочерний селектор */ color: red; /* Красный цвет текста */ }

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet *dolore magna* aliguam erat volutpat.

Соседний селектор

Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа.

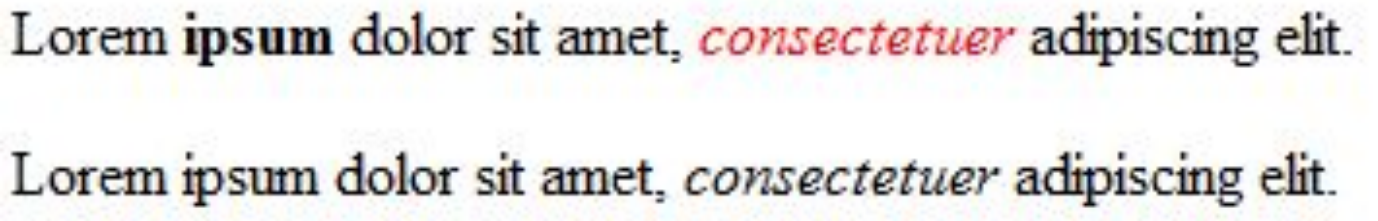
E + F { Описание правил стиля }

Стиль при такой записи применяется к элементу F, но только в том случае, если он является соседним для элемента E и следует сразу после него.

Lorem ipsum dolor sit amet. - дочерний

Lorem ipsum dolor <var>sit</var> amet. - соседний

Соседний селектс



Пример HTML

** **Lorem ****ipsum **** dolor sit amet, ****consectetuer**** adipiscing elit.

** **

** **Lorem ipsum dolor sit amet, ****consectetuer**** adipiscing elit.

** **

CSS

b + **i** { color: red; /* Красный цвет текста */ }

Псевдоклассы

- :first-child применяет стилевое оформление к первому дочернему элементу своего родителя.

```
tr:first-child{
    background-color:#06F;}
```

:nth-child используется для добавления стиля к элементам на основе нумерации в дереве элементов.

```
nav li:nth-child(odd){                background-color:#CCC;}
```

odd	Все нечетные номера элементов.
even	Все четные номера элементов.
число	Порядковый номер дочернего элемента относительно своего родителя. Нумерация начинается с 1, это будет первый элемент в списке.

last-child задает стилевое оформление последнего элемента своего родителя

LI:nth-child(3n+3) - Он выбирает каждый третий элемент внутри маркированного списка: это 3-й, 6-й, 9-й, 12-й и т.д.

Если использовать большое положительное число `b` в формуле `an+b`, то можно выделять все элементы, за исключением начальных. И чем больше `b`, тем больше начальных пропускается. Расчет для `:nth-child(n+8)`:

```
1. n=0; 0+8 = 8;  8-ой элемент
2. n=1; 1+8 = 9;  9-ой элемент
3. n=2; 2+8 = 10; 10-ый элемент
...
```

В общем виде значение этого псевдокласса задаётся с помощью выражения: `an+b`, где `a` и `b` — целые числа, а `n` — счетчик, принимающий целые значения от 0 и больше: 0,1,2,3...

Если после вычисления выражения браузер находит элемент с полученным номером, то он применяет к нему стили. Рассмотрим пример вычисления номеров для `:nth-child(2n)`.

```
1. n=0; 2*0 = 0; нет элементов
2. n=1; 2*1 = 2; 2-ой элемент
3. n=2; 2*2 = 4; 4-й элемент
```

Можно использовать отрицательный `n`. Расчёт для `:nth-child(-n+14)`:

```
1. n=0; 0+14 = 14; 14-ый элемент
2. n=1; -1+14 = 13; 13-ый элемент
...
15. n=14; -14+14 = 0; нет совпадений
```

Т.е. `:nth-child(n+8)` означает *не* выделять 7 элементов вначале, а все остальное выделить. `:nth-child(-n+14)` обозначает выделить 14 элементов в начале, а все остальное *не* выделять.

