

# Internet- технологии

ЛЕКЦИЯ №4

КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ.  
ДИРЕКТИВЫ (@-ПРАВИЛА).  
ПОЗИЦИОНИРОВАНИЕ ЭЛЕМЕНТОВ

# Содержание

- Селекторы CSS. Псевдоклассы.
- Селекторы CSS. Псевдоэлементы.
- Комбинирование селекторов.
- @-правила.
- Способы позиционирования элементов веб-страницы.
- Блочная модель. Свойство float.



# Селекторы. Псевдоклассы

Псевдо-классы предназначены для изменения стиля существующих элементов страницы в зависимости от их **динамического** состояния, например при работе со ссылками (:link, :visited, :hover, :active, :focus).

- link

```
a:link { color: #6699CC; }
```

Это - непосещённая ссылка

Это - непосещённая ссылка

- visited

```
a:visited { color: #660099; }
```

Это - посещённая ссылка

Это - посещённая ссылка

- active

```
a:active { background-color: #FFFF00; }
```

Нажмите здесь и держите клавишу нажатой

Нажмите здесь и держите клавишу нажатой

- hover

```
a:hover { color: orange; font-style: italic; }
```

Проведите указателем мыши над этой ссылкой

Проведите указателем мыши над этой ссылкой



# Селекторы. Псевдоклассы

:checked – применяется к элементам интерфейса, таким как **переключатели** (checkbox) и **флажки** (radio), когда они находятся в положение «включено».

:indeterminate – задаёт стиль для элементов форм, таким как флажки и переключатели, когда они находятся в неопределённом состоянии.

:disabled – применяет стиль к **заблокированным** элементам форм.

:enabled – используется для применения стиля к доступным (не заблокированным) элементам форм.

:required – применяет стилевые правила к элементу input, у которого установлен атрибут required.

:optional – применяет стилевые правила к полю формы, у которого не задан атрибут required.

:invalid – применяется к полям формы, содержимое которых не соответствует указанному типу.





# Селекторы. Псевдоклассы

**:empty** – представляет пустые элементы, т. е. те, которые не содержат дочерних элементов, текста или пробелов.

**:not** – задаёт правила стилей для элементов, которые не содержат указанный селектор. `input:not([type="submit"])`

**:first-of-type** – задаёт правила стилей для первого элемента в списке дочерних элементов своего родителя.

**:nth-of-type** – используется для добавления стиля к элементам указанного типа на основе нумерации в дереве элементов.

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

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

**:only-child** – применяется к дочерним элементам, только если он **единственный** у родителя.



# Селекторы. Псевдоклассы. Пример

CSS:

```
table {  
    border-spacing: 0;  
}  
td {  
    border: 1px solid #333;  
    padding: 3px;  
}  
td:nth-last-child(2n+1) {  
    background: #fofofo;  
}  
td:nth-child(1) {  
    background: #cfc;  
}
```

```
<table>  
  <tr> <td>&nbsp;</td><td>ИТех</td><td>СПО</td><td>КСХ</td></tr>  
  <tr> <td>Иванов</td><td>79</td><td>90</td><td>75</td> </tr>  
  <tr> <td>Петров</td><td>60</td><td>76</td><td>90</td> </tr>  
  <tr> <td>Сидоров</td><td>79</td><td>95</td><td>78</td> </tr>  
</table>
```

	ИТех	СПО	КСХ
Иванов	79	90	75
Петров	60	76	90
Сидоров	79	95	78



# Селекторы. Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов, не определённых в дереве элементов, а также **генерировать содержимое**, которого нет в исходном коде текста. В CSS3 псевдоэлементы обозначаются с двумя двоеточиями, чтобы синтаксис отличался от псевдоклассов.

Пример:

```
<head>
  <style>
    p::first-letter {
      font-family: "Times New Roman", Times, serif;
      font-size: 2em;
      color: red;
    }
  </style>
</head>
<body><p> Виды селекторов </p></body>
```

**В**иды селекторов





# Селекторы. Псевдоэлементы. Примеры

Псевдоэлемент `::after` работает совместно со свойством `content`, позволяет генерировать содержимое (наследует стиль от элемента, к которому он добавляется).

