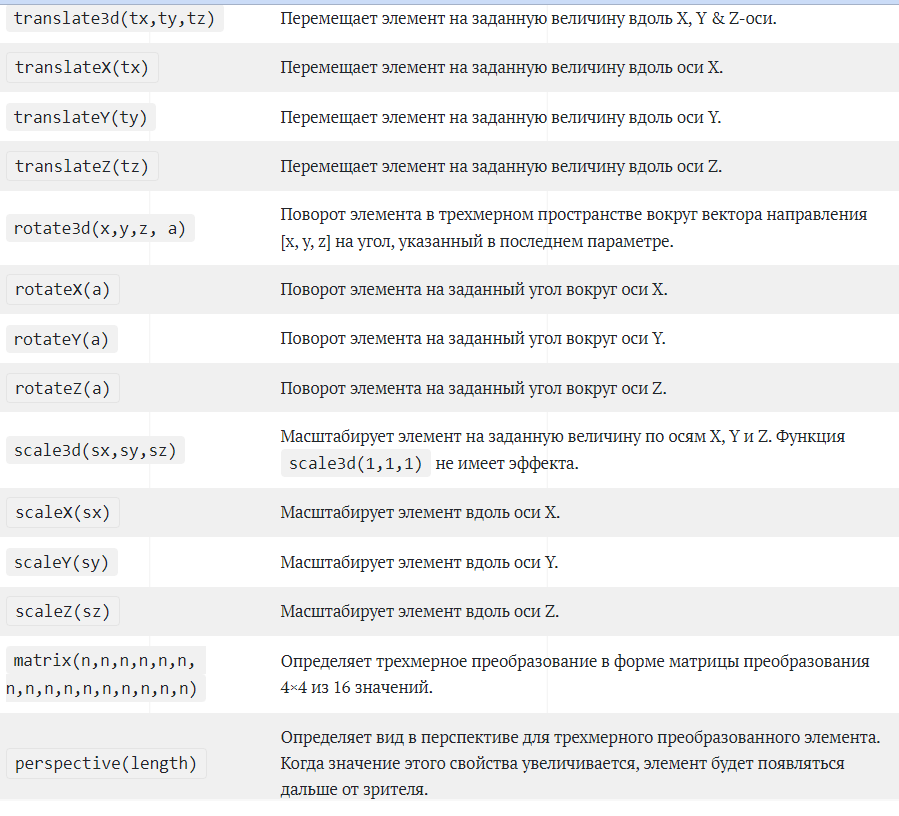
35. Функции 3D-преобразований.

Элементы HTML могут быть преобразованы в трех измерениях:

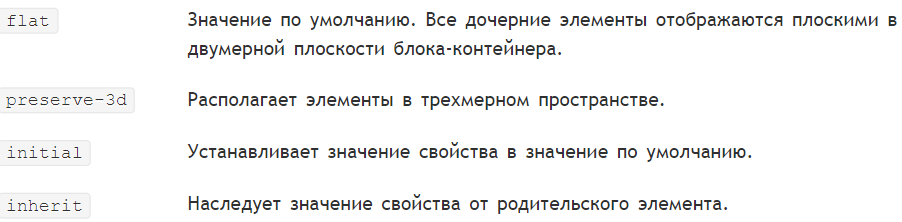
* горизонтальная ось X
* вертикальная ось Y и
* глубина по оси Z

1.свойство **transform** использует функции преобразования для управления системой координат, используемой элементом, чтобы применить эффект преобразования. **Значения:**

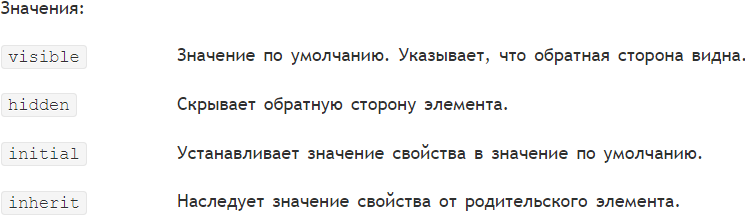


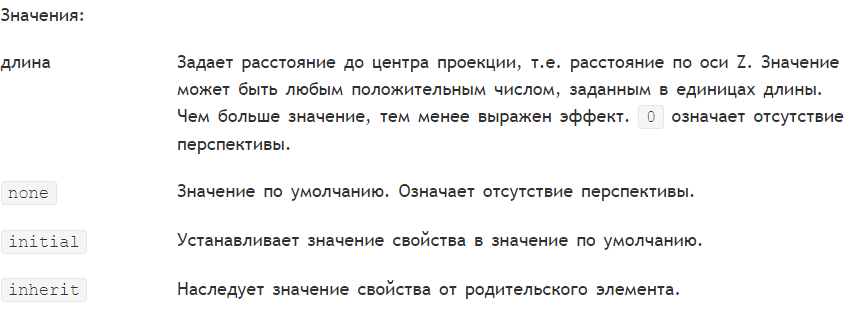
2.Свойство transform-style- позволяет преобразованным 3D-элементам и их 3D-потомкам использовать общее трехмерное пространство, выстраивая иерархи итрехмерных объектов. Значения:





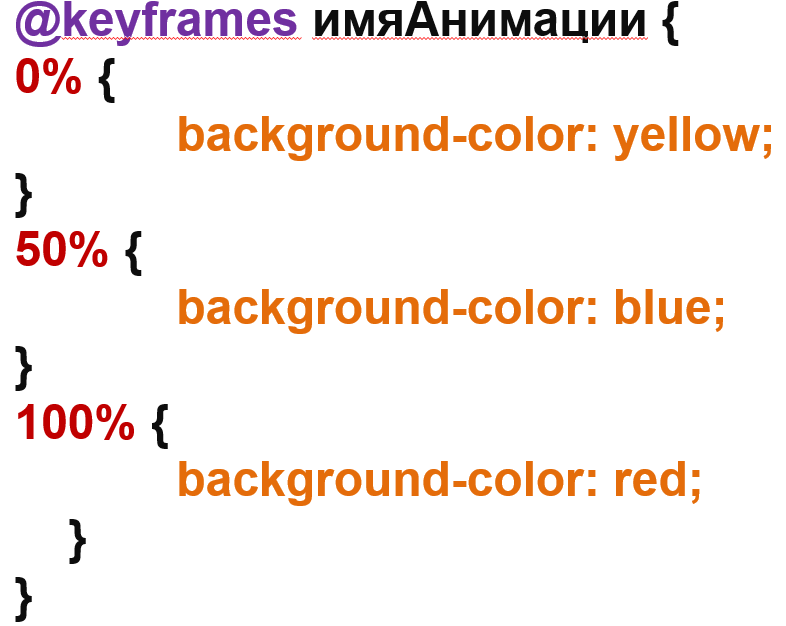
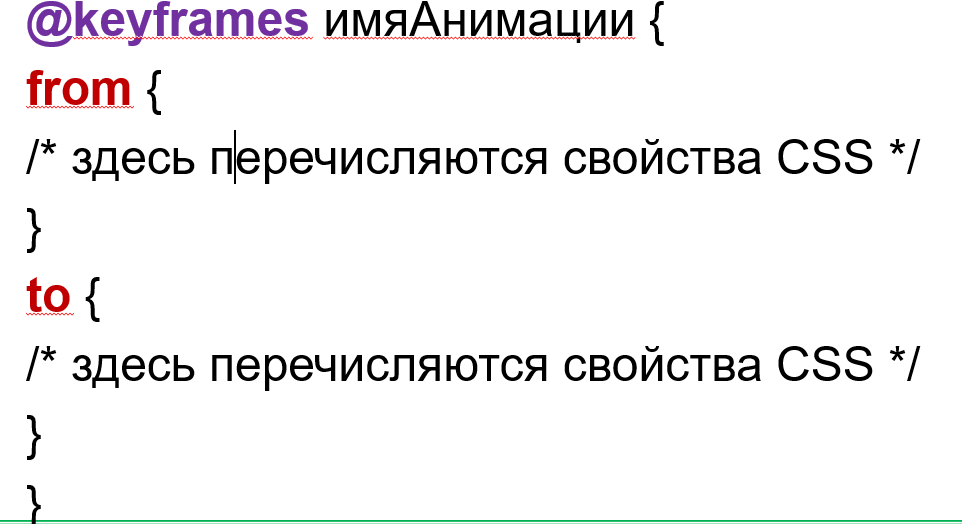
  3.Свойство backface-visibility позволяет делать элемент невидимым, когда его обратная сторона обращена к зрителю



4.Свойства perspective и perspective-origin можно использовать для добавления ощущения глубины в сцену, делая элементы выше по оси Z (ближе к зрителю) и кажущимися большими, а те, которые находятся дальше — меньшими 

36. CSS-анимация.

Анимация  — процесс перехода от одного набора свойств к нескольким наборам свойств



* Свойство **animation** используется для добавления анимации к элементу в сокращенной записи
* Свойство **animation-name** используется для добавления имени анимации к элементу, заданного в **@keyframes**
* Свойство **animation-timing-function** описывает плавность воспроизведения анимации между каждой парой ключевых кадров;
* Свойство **animation-delay** используется для задержки начала воспроизведения анимации
* Свойство **animation-iteration-count** устанавливает сколько раз анимация должна проигрываться
* Свойство **animation-direction** определяет воспроизведение вперед, назад или в случайной последовательности
* Свойство **animation-fill-mode** выясняет состояние анимации, когда она не воспроизводится
* Свойство **animation-play-station** определяет запущена анимация или приостановлена

37. Форматирование списков перечисления.

Списки бывают нескольких типов:

* маркированные списки;
* упорядоченные списки;
* списки определений.

### Списки определений

Это особый вид списков, который не имеет специальных символов перечисления. Содержит следующие теги:

* **<dl>** - список определений;
* **<dt>** - термин;
* **<dd>** - определение термина.

### Форматирование

##### HTML

Для упорядоченных списков **<ul>** существуют следующие виды форматирования:

* **type="circle"** - незаполненная окружность;
* **type="disc"** - заполненный круг;
* **type="square"** - заполненный квадрат.

Для нумерованных списков **<ol>** существуют следующие виды форматирования:

* **type="a"** - нумерация строчными латинскими буквами;
* **type="A"** - нумерация прописными латинскими буквами;
* **type="i"** - нумерация строчными римскими цифрами;
* **type="I"** - нумерация прописными римскими цифрами;
* **type="1"** - нумерация арабскими цифрами.

Нумерованные списки **<ol>** можно начинать с определенного значения:

* **start** - используется с тегом **<ol>**;
* **value** - используется с тегом **<li>** и устанавливает начальное значение списка;

##### CSS

Для списков предусмотрены следующие атрибуты CSS:

* **list-style-type** - вид маркера;
  + none - нет символов;
  + disc - незаполненная окружность;
  + circle - заполненная окружность;
  + square - заполненный квадрат;
  + decimal - нумерация арабскими цифрами;
  + lower-roman - нумерация строчными римскими цифрами;
  + upper-roman - нумерация прописными римскими цифрами;
  + lower-alpha - нумерация строчными латинскими буквами;
  + upper-alpha - нумерация прописными латинскими буквами;
* **list-style-position** - положение маркера относительно текста;
* **list-style-image** - адрес изображения, используемого в качестве маркера;
* **list-style** - позволяет задать значения всех трех или нескольких вышеперечисленных атрибутов одновременно

38. Системы модульной верстки.