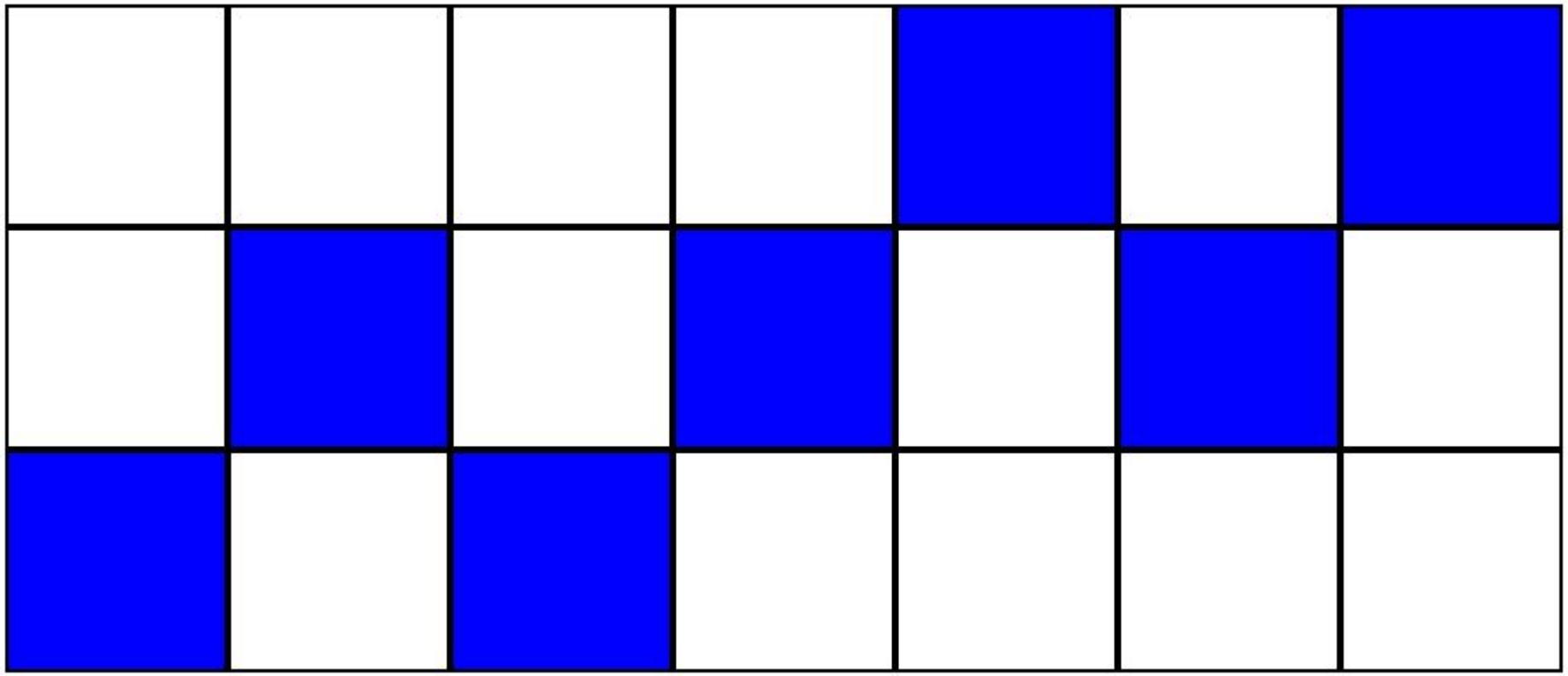
Комбинированные выражения

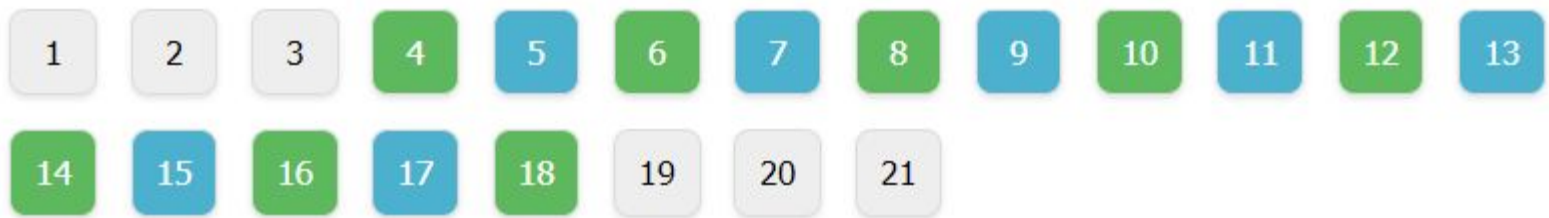
На предыдущих шагах мы с помощью разных выражений выделяли *разные множества* элементов. Существует возможность задавать множество с помощью комбинирования выражений.

Например: `:nth-child(n+8):nth-child(-n+14)`. Итоговое множество получается как пересечение двух исходных:

```
:nth-child(n+8) выделит: 8-21
:nth-child(-n+14) выделит: 1-14
на пересечении 1-14 и 8-21: 8-14
```

Задание




1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

Псевдокласс :only-child

Псевдокласс `:only-child` используется для выбора элементов, если они являются единственными дочерними элементами внутри родительского контейнера.

Например:

```
h2:only-child {
  ...
}
```

Селектор `h2:only-child` выберет все элементы `h2`, если они являются единственными дочерними элементами своего родителя.

Псевдокласс :nth-last-child(выражение)

Псевдокласс `:nth-last-child()` выполняет те же действия что и `:nth-child()` за исключением того, что отсчет элементов в родителе вёдется не с начала, а с конца. В псведоклассе `:nth-last-child(выражение)` в качестве выражения можно использовать те же вещи, т.е. число, формулу, или ключевые слова `odd` или `even`

Псевдокласс :first-of-type

Данный псевдокласс очень похож на `:first-child`, но в отличие от него он выбирает не просто первый элемент, а первый элемент своего родителя **с учётом его типа**.

Например, выберем все элементы `p`, которые являются первыми элементами указанного типа у своего родителя:

```