Псевдоэлемент `::selection` применяет стиль к выделенному пользователем тексту. В правилах стилей **для этого псевдоэлемента** допускается использовать следующие свойства: `color`, `background`, `background-color` и `text-shadow`.

```
р::selection {  
    color: #ff0;  
    background: #000;  
}  
.cite::after {  
    font-style: italic;  
    content: "(см. раздел 4.2)";  
}
```

При выделении этого текста он **изменит свой цвет.**

20-й Юбилейный международный  
молодежный форум(см. раздел 4.2)

```
<body><p> При выделении этого текста он изменит свой  
цвет.</p>20-й Юбилейный международный молодежный  
<span class="cite">форум</span></body>
```





# Селекторы. Псевдоэлементы. Пример

```
<ol>
  <li>Иванов</li>
  <li>Петров</li>
  <li>Сидоров</li>
</ol>
```

1) Иванов  
2) Петров  
3) Сидоров

CSS:

```
ol {
  list-style-type: none;
  counter-reset: item;
}
li:before {
  content: counter(item) ')';
  counter-increment: item;
}
```



# Селекторы. Пример. Нумерация строк таблицы

```
<style>
```

```
table {
```

```
  counter-reset: cnt;
```

```
}
```

```
tr {
```

```
  counter-increment: cnt;
```

```
}
```

```
.trnumber::before {
```

```
  content: counter(cnt); //"."
```

```
}
```

```
</style>
```

```
<table border="1" class="mytable">
```

```
  <tr><td><span
```

```
    class="trnumber">.</span></td><td>1.2</td></tr>
```

```
  <tr><td><span
```

```
    class="trnumber">.</span></td><td>2.2</td></tr>
```

```
  <tr><td><span
```

```
    class="trnumber">.</span></td><td>2.3</td></tr>
```

```
</table>
```

1.	1.2
2.	2.2
3.	2.3

Или, используя JavaScript:

```
var firstcells = document.querySelectorAll('.mytable tr td:first-child');
```

```
for (i = 0; i < firstcells.length; i++) firstcells[i].innerHTML = i + 1 + ".";
```



# Комбинация селекторов. **Вложенные** селекторы

Допускается произвольный уровень вложения. Следует различать дочерние селекторы – когда элемент является непосредственным потомком.

```
p b {  
  font-weight: bold;  
  color: red;  
}  
  
p > b {  
  color: green;  
}
```

**Текст без влияния стиля**

**Демонстрация использования  
дочернего селектора**

**Демонстрация использования  
вложенного селектора**

<div><b>Текст без влияния стиля</b></div>

<p><span><b>Демонстрация использования **вложенного** селектора</b></span></p>

<p><b>Демонстрация использования **дочернего** селектора</b></p>





# Комбинация селекторов. Родственные селекторы

Соседними (+) называются элементы веб-страницы, следующие непосредственно друг за другом в коде документа.

Родственные селекторы (~) применяются к элементам одного элемента-родителя.

```
h1 ~ p { color: red; }  
h1 + p { color: green; }
```

```
<h1>Заголовок</h1>
```

```
<p>Соседний селектор (абзац сразу после заголовка, h1 + p).</p>
```

```
<div><p>Просто абзац.</p></div>
```

```
<p>Родственный селектор (абзац имеет непосредственного общего предка с заголовком, h1 ~ p).</p>
```

## Заголовок

Соседний селектор (абзац сразу после заголовка, h1 + p).

Просто абзац.

Родственный селектор (абзац имеет общего предка с заголовком, h1 ~ p).





# @-правила

**@charset** – указать кодировку внешнего CSS-файла.

**@document** – установить стилевые правила на основе адреса документа.

**@font-face** – настройка шрифтов.

**@import** – импорт содержимого CSS-файла в текущую стилевую таблицу.

**@keyframes** – установить ключевые кадры при анимации элемента.

**@media** – указать тип носителя, для которого будет применяться указанный стиль. В качестве типов выступают различные устройства, например, принтер, коммуникатор, монитор и др.

**@viewport** – позволяет оптимизировать макет веб-страницы в зависимости от различных устройств и их размеров.

**@page** – указать значение полей при печати документа.

```
@page :first {  
    margin: 1cm;  
}
```



# @-правила. @charset

Команда @charset применяется для задания кодировки внешнего CSS-файла, например, если в CSS-файле используются символы национального алфавита. @charset должно указываться **первой строкой** в файле стилей.