# модульной сетки

Системы модульных сеток:

1. **Bootstrap-** свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.
2. **Skeleton-** это легкий CSS фреймворк, созданный [Дэйвом Гаманшем](https://twitter.com/dhg) (Dave Gamache). А точнее это два CSS файла: популярный файл normalize.css и сам файл skeleton.css. В последнем файле содержатся стили фреймворка, около 400 строчек кода в несжатом виде.
3. **Foundation-**  предоставляет адаптивную сетку и HTML и CSS компоненты пользовательского интерфейса, шаблоны и фрагменты кода, включая типографику, формы, кнопки, навигацию и другие элементы интерфейса, а также дополнительные функции, предоставляемые расширениями JavaScript.
4. **Pure** — это каскадная таблица стилей (CSS), разработанная YAHOO. Это помогает в создании более быстрых, красивых и отзывчивых сайтов

**5.Simple Grid-** Легкая CSS-сетка с 12 столбцами, которая поможет вам быстро создавать адаптивные веб-сайты.

Модульные сетка определяет двумерную разметку колонок и строк, в которые можно поместить в элементы. Страница по ширине делится на определенное количество столбцов модульной сетки, которые группируются для создания колонок контейнера. Некоторые элементы заполняют только одну ячейку сетки, а другие способны расширяться и размещаться в соседних колонках и строках.

Система модульной верстки основана на создании следующих элементов:

1. Контейнера. Контейнер **div** содержит одну или несколько строк.
2. Строки. Это также **div** элемент, помещенный в контейнер.
3. Колонки. Колонки определяются элементами **div** в строке.

39. Система модульной верстки Skeleton.

Система Skeleton — адаптивная система модульной верстки вебстраниц. Система позволяет сверстать сетчатый макет структурировав контент страницы в несколько колонок на планшетных устройствах, ноутбуках и компьютерах. На смартфоне содержимое страницы преобразуется в одноколоночный.

Чтобы начать работу, необходимо посетить сайт ***getskeleton.com*** и нажать кнопку Download. Загруженный архив содержит несколько папок и файлов. В каталоге css расположены необходимые файлы **normalize.css**, который сбрасывает базовые стили CSS, чтобы браузеры отображали HTMLэлементы одинаково, и файл ***skeleton.css***, содержащий для компоновки макета с использованием модульной сетки.

Система модульной верстки реализуется следующим образом:

1. Прикрепить CSS-файлы normalize.css и skeleton.css:

<link rel="stylesheet" href="css/normalize.css"> <link rel="stylesheet" href="css/skeleton.css">

1. Добавить контейнеры div.
2. Добавить контейнеры div для строк.
3. Добавить контейнеры div для колонок.

Система Skeleton основана на модульной сетке, состоящей из 12 столбцов, поэтому каждый из добавленных элементов div должен быть не менее одного или не более 12 (полная ширина контейнера) столбцов в ширину.

Например, чтобы создать три равные по ширине колонки, можно добавить три следующих элемента div:

<div class="container">

<div class="row">

<div class="four columns">

</div>

<div class="four columns">

</div>

<div class="four columns">

</div>

</div>

</div>

1. Добавить контент в элементы div колонок.

6.Создать собственные стили, например **custom.css**.

40. Структурирование HTML-кода под модульную сетку и ее использование. ????

# модульной сетки

Системы модульных сеток:

1. **Bootstrap-** свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.
2. **Skeleton-** это легкий CSS фреймворк, созданный [Дэйвом Гаманшем](https://twitter.com/dhg) (Dave Gamache). А точнее это два CSS файла: популярный файл normalize.css и сам файл skeleton.css. В последнем файле содержатся стили фреймворка, около 400 строчек кода в несжатом виде.
3. **Foundation-**  предоставляет адаптивную сетку и HTML и CSS компоненты пользовательского интерфейса, шаблоны и фрагменты кода, включая типографику, формы, кнопки, навигацию и другие элементы интерфейса, а также дополнительные функции, предоставляемые расширениями JavaScript.
4. **Pure** — это каскадная таблица стилей (CSS), разработанная YAHOO. Это помогает в создании более быстрых, красивых и отзывчивых сайтов

**5.Simple Grid-** Легкая CSS-сетка с 12 столбцами, которая поможет вам быстро создавать адаптивные веб-сайты.

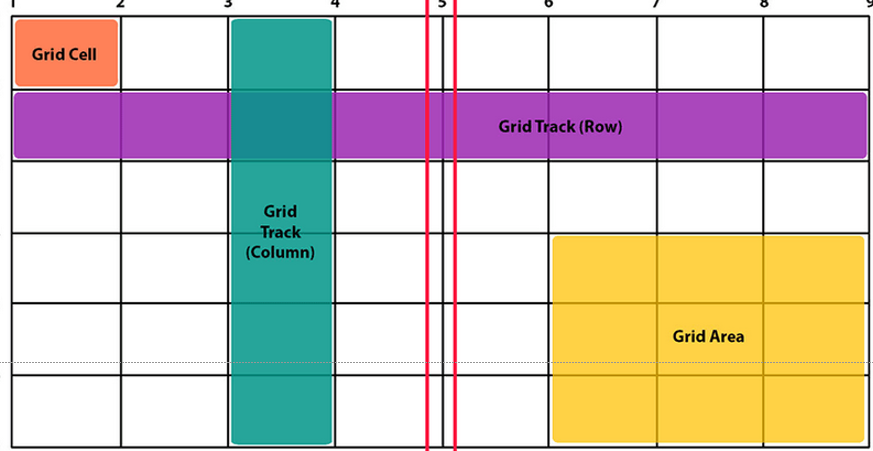
Модульные сетка определяет двумерную разметку колонок и строк, в которые можно поместить в элементы. Страница по ширине делится на определенное количество столбцов модульной сетки, которые группируются для создания колонок контейнера. Некоторые элементы заполняют только одну ячейку сетки, а другие способны расширяться и размещаться в соседних колонках и строках.

Система модульной верстки основана на создании следующих элементов:

1. Контейнера. Контейнер **div** содержит одну или несколько строк.
2. Строки. Это также **div** элемент, помещенный в контейнер.
3. Колонки. Колонки определяются элементами **div** в строке.

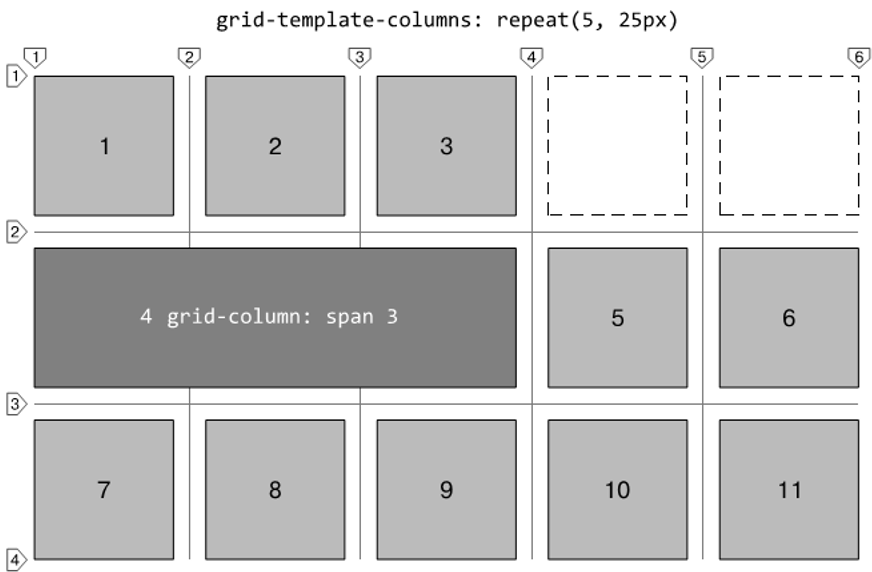
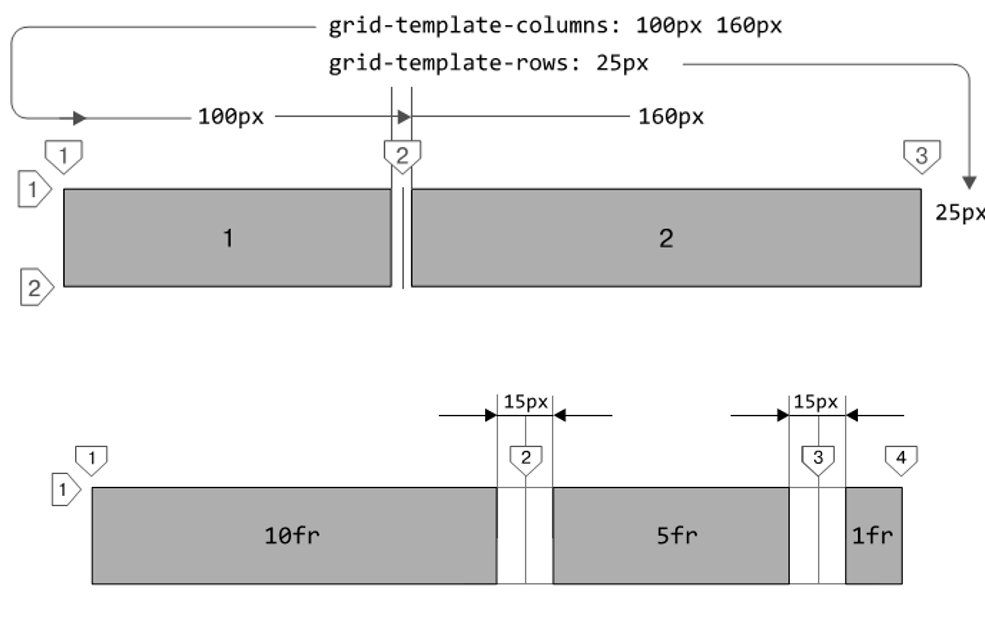
41. Основы Grid-верстки.

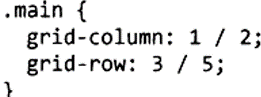
Grid-верстка определяет разметку с помощью разделения на колонки и строки для последующего добавления в нее элементов



**Свойства Grid-верстки:**

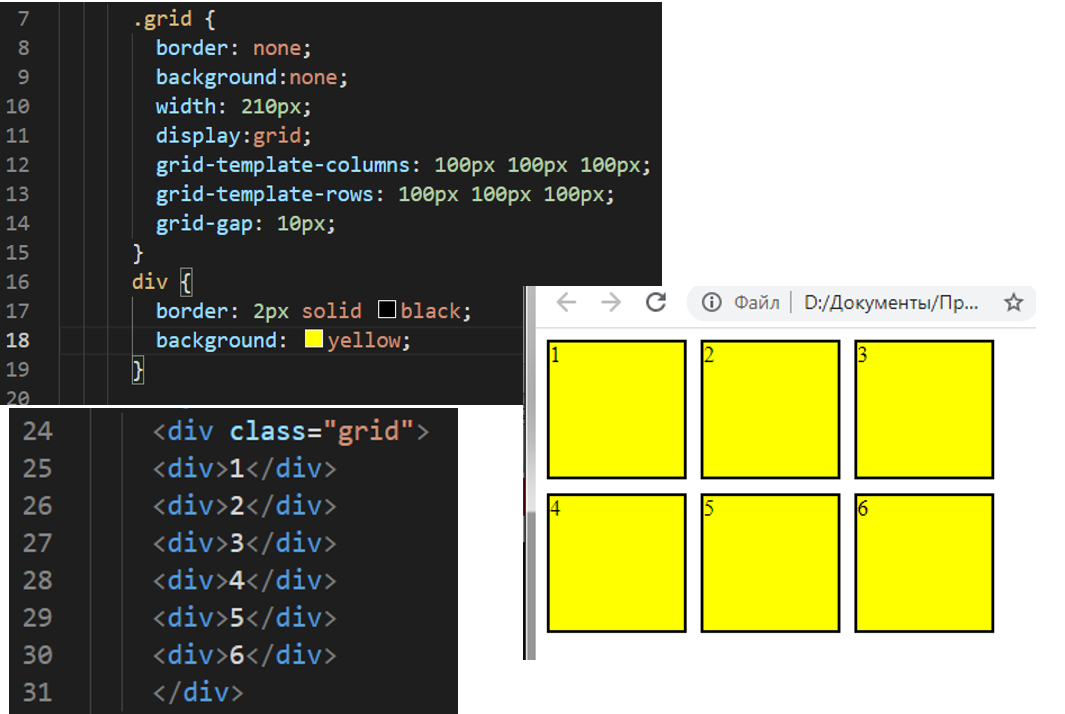
1. Свойство **display:grid;** преобразует HTML-элемент в контейнер сетки
2. Свойство **grid-template-columns** устанавливает размер каждой колонки
3. Свойство **grid-template-rows** устанавливает размер каждой строки
4. Свойство **grid-gap** устанавливает промежуток между ячейками сетки
5. Свойство **grid-auto-rows**   
   устанавливает автоматически размер каждой строки
6. Свойство **grid-auto-columns** устанавливает автоматически размер каждой колонки
7. Свойство **grid-auto –flow** определяет автоматически распределение элементов в сетке