p:first-of-type {
  ...
}
```

HTML:

```
<section>
  <h2>...</h2>
  <p>...</p> <!-- будет выбран этот элемент -->
  <p>...</p>
</section>
<section>
  <h2>...</h2>
  <p>...</p> <!-- будет выбран этот элемент -->
  <p>...</p>
</section>
```

Псевдокласс :last-of-type

Данный псевдокласс предназначен для выбора элементов, которые являются последними дочерними элементами данного типа своего родителя.

Пример записи:

```
li:last-of-type {
  ...
}
```

Псевдокласс :only-of-type

Псевдокласс `:only-of-type` применяется для выбора элементов, если каждый из них являются единственным дочерним элементом данного типа внутри своего родителя. В отличие от `:only-child` псведокласс `:only-of-type` работает аналогично, но с учётом типом элемента.

Пример:

```
p:only-of-type {
  ...
}
```

Псевдокласс :nth-of-type(выражение)

Данный псевдокласс предназначен для выбора элементов по их порядковому номеру в родителе с учетом их типа.

Псевдокласс `:nth-of-type(выражение)` похож на `:nth-child(выражение)` с разницей лишь а том, что он учитывает тип элемента. В качестве выражения также можно использовать число, формулу или ключевые слова `odd` или `even`. Отсчёт элементов в родителе начинается с 1.

Например:

```
<section>
  <h2>...</h2>
  <p>...</p> <!-- Селектор p:nth-child(2) выберет этот элемент -->
  <p>...</p> <!-- Селектор p:nth-of-type(2) выберет этот элемент -->
</section>
```

Селектор `p:nth-child(2)` выберет второй по счёту элемент, если он является `p`, а селектор `p:nth-of-type(2)` веберет второй дочерний элемент `p` среди дочерних `p`.