Для внешней таблицы стилей браузер последовательно просматривает следующие пункты для определения кодировки таблицы стилей:

1. кодировка, используемая веб-сервером;
2. правило @charset;
3. атрибут charset элемента <link>;

Пример:





```
@charset "windows-1251";
```



# @-правила. @font-face

Правило @font-face позволяет определить настройки шрифтов, а также загрузить специфичный шрифт на компьютер пользователя. Шрифты могут быть в форматах TTF, OTF, EOT, SVG и WOFF.

```
// объявление шрифта
@font-face {
  font-family: 'Имя шрифта';
  src: url('путь_до_него');
}
// использование шрифта
p {
  font-family: 'Имя шрифта', Arial;
}
```

Имя	Размер	Тип
 americantext	1 КБ	Cascading Style Sheet Document
 americantextc	30 КБ	Файл шрифта TrueType
 americantextc.woff	18 КБ	Файл "WOFF"
 americantextc.woff2	14 КБ	Файл "WOFF2"



# @font-face. Поддержка форматов браузерами

Формат	Поддерживают браузеры
EOT	IE6+
WOFF	IE9+ FF 3.6+ 11+ Chrome 6+ 5.1+ Opera mobile 11+

Формат	Поддерживают браузеры
TTF и OTF	FF 3.5+ IE10+ Chrome 4+ 3.2+ Opera mobile 10+ iOS 4.2+ Android 2.2+
SVG	IE9+ Chrome 4+ 3.2+ Opera mobile 10+ iOS Android 3





# @font-face. Пример подключения шрифтов

```
@font-face {  
  font-family: 'American TextC';  
  src: local('American TextC'),  
       url('americantextc.woff2') format('woff2'),  
       url('americantextc.woff') format('woff'),  
       url('americantextc.ttf') format('truetype');  
}  
  
p {  
  font-family: 'American TextC', Arial;  
  font-size: 250%;  
}
```

<p> Как использовать Google Fonts API </p>

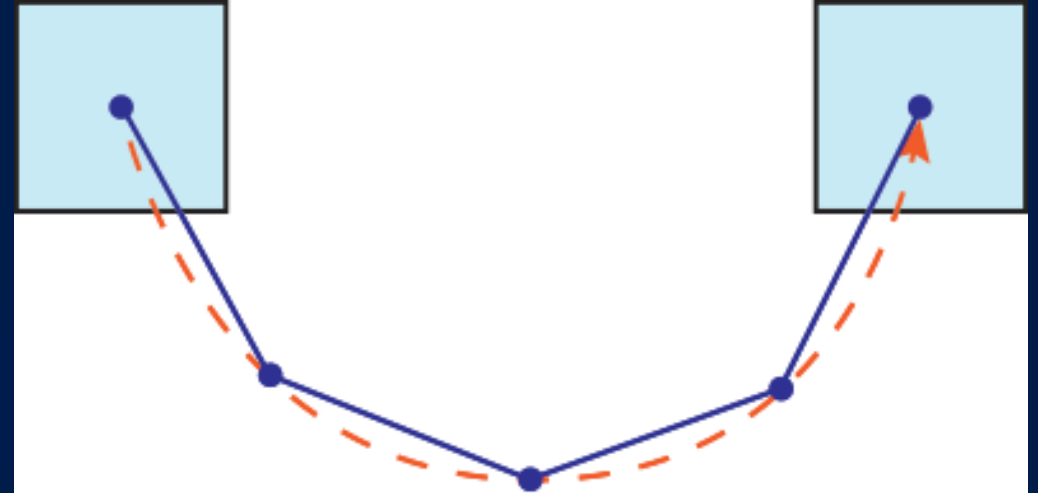


# @-правила. @keyframes

Правило @keyframes устанавливает ключевые кадры при анимации элемента. Ключевой кадр это **свойства элемента** (прозрачность, цвет, положение и др.), которые должны применяться к элементу в заданный момент времени. Таким образом, анимация представляет собой плавный переход стиливых свойств от одного ключевого кадра к другому. Вычисление промежуточных значений между такими кадрами берёт на себя браузер.