1. Свойство **grid-template-areas** составляет сетку в зависимости от имени каждой ячейки
2. Свойство **grid-area** устанавливает имя для каждой ячейки каждой сетки
3. Свойство **grid-column** определяет местоположение элемента по номеру колонки
4. Свойство **grid-row** определяет местоположение элемента по номеру строки 

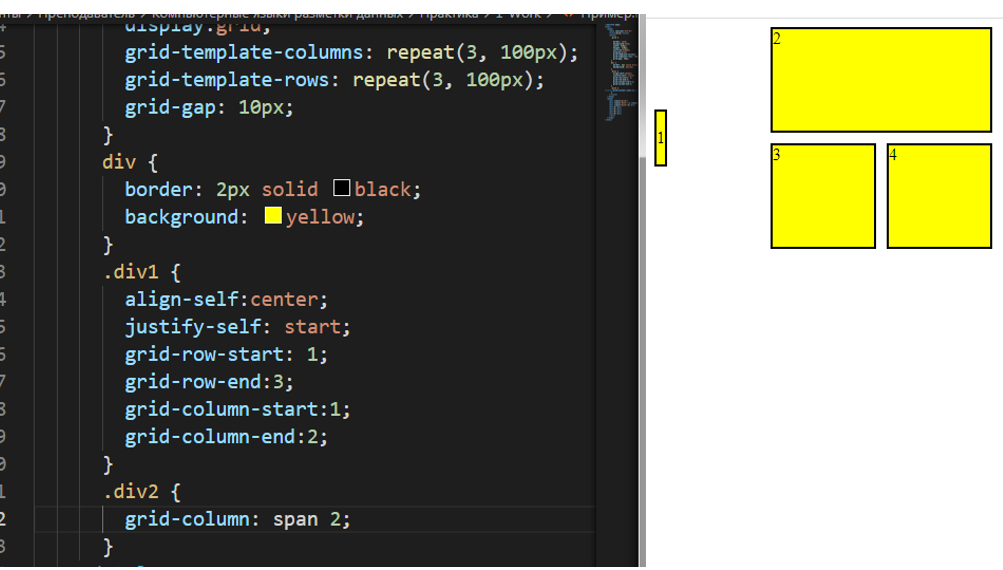
12.Свойство **align-self** определяет местоположение контента ячейки по высоте

13. **justify-self** определяет местоположение контента ячейки по горизонтали

Пример:







42. Медиазапросы. Создание адаптивного дизайна

Медиазапросы  — правила, которые позволяют назначать стили в зависимости от размера окна браузера.

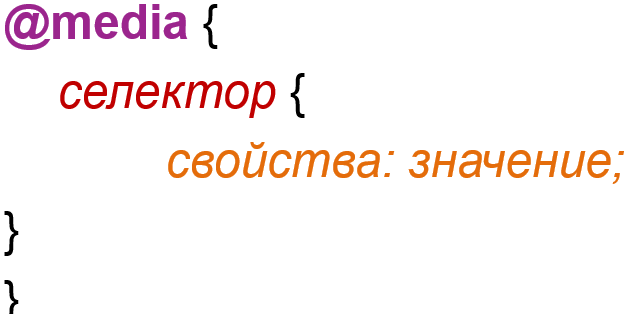
Они подключаются следующим образом:

1. Атрибут media тега link

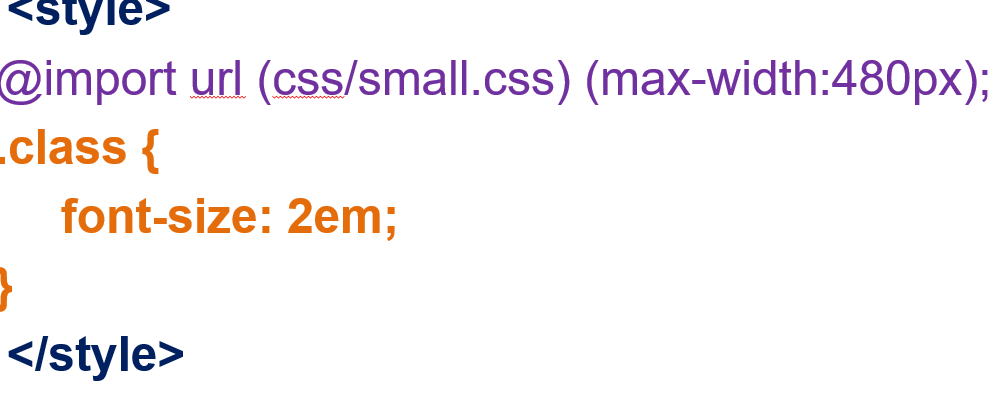
1. Правило @media

2. Правило @import

**Правило @media: Используются для внедрения в таблицу стилей медиазапросов**



**Правило @import: Используются для внедрения в таблицу стилей внешних стилей с медизапросами**



**Атрибут media тега link:**

*Пример 1*

**<link href="css/cssfile.css" rel="stylesheet" media="(width: 480px)">**

*Пример 2*

**<link href="css/cssfile.css" rel="stylesheet" media="(max-width:480px)">**

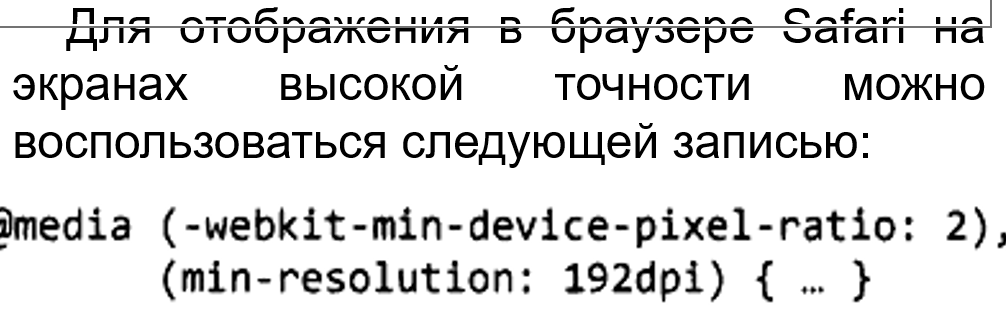
**Использование тега meta:**

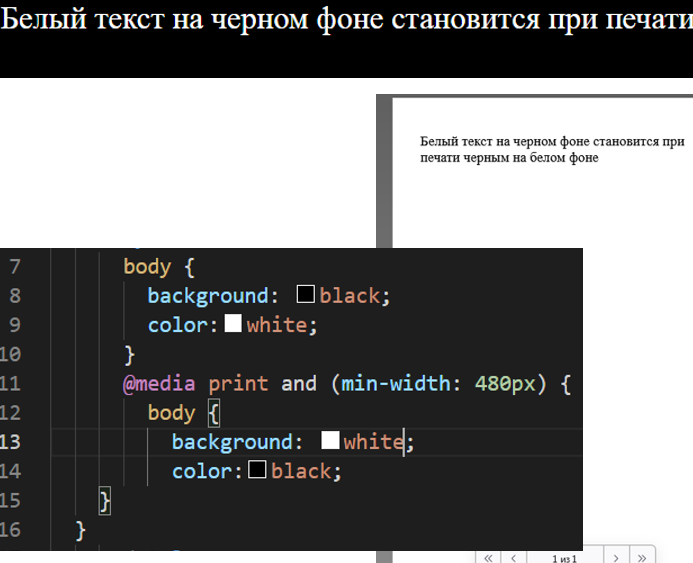
<meta name="viewport" content="width=device-width, initial-scale=1.0">

В примере ***viewport*** обозначает экран браузера, а для атрибута ***content*** устанавливается ширина окна браузера, равная разрешению по горизонтали экрана смартфона.

**Типы медиазапросов:**

* **@media (min-width: 20em) and (max-width: 35em)**
* **@media @media (max-width: 20em), (min-width: 35em)**
* **@media screen**
* **@media print**
* **@media speech**
* **@media (max-height: 20em)**
* **@media (orientation: landscape)**
* **@media (orientation: portrait)**
* **(min-resolution:** **2dppx**)





43. Основы Flexbox-верстки.

# Flexbox-верстка

**Flexbox-верстка** — разметка макетов веб-страниц, которая позволяет автоматически настраивать ширину элементов, находящихся внутри flex контейнера. Для применения данного способа разметки следует учитывать разделение применяемых свойств на свойства flex-контейнера и свойства flex-элемента.

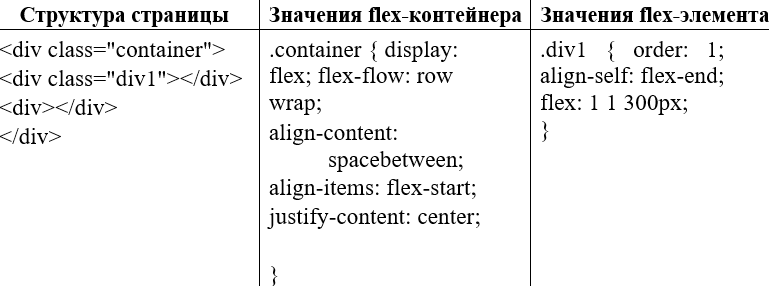
С помощью ***display: flex;*** осуществляется преобразование HTMLэлемента в flex-контейнер, а элементы находящийся внутри него в **flex элементы**. По умолчанию flex-элементы помещаются друг за другом в одной строке. Свойство ***flex-flow*** позволяет выбрать направление отображения элементов в контейнере, а также указать их перенос на следующую строку. Первое значение определяет направление, а второе перенос на следующую строку.

Свойство ***flex-flow*** является сокращенной запись и содержит значения свойств ***flex-direction*** и ***flex-wrap***. Flex-элементы могут размещаться в строке (значение **row**) или в колонке (значение **column**). Перенос элементов осуществляется с помощью значения **wrap**.

Свойство ***justify-content*** определяет способ выключки flex-элементов в строке. Для выравнивания левому краю используется значения **flex-start**, а по правому краю — **flex-end**. Для равномерного распределения элементов создавая пространство между ними необходимо значение **space-between**, а **space-around** добавляет поля по левому и правому краям крайних элементов.