Псевдокласс :nth-last-of-type(выражение)

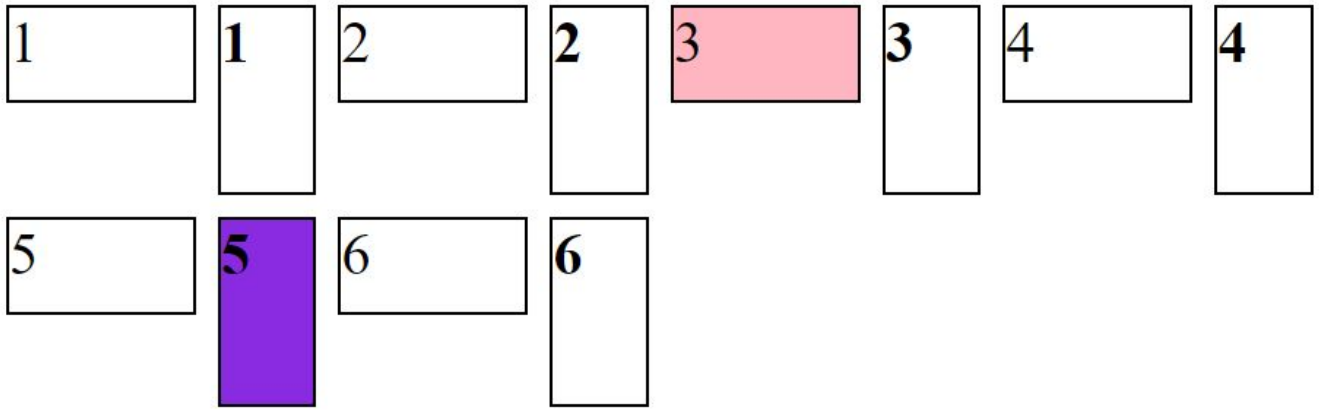
Псевдокласс `:nth-last-of-type(выражение)` аналогичен классу `:nth-of-type(выражение)` с разницей в том, что отсчёт дочерних элементов ведётся с конца.

Задание

```
<div>
    <span>1</span>
    <strong>1</strong>
    <span>2</span>
    <strong>2</strong>
    <span>3</span>
    <strong>3</strong>
    <span>4</span>
    <strong>4</strong>
    <span>5</span>
    <strong>5</strong>
    <span>6</span>
    <strong>6</strong>
</div>
```

Выделить третий span

Выделить пятый strong



Псевдоклассы для выбора элементов в зависимости от их состояния

К этой группе псевдоклассов можно отнести псевдоклассы: `:link`, `:visited`, `:hover`, `:active` и `:focus`.

Псевдоклассы `:link` и `:visited` предназначены исключительно для ссылок (элементов `a` с атрибутом `href`).

Псевдоклассы `:hover`, `:active` и `:focus` могут применяться не только к ссылкам, но и к другим элементам.

Псевдокласс :link

Псевдокласс `:link` предназначен для выбора не посещённых ссылок.

➔ Пример задания правила для всех элементов `a` с классом `external`, которые пользователь ещё не посетил:

```
/* селектор a.external:link выберет все элементы a с классом external пользователь по которым ещё не переходил */
a.external:link {
  color: red;
}
```

Псевдокласс :visited

Псевдокласс `:visited` предназначен для выбора посещённых ссылок.

➔ Пример задания правила для всех элементов `a`, расположенных в `.aside`, пользователь которые уже посетил:

```
/* селектор .side a:visited выберет все элементы a, находящиеся в .aside, пользователь которые уже посетил */
.aside a:visited {
  color: #000;
}
```

Псевдокласс :active

Псевдокласс `:active` предназначен для выбора элементов в момент когда они активируются пользователем. Например, когда пользователь нажал левой кнопкой мышкой на ссылку, но её ещё не отпустил. В основном данный класс применяется для ссылок (`a`) и кнопок (`button`), но может также использоваться и для других элементов.