Пример:

```
@keyframes moving-box {  
  from { left: 0px; }  
  25% { left: 75px; top: 100px; }  
  50% { left: 150px; top: 200px; }  
  75% { left: 225px; top: 100px; }  
  to { left: 300px; }  
}
```



# @keyframes. Пример

```
<head>
<style>
  div {
    height:50px;
    width:50px;
    background: red;
    position: absolute;

    animation: diagonal-slide;
    animation-duration: 5s;
    animation-iteration-count: 10;
  }
```

```
@keyframes diagonal-slide {
  from {
    left: 0px;    top: 0px;    opacity: 0;
  }
  50% {
    top: 100px;    background: green;
  }
  to {
    left: 100px;    top: 100px;    opacity: 1;
  }
}
</style>
</head>

<body><div></div></body>
```

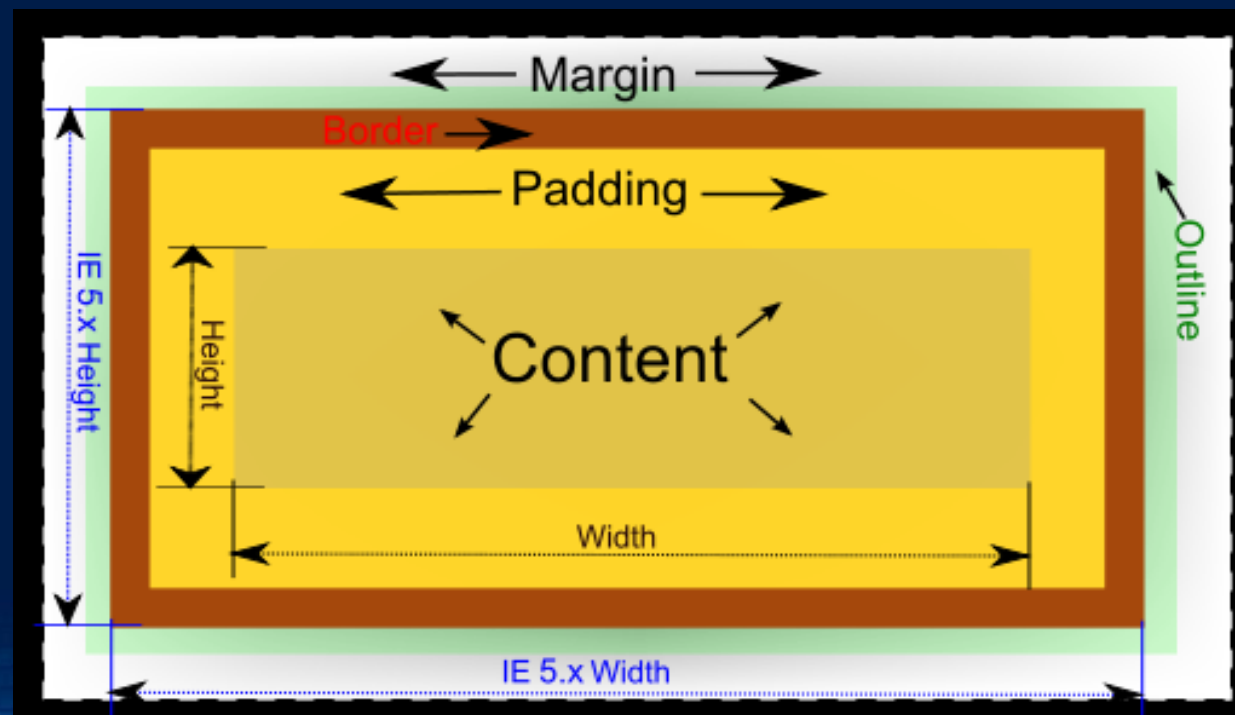




# Блочная модель

Определяет формирование размеров отдельных элементов на странице, и их взаимодействие между собой. Характеризуется следующими свойствами:

- border и outline – граница и внешняя граница; (второе свойство не участвует в формировании размеров блока)
- height и width – высота и ширина элемента.
- content – содержимое блока;
- padding – внутренний отступ;
- margin – внешний отступ;





# Позиционирование элементов

При помощи CSS-позиционирования вы можете разместить элемент точно в нужном месте страницы. Используются следующие свойства:

**position** – устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

**bottom** (или **left**, **right**, **top**) – устанавливают положение нижнего (левого, правого, верхнего) края содержимого элемента без учета толщины рамок и отступов.