Свойство ***align-items*** определяет, как flex-элементы различной высоты будут выровнены по высоте строки в flex-контейнере. Для выравнивания верхнему краю используется значения **flex-start**, а по нижнему краю — **flexend**. Значение **stretch** позволяет растянуть каждый элемент по высоте контейнера, делая их высоты одинаковыми. Свойство *align-content* определяет как будут размещены flex-элементы, занимающие несколько строк.

Для flex-элементов основным свойством являет *flex*, которое обеспечивает их гибкость и управляет шириной, что позволяет создавать «гибкие» колонки или изменять их ширину в соответствии с размером контейнера, даже если размер неизвестен или меняется динамически. **Первое значение свойства flex** — число параметра *flex-grow*, которое указывает на относительную ширину flex-элемента, которая определяет во сколько размеры элементов отличаются между собой. Второе значение — число свойства *flex-shrink*, которое определяет насколько flex-элемент может быть сжат, если суммарная ширина элементов больше ширины контейнера. Последнее значение — свойство *flex-basis*, которое определяет базовую ширину flex-элемента. Пример кода с возможным свойствами представлен в таблице



44. Основы Sass

**Sass** расшифровывается как *Syntactically Awesome Style Sheets* и переводится «синтаксически привлекательные таблицы стилей»

Препроцессор  — программа, имеющий собственный синтаксис, который затем компилируется в стандартный CSS-код

**Sass** — это препроцессор СSS, который представляет собой метаязык на основе CSS, предназначенный для расширения возможностей CSS и упрощения каскадных таблиц стилей

Преимущества Sass:

1. Совместимость с различными версиями CSS

2. Функциональность

3. Позволяет использовать конструкции из программирования для разработки кода CSS

4. Совместимость с CSS-фреймворками

5. Организация СSS-кода в файлы меньшего размера

6. Устранение трудностей связанных с обновлением свойств

7. Уменьшение размера кода

CSS/Sass-правила можно разделить на несколько файлов, а затем объединить их в один CSS-файл. Чтобы сообщить препроцессору Sass, что вы нехотите преобразовывать эти фрагменты в отдельные CSS-файлы, их имена должны начинаться с символа подчеркивания (\_)

Чтобы скомпилировать в итоговый файл определенный фрагмент используется:

@import ‘\_style.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

***ИЛИ***

@import 'first';

@import 'second';

@import 'third';

45. Основы Sass. Наследование свойств

**Sass** расшифровывается как *Syntactically Awesome Style Sheets* и переводится «синтаксически привлекательные таблицы стилей»

Препроцессор  — программа, имеющий собственный синтаксис, который затем компилируется в стандартный CSS-код

**Sass** — это препроцессор СSS, который представляет собой метаязык на основе CSS, предназначенный для расширения возможностей CSS и упрощения каскадных таблиц стилей

Преимущества Sass:

1. Совместимость с различными версиями CSS

2. Функциональность

3. Позволяет использовать конструкции из программирования для разработки кода CSS

4. Совместимость с CSS-фреймворками

5. Организация СSS-кода в файлы меньшего размера

6. Устранение трудностей связанных с обновлением свойств

7. Уменьшение размера кода

CSS/Sass-правила можно разделить на несколько файлов, а затем объединить их в один CSS-файл. Чтобы сообщить препроцессору Sass, что вы нехотите преобразовывать эти фрагменты в отдельные CSS-файлы, их имена должны начинаться с символа подчеркивания (\_)

Чтобы скомпилировать в итоговый файл определенный фрагмент используется:

@import ‘\_style.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

***ИЛИ***

@import 'first';

@import 'second';

@import 'third';

**Наследование:**

Преимущество sass- возможность наследовать свойства определенные для других селекторов. Чтобы воспользоваться такой возможностью, необходимо использовать ключевое слово **@extend**, после которого указать селектор, все свойства которого вы хотите унаследовать. Это позволяет сделать код более компактным.

Пример:

Sass:

|  |
| --- |
| h1{  border: 4px solid #ff9aa9 }  .speaker {  @extend h1  border-width: 2px } |

Что будет скомпилировано в:

|  |
| --- |
| h1,.speaker {  border: 4px solid #ff9aa9; } |

46. Установка Sass

Sass создается на языке Ruby. Перед использованием необходима установка среды разработки Ruby:

1. Загрузить установочный пакет Ruby с сайта rubyinstaller.org/downloads/

2. Запустите установочный файл.

3. После завершения установки нажать кнопку Finish(Готово) в последнем диалоговом окне

4. Запустить приложение командной строки

5. Ввести команду gem install sass

6. Ввести сd “Путь к папке с файлом .scss”

7. Затем ввести “D:”, т.е. диск, где находится папка с вашим файлом

8. Ввести команду sass --watch sass:css

47. Примеси Sass (миксины)

Миксины позволяют вам повторно использовать целые фрагменты SASS кода и при необходимости даже передавать им аргументы. Можно указывать их исходные значения.

Для того чтобы определение миксинов используйте специально зарезервированное в SASS ключевое слово **@mixin**, а после него укажите выбранное вами имя миксина. Если есть необходимость в аргументах, то перечислите их в круглых скобках. Значения, устанавливаемые для аргумента по умолчанию, указываются при определении миксина через двоеточие. (\*Другими словами, миксины представляют собой CSS функции.)

Процедура **использования миксинов** очень проста – после ключевого слова **@include** укажите имя нужного миксина и перечислите в скобках необходимые параметры, если таковые имеются.

пример:

|  |
| --- |
| @mixin border-radius($amount: 5px) /\* определяем миксин \*/  -moz-border-radius: $amount  -webkit-border-radius: $amount  border-radius: $amount  h1 /\* используем миксин \*/  @include border-radius(2px)  .speaker  @include border-radius |

Этот SASS после компиляции будет преобразован в CSS код:

|  |
| --- |
| h1 {  -moz-border-radius: 2px;  -webkit-border-radius: 2px;  border-radius: 2x; }  .speaker {  -moz-border-radius: 5px;  -webkit-border-radius: 5px;  border-radius: 5px; } |

Для заголовка h1 мы явно указали значение радиуса закругления углов границы, но для элемента с именем класса .speaker мы этого не делали, поэтому было взято дефолтное значение 5px.

48. Переменные Sass

Переменная  — атрибут, которому присвоено значение которое может изменяться

определять переменные, которые можно использовать в любом месте вашего Sass файла. Цвета, дефолтные значения, единицы, все это можно взять в переменную и использовать в дальнейшем. Переменная определяется так: **$название: значение**.

С переменными вы можете производить арифметические действия, такие как сложение и вычитание, при условии, что используемые для этого значения имеют один и тот же тип данных.

**Например, в файле .scss:**

**$font\_color: #083B91;**

h1 {

color: **$font\_color;**

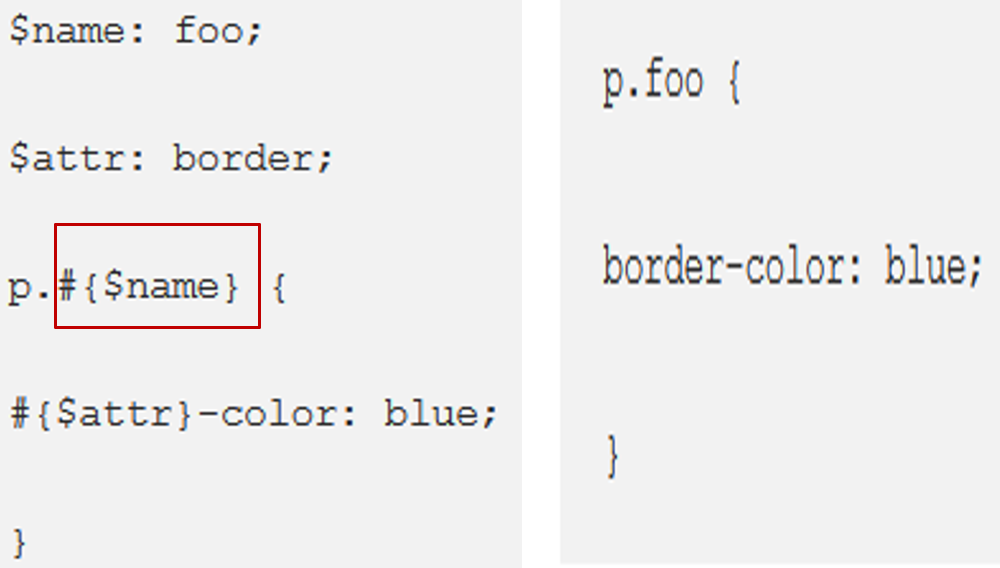
}

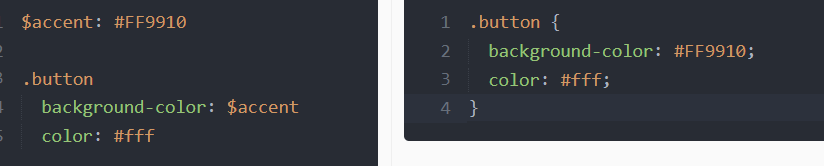
**В файле .css:**

h1 {

color: #083B91;

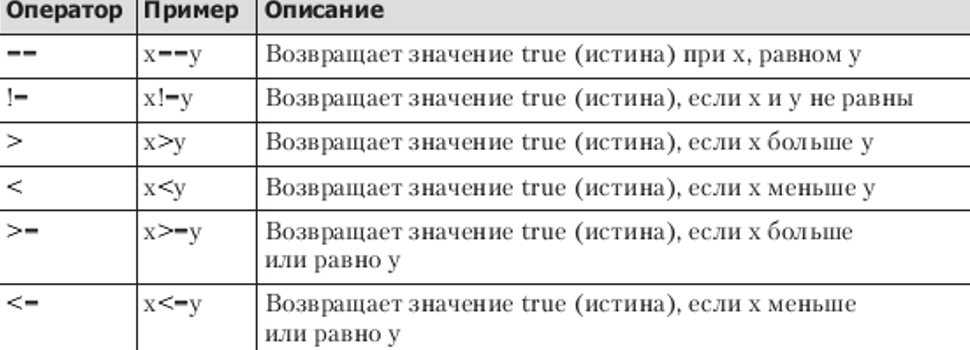
}





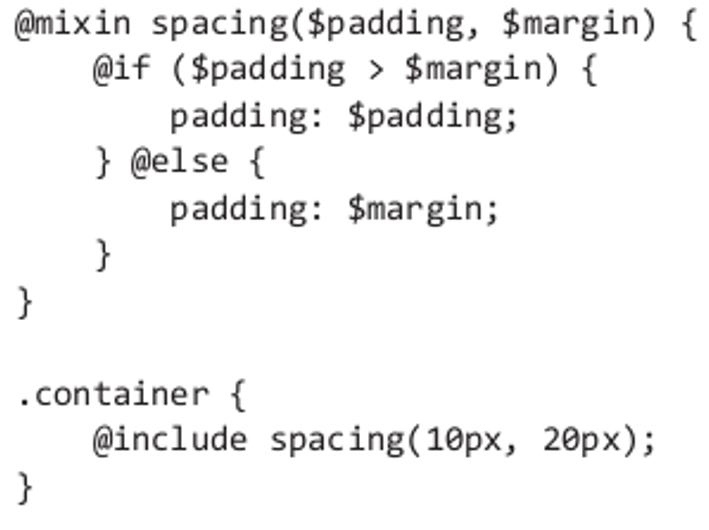
49. Операторы Sass

Использовать математику в CSS очень полезно. Sass имеет несколько стандартных математических операторов, таких как +, -, \*, / и %