➔ Пример задания CSS правила для всех элементов `a` когда они активируются пользователем:

```
/* селектор a:active выберет все элементы a, находящиеся в активном состоянии */
a:active {
  background-color: yellow;
}
```

Псевдокласс :hover

Псевдокласс `:hover` предназначен для выбора элементов при поднесении к ним курсора (при наведении на них).

➔ Пример задания CSS правила для всех элементов `.btn-warning` при поднесении к ним курсора:

```
/* селектор .btn-warning:hover выберет все элементы .btn-warning при поднесении к ним курсора */
.btn-warning:hover {
  color: #fff;
  background-color: #ff8f00;
}
```

Псевдокласс `:not(селектор)` является отрицающим селектором. С его помощью можно выбрать элементы, которые *НЕ* содержат указанный селектор:

```
li:not(:last-child) { }
```

Этот селектор выберет все теги `<li>`, *НЕ* являющиеся последними в их родителе.

Псевдокласс `:not` похож на оператор `!` в программировании:

```
if (!selector) { ... }
```

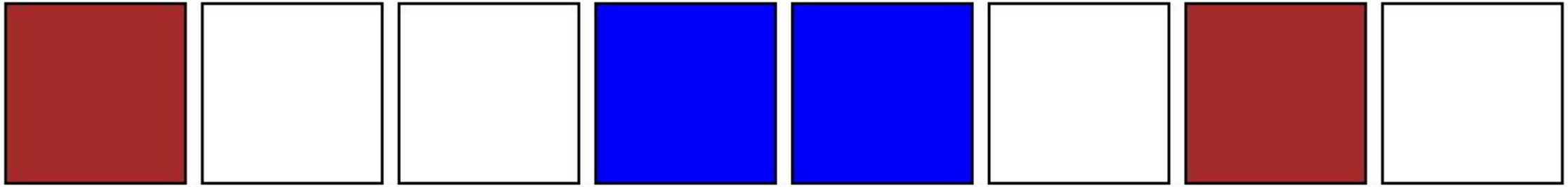
В качестве селектора могут указываться псевдоклассы, теги, идентификаторы, классы и селекторы атрибутов. Нельзя использовать двойной псевдокласс `:not`, то есть конструкция `:not(:not(...))` не сработает.

Также в комбинации с `:not` не применяются:

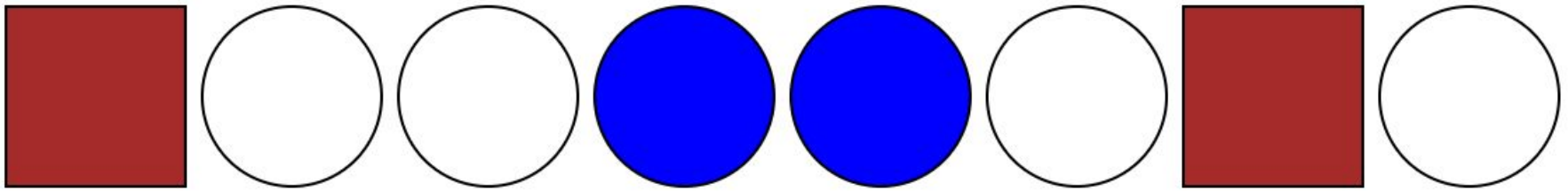
- объединение селекторов: например, `li:not(.heart.jack)` – некорректный селектор;
- псевдоэлементы: `li:not(::after)` – неправильная запись (подробнее о псевдоэлементах рассказано далее в курсе);
- селекторы-потомки, групповые селекторы или комбинации: например, нельзя писать `li:not(a span)` или `li:not(a + span)`.