**z-index** – задаёт положение элементов по z-оси.

```
h1 {  
  position: absolute;  
  top: 100px;  
  left: 200px;  
}
```



# Позиционирование элементов

Свойство **position** определяет способ позиционирования элемента относительно окна браузера, границ документа или других объектов на веб-странице. Возможны следующие значения:

**absolute** – 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.

**fixed** – в отличие от **absolute**, 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 исходного.

**static** – элементы отображаются как обычно. Использование свойств left, top, right и bottom не приводит к каким-либо результатам.

**inherit** – наследует значение родителя.



# Пример абсолютного и относительного позиционирования

```
<head>
<style>
.absolute_box {
  position: absolute;
  top: 50px;
  left: 50px;

  height: 50px;
  width: 100px;
  background: red;
}
```

```
.relative_box {
  position: relative;
  top: 0px;
  left: 100px;

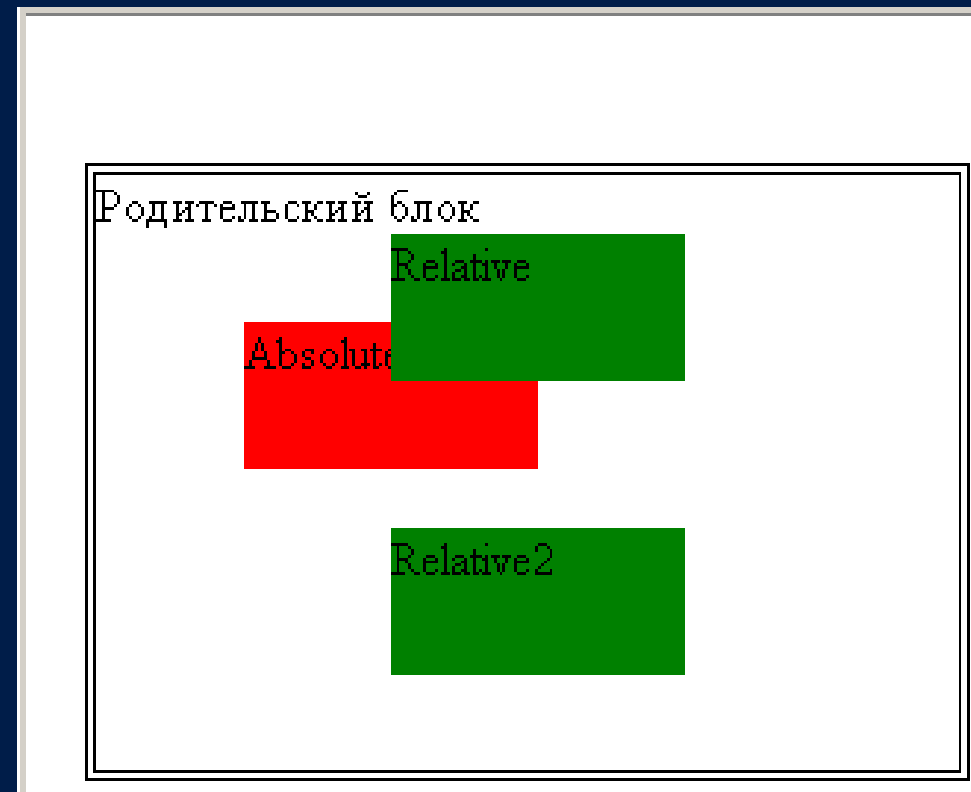
  height: 50px;
  width: 100px;
  background: green;
}
</style>
</head>
```