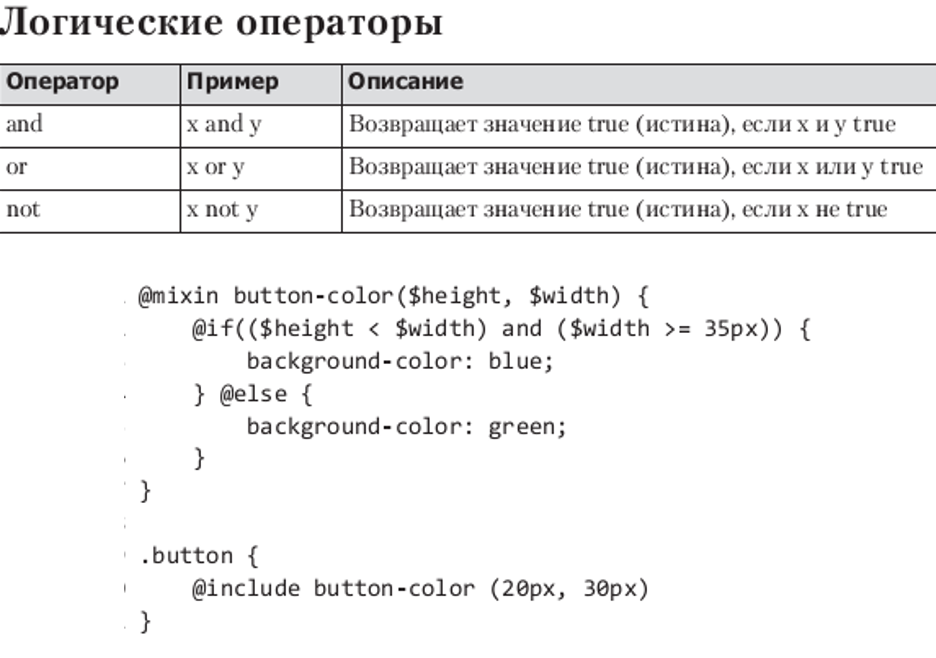


Операторы сравнения

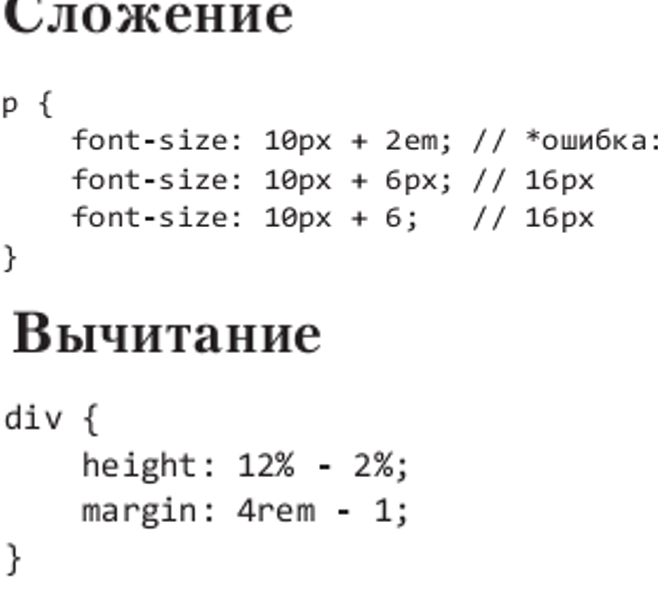
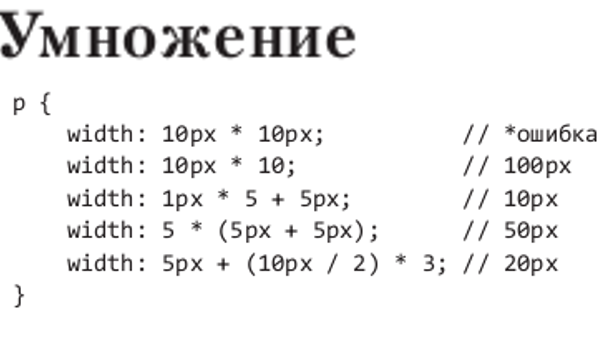
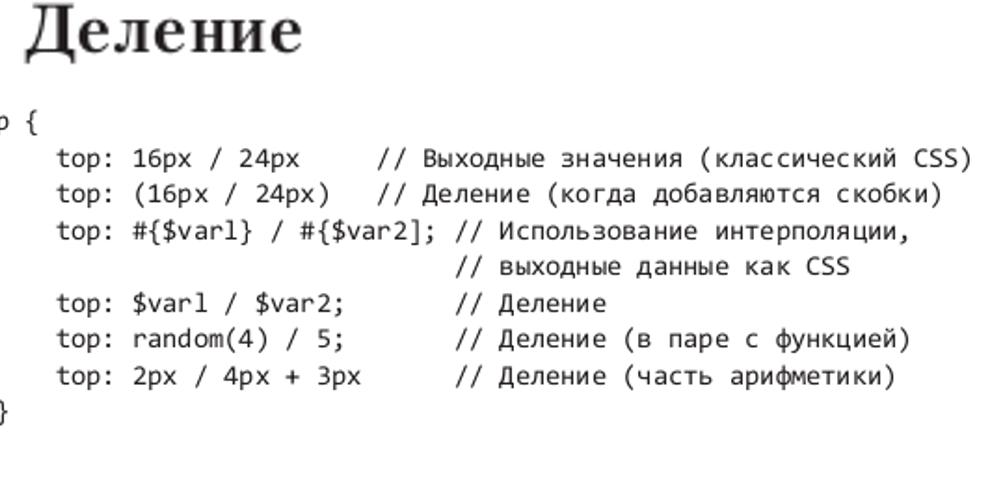
Директива @if принимает выражение SassScript и использует вложенные в неё стили в случае, если выражение возвращает любое значение, кроме false или null.



Использование логических операторов Sass для создания кнопки, у которой будет меняться фон в зависимости от её ширины.



**Арифметические операторы: сложение, вычитание, деление, умножение**

50. Основы Sass. Вложенные правила

**Sass** расшифровывается как *Syntactically Awesome Style Sheets* и переводится «синтаксически привлекательные таблицы стилей»

Препроцессор  — программа, имеющий собственный синтаксис, который затем компилируется в стандартный CSS-код

**Sass** — это препроцессор СSS, который представляет собой метаязык на основе CSS, предназначенный для расширения возможностей CSS и упрощения каскадных таблиц стилей

Преимущества Sass:

1. Совместимость с различными версиями CSS

2. Функциональность

3. Позволяет использовать конструкции из программирования для разработки кода CSS

4. Совместимость с CSS-фреймворками

5. Организация СSS-кода в файлы меньшего размера

6. Устранение трудностей связанных с обновлением свойств

7. Уменьшение размера кода

CSS/Sass-правила можно разделить на несколько файлов, а затем объединить их в один CSS-файл. Чтобы сообщить препроцессору Sass, что вы нехотите преобразовывать эти фрагменты в отдельные CSS-файлы, их имена должны начинаться с символа подчеркивания (\_)

Чтобы скомпилировать в итоговый файл определенный фрагмент используется:

@import ‘\_style.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

Если требуется создать одиночный CSS-файл с именем styles.css. Для этого можно использовать Sass-файл с именем styles.scss и поместить его в папку sass и он содержит

@import '\_first.scss';

@import '\_second.scss';

@import '\_third.scss';

***ИЛИ***

@import 'first';

@import 'second';

@import 'third';

### **Правила вложения**

В SASS вложенность бывает двух типов:

1. Вложенность селекторов.

Это первый тип вложенности, который схож с тем, какой используется для структурирования HTML кода:

$fontsize: 12px

.speaker

.name

font:

weight: bold

size: $fontsize + 10px

.position

font:

size: $fontsize

Путем вложенности класса .name в класс .speaker (\*тем же способом — с помощью двух пробелов в начале следующей строки) создается CSS селектор .speaker.name. Это же касается и следующего имени класса .position, расположенного после декларирования свойств для первого селектора, в результате вложенности которого образуется второй селектор .speaker.position:

.speaker.name {

font-weight: bold;

font-size: 22px; }

.speaker.position {

font-size: 12px; }

1. Вложенность свойств.

Второй тип вложенности позволяет структурировать свойства с одним префиксом (\* font-family, font-size, font-weight или border-style, border-color, border-radius и т.д.):

$fontsize: 12px

.speaker

.name

font:

weight: bold

size: $fontsize + 10px

.position

font:

size: $fontsize

В представленном выше примере мы используем объявление родительского, универсального свойства font: и далее на новой строке после двухпробельного отступа указываем вторую часть свойства, которая обычно находится после дефиса.

То есть если мы укажем свойство font:, на следующей строке после двух пробелов свойство weight:, то в результате это будет эквивалентно хорошо знакомому нам свойству font-weight.

.speaker.name {

font-weight: bold;

font-size: 22px; }

.speaker.position {

font-size: 12px; }

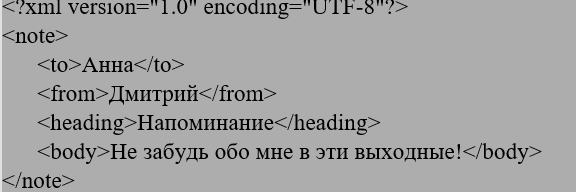
В модуле обеспечивается поддержка всех дефис-содержащих свойств.

Такого рода вложенность является замечательным способом организации и структурирования вашего CSS кода и позволяет избежать его нежелательного повторения.

51. Основы XML

XML (eXtensible Markup Language) — расширяемый язык разметки, созданный для хранения, транспортировки и обмена данными. Он включает в себя элементы SGML и предназначен для определения HTML-подобных языков. В нем отсутствуют элементы SGML, не применимые к языкам типа HTML, а другие элементы упрощены, чтобы облегчить их понимание и использование. В языке XML нет предопределенных тегов, автор определяет свои языковые теги и свою структуру документа. Формально XML представляет собой набор правил для создания собственных языков разметки, а также чтения и написания документов на языке разметки.