Задание

- Сделать из списка



Используя not сделать пункты списка круглыми кроме



Отрицающий селектор `:not`, как и любые другие селекторы, можно комбинировать с другими. Например:

```
li:not(:first-child):not(:last-child) { }
```

Выберет все теги `<li>`, которые *НЕ* являются первыми и последними в их родителе.

Объединять можно неограниченное количество селекторов.

::first-line и ::first-letter

Псевдоэлемент `first-line` задает 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.

Пример использования:

```
p::first-line { }
```

Аналогично 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.

Пример использования:

```
p::first-letter { }
```

Задание

В Техасском Холдеме за столом играют до 10 игроков . Рядом с одним из игроков находится фишка с буквой «Д». Эта фишка называется фишкой дилера. Фишка дилера перемещается по кругу влево перед каждым игровым раундом.

Псевдоэлементы :before и :after

Псевдоэлемент :before применяется для отображения желаемого контента до содержимого элемента, к которому он добавляется. Работает совместно со свойством content.

:before наследует стиль от элемента, к которому он добавляется.

after
Псевдоэлемент, который используется для вывода желаемого текста после содержимого элемента, к которому он добавляется.

Свойство 'content'

Значение:	[<строка> | <uri> | <счетчик> |
Начальное значение:	пустая строка
Область применения:	псевдоэлементы :before и :after
Наследование:	нет
Процентное задание:	нет
Устройства:	все

<строка>
Текстовое содержимое

<uri>
Значением является URI,
обозначающий внешний ресурс.

<счетчик>
Счетчики могут задаваться с помощью двух различных функций
'counter()' или 'counters()'.

Пример

1. one
    1.1. one-one
        1.1.1. one-one-one
        1.1.2. one-one-two
        1.1.3. one-one-three
    1.2. one-two
    1.3. one-three
2. two
3. three
    3.1. three-one
    3.2. three-two
    3.3. three-three

```
ol>li{
display:block
}

ol>li:before{
content:counters(item,".") ". ";
counter-increment:item
}
ol{
counter-reset:item
}
```

```
<ol>
<li>one
    <ol>
    <li>one-one
        <ol>
        <li>one-one-one</li>

        <li>one-one-two</li>
        <li>one-one-three</li>
        </ol>
    </li>
    <li>one-two</li>
    <li>one-three</li>

    </ol>
</li>
<li>two</li>
<li>three
    <ol>
    <li>three-one</li>

    <li>three-two</li>
    <li>three-three</li>
    </ol>
</li>
</ol>
```

ol>li) используются для того, чтобы не обрабатывались элементы вложенных *ненумерованных* списков.

Свойство 'counter-increment' Оно определяет величину, на которую увеличивается содержимое счетчика при каждом новом вхождении элемента. По умолчанию приращение равно 1. Допускается использование отрицательных целых чисел.
Свойство 'counter-reset' Оно задает значение, которое присваивается счетчику при каждом новом вхождении элемента. По умолчанию оно равно 0.

```
<h1>This is a heading</h1>
<p>The ::before pseudo-element inserts content before
the content of an element.</p>
<h1>This is a heading</h1>
<p><b>Note:</b> IE8 supports the content property only
if a !DOCTYPE is specified.</p>
```

```
h1::before {
    content: url(smiley.gif);
}
```

This is a heading

The ::before pseudo-element inserts content before the content of an element.

This is a heading

Note: IE8 supports the content property only if a !DOCTYPE is specified.

Объединение селекторов

В CSS есть запись, которая фактически выполняет операцию логического умножения, «И». Селекторы, применяемые к одному элементу, в этом случае пишутся без пробелов:

```
.class1.class2 { }
```

Стили будут применяться ко всем элементам, которые одновременно имеют класс `class1` и `class2`. Ведь это же не новость для вас, что HTML-элементы могут одновременно иметь несколько классов. Например:

```
<div class="class1 class2">Блок с двумя классами</div>
```

По такому же принципу можно объединять любое количество абсолютно разных селекторов. И чем больше селекторов вы объединяете, тем больше условий должно совпасть для применения стилей.