# Пример абсолютного и относительного позиционирования

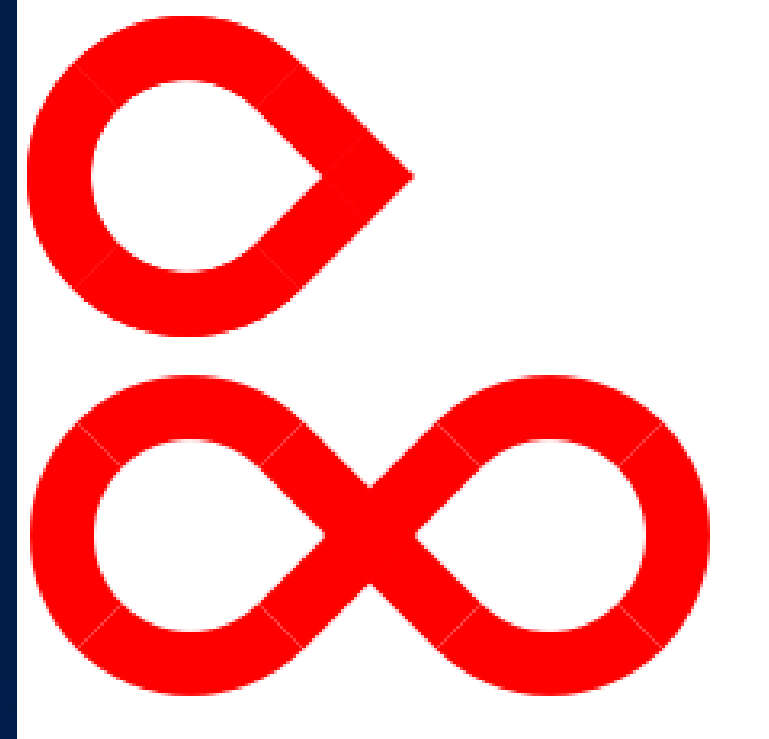
```
<body>
  <div style="position: absolute; top: 50px; left:
20px; border: 4px double black; height:50%;
width:50%;"> Родительский блок
    <div class="absolute_box"> Absolute</div>
    <div class="relative_box"> Relative</div>
    <div class="relative_box" style="top: 50px;">
Relative2</div>
  </div>
</body>
```





# Рисование. Пример

```
#infinity {  
  position: relative; width: 212px; height: 100px;  
}  
#infinity:before, #infinity:after {  
  content: ""; position: absolute;  
  top: 0; left: 0; width: 60px; height: 60px;  
  border: 20px solid red;  
  -moz-border-radius: 50px 50px 0 50px;  
  border-radius: 50px 50px 0 50px;  
  transform: rotate(-45deg);  
}  
#infinity:after {  
  left: auto; right: 0;  
  -moz-border-radius: 50px 50px 50px 0;  
  border-radius: 50px 50px 50px 0;  
  transform: rotate(45deg);  
}
```



# Позиционирование элементов. Свойство *float*

Свойство **float** определяет, по какой стороне будет выравниваться элемент, при этом остальные элементы будут обтекать его с других сторон.

Используются следующие значения:

**left** – выравнивает элемент по левому краю, а все остальные элементы, (например, текст), обтекают его по правой стороне.

**right** – выравнивает элемент по правому краю, а все остальные элементы обтекают его по левой стороне.

**none** – обтекание элемента не задаётся.

Свойство **clear** определяет, с какой стороны элемента запрещено его обтекание другими элементами. Если задано обтекание элемента с помощью свойства *float*, то *clear* отменяет его действие для указанных сторон.

`clear: none | left | right | both`



# Свойство *float*. Пример

```
.layer1 {  
  float: left;  
  background: #fdo;  
  border: 1px solid black;  
  padding: 10px;  
  margin-right: 20px;  
  width: 40%;  
}
```

Lorem ipsum  
dolor sit amet,  
consectetuer  
adipiscing elit,  
sed diam  
nonummy nibh  
euismod  
tincidunt ut  
laccet dolore  
magna aliquam  
erat volutpat.

Duis autem dolor  
in hendrerit in  
vulputate velit esse  
molestie  
consequat, vel  
illum dolore eu  
feugiat nulla  
facilisis at vero  
eros et accumsan  
et iusto odio  
dignissim qui  
blandit praesent  
luptatum zzril

delenit au gue duis dolore te feugiat nulla  
facilisi.





# Вопросы

- Что такое псевдоэлементы?
- В чем отличие директивы @media от применения аналогичного атрибута тега <link>?
- Каковы требования при использовании директивы @charset?
- Как подключить нестандартные шрифты?
- Для чего применяется директива @keyframes?
- В чем различие между абсолютным и фиксированным позиционированием?
- Назначение свойства float.