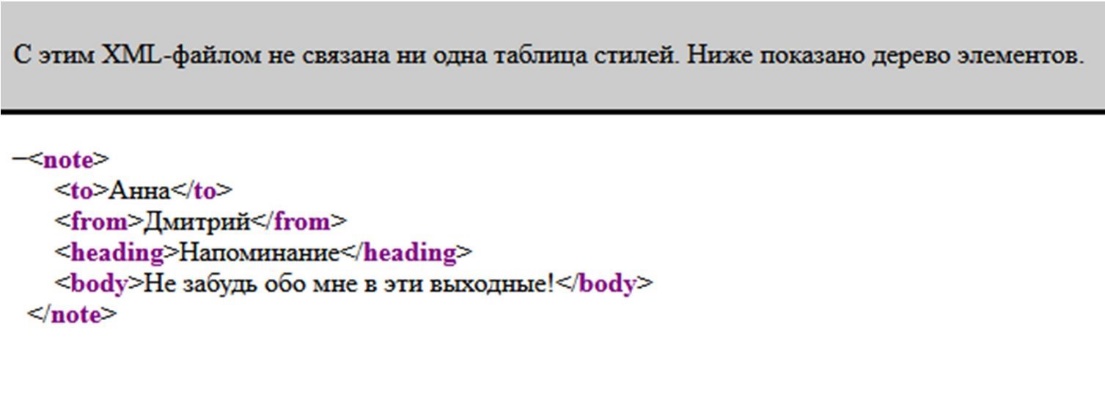
Пример XML-документа



Согласно примеру XML-документ состоит из содержимого, заключенного между тегов. Элемент состоит из своего начального и конечного тегов и всего, что находится между ними. Первые строки называются прологом и объявляют процессору XML, что данный документ размечен в XML. Пролог включает объявление (XML-декларация), которое начинается с разделителя, состоящего из пяти символов **<?xml**, за которым следует некоторое количество определений свойств, каждое из которых состоит из имени свойства и его значения, заключенного в кавычки. Объявление завершается закрывающим разделителем из двух символов **?>**. Варианты объявлений могут быть разные в зависимости от предпочтений пользователя:

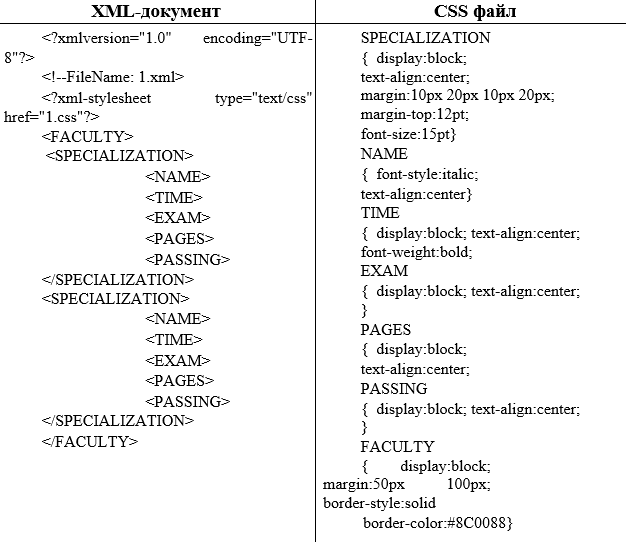
## <?xml version="1.0"?> <?xml version='1.0' encoding='UTF-8' standalone='yes'?> <?xml version = '1.0' encoding= 'UTF-8' standalone ="no"?>

Элемент документа <note> является корневым. Следующие 4 строки описывают дочерние элементы корневого элемента: <to>Анна</to>, <from>Дмитрий</from>, <heading>Напоминание</heading>, <body>Не забудь обо мне в эти выходные!</body>. Последняя строка определяет конец корневого элемента: </note>. При открытии его в браузерах, например Mozilla или Opera, будет отображаться следующим образом:



Таким образом, необходимо подключение таблицы стилей для создания конечного продукта после объявления XML-документа.

## <?xml-stylesheet type="text/css" href="style.css"?>



Также следует различать синтаксически верный документ и валидные документы. К синтаксически верным (корректным) относятся XMLдокументы, которые соответствуют правилам синтаксиса XML:

1. Все XML элементы должны иметь закрывающий тег.
2. Теги XML являются регистрозависимыми.
3. Перед закрывающей угловой скобкой в пустых элементах XML требуется ставить косую черту.
4. Значения должны быть заключены в одинарные или двойные кавычки.
5. Все элементы обязаны соблюдать корректную вложенность.
6. XML документ должен содержать один корневой элемент, который будет родительским для всех других элементов.
7. Учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).

52. Правила документов XML.

В общем случае XML- документы должны удовлетворять следующим требованиям:

* В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация
* Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника", т.е., в отличие от HTML, нельзя опускать закрывающие тэги
* В XML учитывается регистр символов
* Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки
* Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов
* Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования ( т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML)

Если XML- документ не нарушает приведенные правила, то он называется формально-правильным и все анализаторы, предназначенные для разбора XML- документов, смогут работать с ним корректно.

К синтаксически верным (корректным) относятся XMLдокументы, которые соответствуют правилам синтаксиса XML:

1. Все XML элементы должны иметь закрывающий тег.
2. Теги XML являются регистрозависимыми.
3. Перед закрывающей угловой скобкой в пустых элементах XML требуется ставить косую черту.
4. Значения должны быть заключены в одинарные или двойные кавычки.
5. Все элементы обязаны соблюдать корректную вложенность.
6. XML документ должен содержать один корневой элемент, который будет родительским для всех других элементов.
7. Учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).

В общем виде ХМL-документ после пролога можно представить в следующем виде:

## <корневой> <потомок> <подпотомок>.....</подпотомок> </потомок> </корневой>

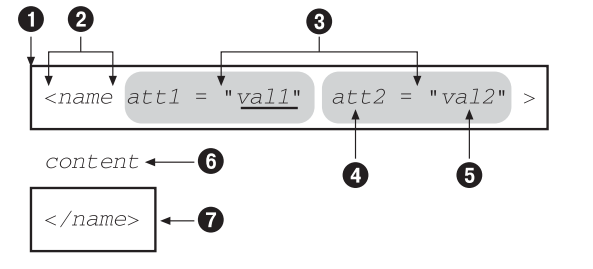
**Имена элементов могут начинаться** только с букв и символов подчеркивания и могут содержать только буквы, цифры, дефисы, точки и символы подчеркивания и не могут начинаться с сочетания «xml». В качестве имен можно использовать любые слова, зарезервированных слов нет.

53. Элементы и атрибуты XML

Элементы суть части документа. Можно разделить документ на части, которые будут выводиться различным образом или использоваться поисковым механизмом. Элементы могут быть контейнерами, содержащими смесь текста и других элементов.

Следующий элемент содержит только текст: 

Некоторые элементы являются пустыми, и информация, которую они несут, определяется положением и атрибутами.

Атрибут определяет свойство элемента и состоит из имени (4), соединяемого знаком равенства (=) со значением, заключенным в кавычки (5). У элемента может быть любое количество атрибутов. За открывающим тегом следует содержимое элемента (6), за которым 

Имя элемента должно начинаться с буквы или символа подчеркивания и может содержать любое число букв, цифр, дефисов, точек и символов подчеркивания.Имена элементов могут содержать латинские символы.

Есть два правила, касающиеся расположения открывающего и закрывающего тегов:

• Закрывающий тег должен помещаться после открывающего тега.

• Открывающий и закрывающий теги элемента должны иметь одного и того же родителя.

**Атрибут**

Иногда требуется передать об элементе больше информации, чем могут выразить его имя и содержимое. Атрибуты позволяют более ясно описывать детали элементов, например, для того чтобы дать элементу уникальную метку, по которой его можно легко найти, или описать свойство элемента, скажем, адрес файла, на который указывает ссылка. При помощи атрибута можно описать какую-либо сторону поведения элемента или создать подтип

У элемента может быть любое число атрибутов, но каждый из них должен обладать уникальным именем. Вот элемент с тремя атрибутами: 

**Некоторые имена атрибутов зарезервированы** рабочей группой XML для специальных целей – для использования в XML, и начинаются с префикса xml: пример:

1. **xml:attribute**, который позволяет «переназначить» эти специальные атрибуты
2. **xml:link** Сообщает процессору XLink, что элемент является элементом ссылки
3. **xml:space** Указывает, должны ли пробельные символы сохраняться в содержимом элемента.
4. **xml:lang** Классифицирует элемент по языку, на котором написано содержание

54. Стандарты XML????

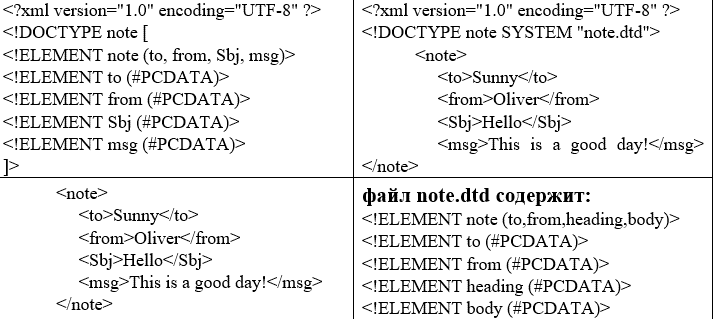
Стандарт XML можно разделить на три группы. В первую группу попадают базовые стандарты, с помощью которых формируются XML-документы. Такими спецификациями принято считать рекомендации консорциума W3C: XML 1.0 и XML 1.1, Namespaces in XML 1.0 и 1.1, Canonical XML, XLink (XML Linking Language), XML Base, XML Infoset, XML Schema Part 1: Structures и XML Schema Part 2: Datatypes, XPath, XPointer, а также стандарт OASIS RELAX NG.

Вторая группа - это стандарты, относящиеся к обработке XML-документов: XSLT, DOM, XQuery, SOAP, WSDL и некоторые другие.

Третий класс XML-спецификаций составляют XML-приложения, или словари: XHTML, Docbook, XSL-FO, SVG, VoiceXML, MathML, SMIL. Словари, основанные на XML, сами по себе формально описаны, что позволяет программно изменять и проверять документы на основе этих словарей, не зная их семантики, то есть, не зная смыслового значения элементов.

55. Cхема DTD

DTD (Document Type Definition, определение типа документа) — это язык описания структуры XML-документа, который используется для проверки граматики XML-документа и его соответствия определенному типу. Цель DTD состоит в том, чтобы определить структуру XML документа. Это делается путем определения списка допустимых элементов. В прологе декларация объявления элементов может быть внутренняя или внешняя

DTD в приведенном примере интерпретируется следующим образом:

1. **!DOCTYPE note** определяет, что корневым элементом документа является note
2. **!ELEMENT note** определяет, что элемент note содержит четыре

элемента: to, from, heading, body

1. **!ELEMENT to** определяет, что элемент to должен быть типа "#PCDATA"
2. **!ELEMENT from** определяет, что элемент from должен быть типа "#PCDATA"
3. **!ELEMENT heading** определяет, что элемент heading должен быть типа "#PCDATA"
4. **!ELEMENT body** определяет, что элемент body должен быть типа

"#PCDATA"

**#PCDATA** означает разбираемые текстовые данные. Кроме инструкции ELEMENT существуют следующие:

**— ATTLIST** для перечисления и объявления атрибутов, которые могут принадлежать элементу.

**— ENTITY** для определения сущностей в DTD с целью их использования как в связанном с DTD XML-документе, так и собственно в DTD.

|  |  |  |
| --- | --- | --- |
| <?xml version="1.0"?>  <!DOCTYPE note [  <!ENTITY name “Hello, world!”>  ]>  <element>&name;</element> | | |
| — **NOTATION** используется для определения нотаций. |  |

56. XML Schema

XML Schema, которая описывают структуру XML документа. XML документ, прошедший проверку по XML схеме, является "синтаксически верным" и "валидным".

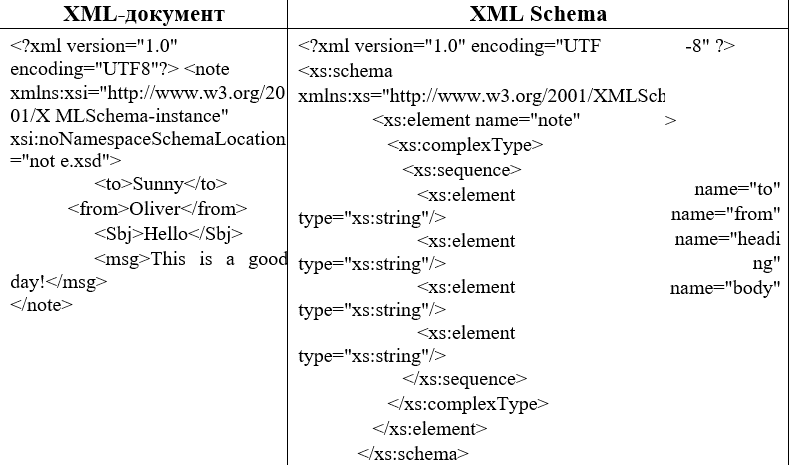
Кроме этого Schema имеет ряд преимуществ перед DTD

— XML схема пишется на XML;

— XML схема легко расширяется;

— XML схема поддерживает типы данных;

— XML схема поддерживает пространства имен;



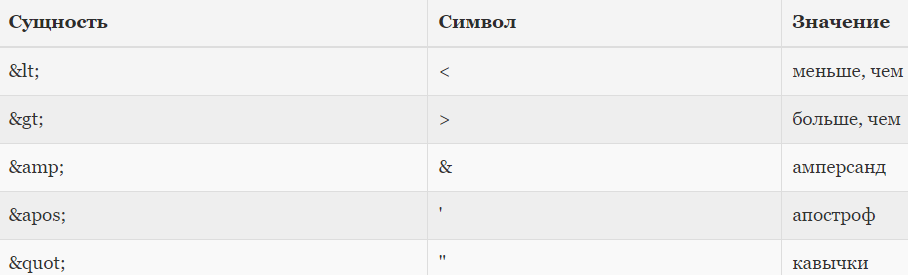
Приведенная выше схема интерпретируется следующим образом:

1. **<xs:element name="note">** определяет элемент "note";
2. **<xs:complexType>** у элемента "note" комплексный тип;
3. **<xs:sequence>** последовательность элементов элемента ;
4. **<xs:element name="to" type="xs:string">** у элемента "to" строковый тип (текст);
5. **<xs:element name="from" type="xs:string">** у элемента "from" строковый тип;
6. **<xs:element name="heading" type="xs:string">** у элемента "heading" строковый тип;
7. **<xs:element name="body" type="xs:string">** у элемента "body" строковый тип.

Как видно из примера, каждая XML схема состоит с корневого элемента «schema» и обязательного пространства имен

**«http://www.w3.org/2001/XMLSchema»,** а далее идет описание схемы.

57. Сущности XML

Сущность является заместителем содержания, которую можно однажды объявить и многократно использовать почти в любом месте документа Встроенные сущности:



ENTITY для определения сущностей в DTD с целью их использования как в связанном с DTD XML-документе, так и собственно в DTD.

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE note [  <!ENTITY name “Hello, world!”>  ]>  <element>&name;</element> |

58. Язык преобразований XSLT

**XSLT (eXtensible Stylesheet Language Transformations)** — это декларативное описание преобразования (трансформации) любого XML документа. Спецификация XSLT входит в состав XSL и является рекомендацией W3C.

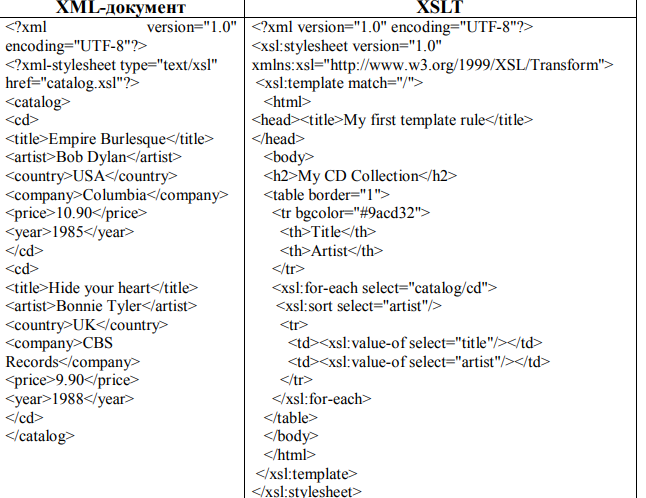
Существует **три основных способа преобразования XML-документов с помощью XSLT** в другие форматы, например, в HTML:

1. XML-документ и связанная с ним таблица стилей отправляются клиенту (веб-браузеру), который преобразует документ как указано в таблице стилей и затем предоставляет результат преобразования пользователю.

2. Сервер применяет таблицу стилей XSLT к XML-документу и преобразует его в другой формат (обычно, в HTML). После этого результат отправляется клиенту (веб-браузеру).

3. Какая-то программа преобразует оригинальный XML-документ в другой формат (обычно, в HTML), затем результат помещается на сервер. Таким образом, сервер и клиент имеет дело с преобразованным документом

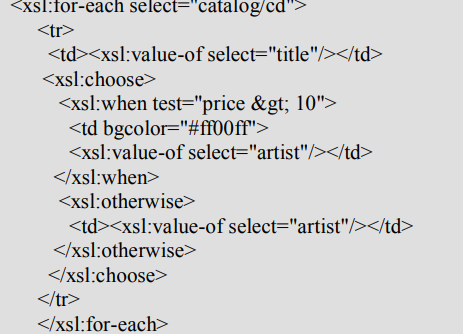
При помощи XSLT можно добавлять/удалять элементы и атрибуты в конечный файл. Также, можно реорганизовывать и сортировать элементы, выполнять тесты, определять, какие элементы скрыть или отобразить, и т. п. Таблица стилей XSL содержит один или больше наборов правил преобразования, которые называются шаблонами преобразования. Шаблон преобразования содержит правила, которые применяются, когда найден узел (элемент, атрибут, текст, комментарий), соответствующий условию поиска.

 Приведенный пример трактуется следующим образом:

1. **<xsl:stylesheet>** определяет, что данный документ является таблицей стилей XSLT с атрибутами номера версии и пространства имен XSLT.
2. **<xsl:output>**выбирает HTML как выходной формат,
3. **<xsl:template>** указывает, как должны преобразовываться части документа XML. Значение «/» атрибут match используется, чтобы определить шаблон для всего XML документа целиком.
4. **<xsl:value-of>** используется для извлечения значения отобранного XML элемента и добавления его в выходной поток преобразовываемого документа.
5. **<xsl:for-each>** может использоваться для выбора каждого XML элемента заданного узлового набора
6. **<xsl:sort>** используется для сортировки выходных данных и располагается внутри элемента <xsl:for-each>

Также могут быть использованы следующие элементы:

1. **<xsl:choose>** используется вместе с элементами <**xsl:when>** и **<xsl:otherwise**>, **чтобы определить проверку на выполнение условия**



**<xsl:apply-templates>** применяет некий шаблон к текущему элементу или к дочернему узлу текущего элемента

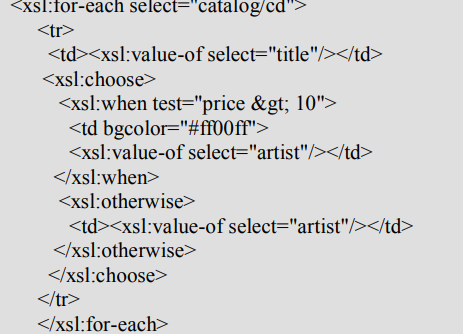
**К атрибутам элемента <xsl:sort> относятся**:

* **select** — обязательный атрибут, значением которого является выражение, называемое также ключевым выражением. Это выражение вычисляется для каждого узла обрабатываемого множества, преобразуется в строку и затем используется как значение ключа при сортировке.
* order – необязательный атрибут, определяет порядок, в котором узлы должны сортироваться по своим ключам.
* lang – необязательный атрибут, определяет язык ключей сортировки. Если значение этого атрибута не определено, процессор может либо определять язык исходя из параметров системы, либо сортировать строки исходя из порядка кодов символов Unicode;
* data-type – необязательный атрибут, определяет тип данных, который несут строковые значения ключей.

59. Сортировка данных в XML

Также могут быть использованы следующие элементы:

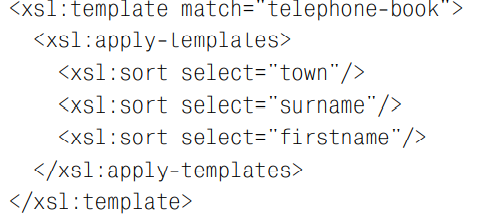
1. **<xsl:choose>** используется вместе с элементами <**xsl:when>** и **<xsl:otherwise**>, **чтобы определить проверку на выполнение условия**



**<xsl:apply-templates>** применяет некий шаблон к текущему элементу или к дочернему узлу текущего элемента

**К атрибутам элемента <xsl:sort> относятся**:

* **select** — обязательный атрибут, значением которого является выражение, называемое также ключевым выражением. Это выражение вычисляется для каждого узла обрабатываемого множества, преобразуется в строку и затем используется как значение ключа при сортировке.
* order – необязательный атрибут, определяет порядок, в котором узлы должны сортироваться по своим ключам.
* lang – необязательный атрибут, определяет язык ключей сортировки. Если значение этого атрибута не определено, процессор может либо определять язык исходя из параметров системы, либо сортировать строки исходя из порядка кодов символов Unicode;
* data-type – необязательный атрибут, определяет тип данных, который несут строковые значения ключей.

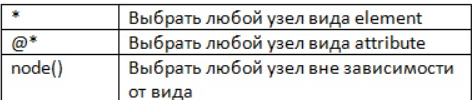
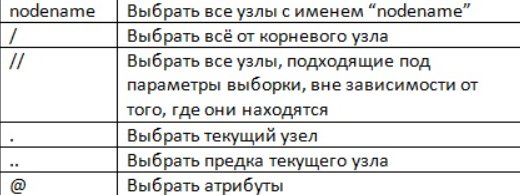


Здесь есть три направления сортировки. Сначала все результаты сортируются по городу (town). Затем записи сортируются по фамилии (surname). Наконец, они сортируются по имени (firstname)

60. Язык переадресации XPath

Xpath — это язык запросов к элементам xml или xhtml документа. Также как SQL, xpath является декларативным языком запросов. Чтобы получить интересующие данные, необходимо всего лишь создать запрос, описывающий эти данные.

При выполнении запросов язык XPath оперирует такими сущностями как узлы. Узлы бывают **нескольких видов: element** (узел-элемент), **attribute** (узел-атрибут**), text** (узел-текст**), namespace** (узел-пространство имён), processing-instruction (узел-исполняемая инструкция), **commen**t (узел-комментарий), **document** (узел-документ).

остроение правила выборки на каждом шаге выборки осуществляется относительно текущего узла и учитывает:

* Название оси, относительно которой следует производить выборку
* Условие выборки узла по имени или по положению
* Ноль или более предикатов

В общем случае синтаксис одного шага выборки имеет вид:  

61. Язык запросов XQuery

62. Введение в XLinks. Применение XPointer

XLinks (XML Linking Language) — стандарт, который определяет правила создания гиперссылок в XML. Сравнивание создания ссылок с помощью XLinks и в HTML можно выделить следующие преимущества:

1. Любой элемент XML может быть преобразован в ссылку. Это отличие от HTML, в котором существует ограниченное количество элементов определяющих гиперссылки.
2. XLinks использует язык указателей XML (XPointer) для создания ссылок на элементы внутри документа.
3. XML может использовать XLinks для импорта текста и разметки.
4. Указатели XPointer могут определять диапазон разметки XML для ссылки на подмножество документа.

Любой элемент XML можно преобразовать в ссылку при помощи определенных атрибутов XLink: **type, href, role, title, show и actuate**. Применяя эти атрибуты, нужно использовать пространство имен, поставленный в соответствие URI XLink. Процессор XML использует пространство имен для интерпретации этих атрибутов как параметров ссылки.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <university xmlns:xlink="http://www.w3.org/1999/xlink">  <faculty xlink:type="simple" xlink:href=" https://it.belstu.by/ " xlink:show="embed"  xlink:actuate="onLoad">ИТ</faculty>  <faculty xlink:type="simple" xlink:href="https://pim.belstu.by"  xlink:show="new"  xlink:actuate="onRequest">ПиМ</faculty>  </ university > |

Атрибут **type** определяет тип ссылки. Значение *simple* создает простую "HTML-подобную" ссылку, значение *resource —* ссылка на внутренний ресурс, значение *none* — элемент не имеет смысла, определяемого XLink.

Атрибут **href** определяет URL ссылки.

Атрибут **show** определяет где открывать ссылку. Значение *embed* указывает, что ресурс загружается в существующий документ; значение *replace* — документ заменяется на ресурс в том же окне; значение *new* — открыть новое окно для вывода содержимого удаленного ресурса, не прекращая просмотр текущего документа.

Атрибут **actuate** указывает, в какое время осуществлять загрузка ресурса и его отображение. Значение *onLoad* объявляет, что ссылка должна быть загружена и показана сразу после загрузки документа, значение *onRequest* — содержимое ссылки загружается и показывается только при нажатии на ссылку.

Браузеры не имеют поддержки XLink в XML документах. Тем не менее, все основные браузеры поддерживают простые XLink в SVG.

|  |
| --- |
| <body>  <svg width="300" height="200" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  <a xlink:href="image.jpg"><circle r="50" cx="75" cy="75"  fill="green"/></a>  </svg>  </body> |

Рис. 13.1

<body>

<svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="400" height="400" viewBox="0 0 400 400">

<symbol id="rect">

<rect x="50" y="50" width="100" height="100" rx="15"/>

</symbol>

<use xlink:href="#rect" style="fill:purple;" />

<use xlink:href="#rect" x="150" style="fill:gold;" />

</svg>

</body>

Рис. 13.2

Для создания ссылки из svg-элемента используется XLink (рис. 13.1) в элементе <a>. На рис. 13.2 гиперссылка позволяет получить доступ к созданным элементам, создав при этом копию элемента.

# Использование XPointer

XML Pointer Language — язык указателей XML. Указатель XPointer действует просто как идентификатор фрагмента, осуществляя связывание с элементом внутри целевого ресурса, имеющим атрибут ID. Однако XPointer более гибок, поскольку его целью может быть любой элемент, в отличие от HTML, где целью всегда является элемент <а>. Пример использования XPointer представлен на рис. 13.1 и рис. 13.2:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <university>  <faculty id="ISiT">  <name>ИТ</name>  <kafedra>ИСиТ</kafedra> |
| </faculty>  <faculty id= "PP">  <name>факультет ПиМ</name>  <kafedra>ПП</kafedra>  </faculty>  </university> |

Рис. 13.3

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <mykafedra>  <myfaculty>  <mykafedra xlink:type="simple"  xlink:href="http://univer.com/kafedra.xml***#ISiT***">ИСиТ</mykafedra>  <description>…</description>  </myfaculty>  <myfaculty>  <mykafedra xlink:type="simple"  xlink:href="http://univer.com/kafedra.xml***#PP***">ПП</mykafedra>  <description>…</description>  </myfaculty>  </ university> |

Рис. 13.4

На рис. 13.1 с помощью id создается фрагменты, гиперссылку на которые можно осуществить в другом документе, представленном на рис. 13.2, где определенную часть страницы, нужно добавить символ # и имя id после URL в атрибуте ***xlink:href.***

Браузеры не поддерживают XPointer, однако XPointer используется в других XML языках программирования.

63. Основы SVG

Масштабируемая векторная графика (Scalable Vector Graphics, SVG) представляет собой вид графики, который создается с помощью математического описания геометрических примитивов (линий, кругов, эллипсов, прямоугольников, кривых), которые образуют изображение

Приложения на языке **SVG** создаются по общим правилам XML с использованием тегов, которые могут иметь атрибуты. **SVG** позволяет описывать собственно векторные изображения (линии и фигуры), форматированный текст, а также включать в документ и работать с растровыми изображениями: масштабировать и трансформировать их подобно векторным объектам. Кроме того, он дает возможность создавать анимационную графику, интерактивные приложения, управлять поведением и внешним видом с помощью JavaScript и CSS

Достоинства SVG:

1. Отсутствие потери качестве при масштабировании
2. Могут создаваться и редактироваться в любом текстовом редакторе
3. Совместимость со стандартами консорциума W3C: DOM и XSL
4. Размеры их файлов являются небольшими по сравнению с любым другим типом файлов изображений
5. Можно добавлять несколько гиперссылок
6. Поддержка скриптов и анимации в SVG позволяют создавать динамичную и интерактивную графику

Минусы:

1. Не поддерживается старыми браузерами — Internet Explorer 8 и старше;
2. WordPress воспринимает .svg как вредоносный код, поэтому блокирует отображение. Обойти блокировку можно с помощью плагинов;
3. Подходит для создания примитивных объектов, которые могут быть описаны простыми фигурами или их частями

**Подключение SVG:**

. С помощью элемента **<object>, <embed>, <img>** изображений в формате **.svg**

1.<img src="example.svg"/>

<embed src="example.svg" type="image/svg+xml"/>

<object data="example.svg" type="image/svg+xml"/>

<iframe src="example.svg" width="200" height="300" style="border: none"/>

2. Вставка кода в HTML-документ в элементе <svg>...</svg>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">  
<!-- SVG-код -->  
</svg>

3. Использование SVG-файла в качестве фонового изображения

<style>  
 .home {  
 display: block;  
 width: 200px;  
 height: 300px;  
 background: url(*example.svg*) 0 0;  
 background-size: contain;  
 }  
 </style>

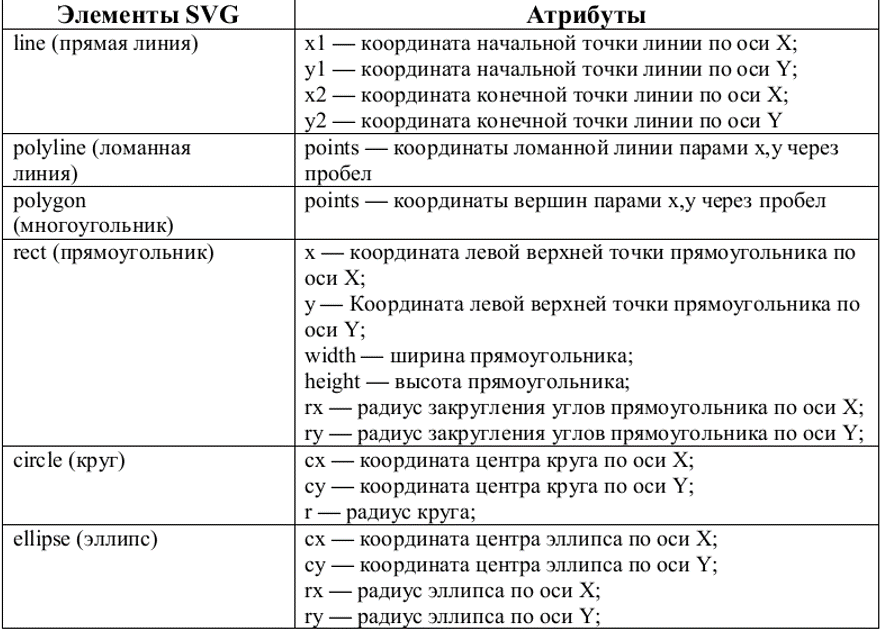
4. Подключение в php

5. Подключение с помощью js

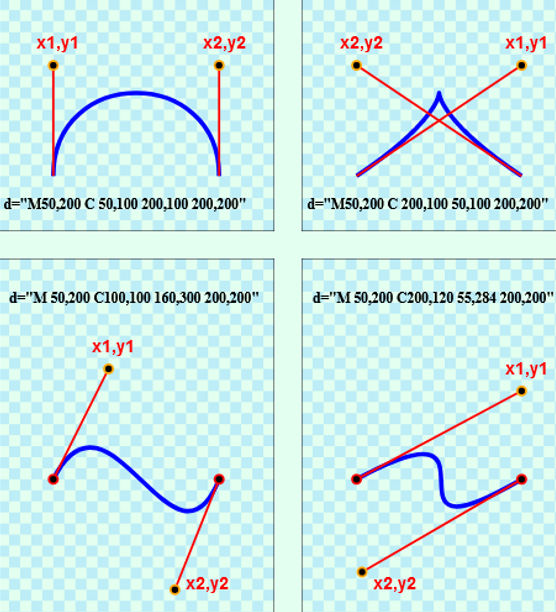
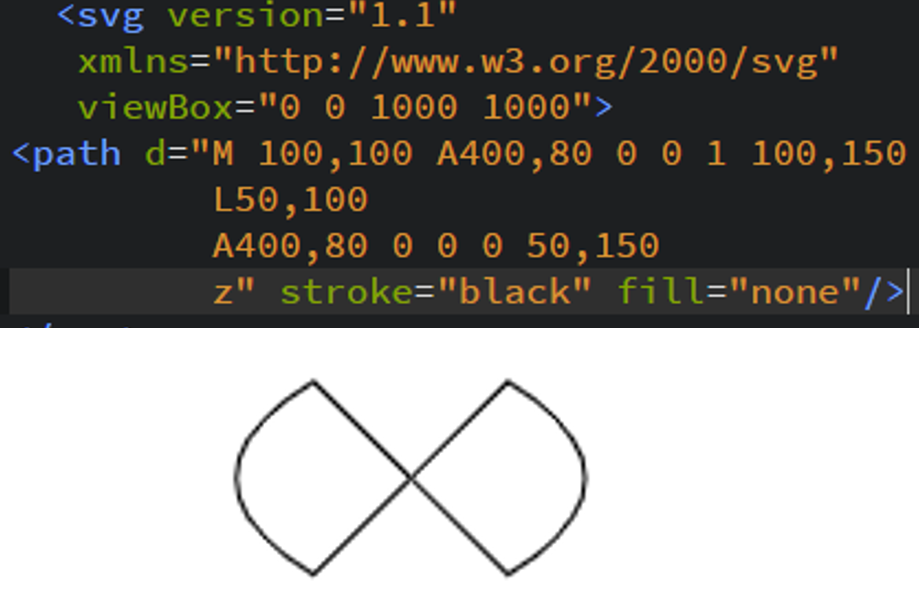
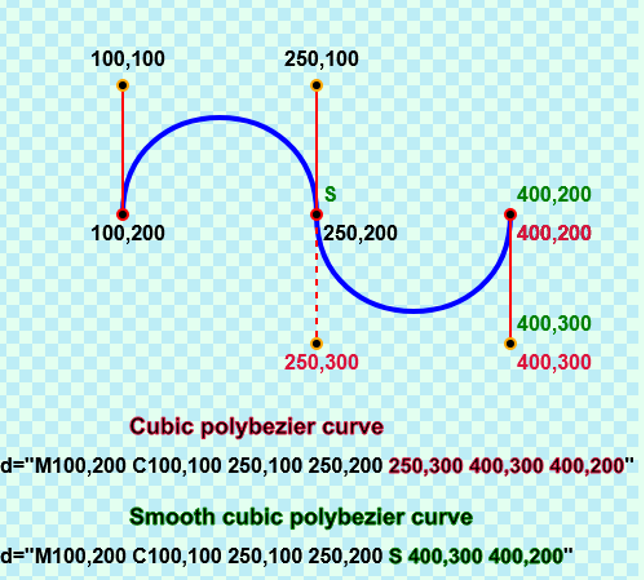
64. Атрибуты SVG

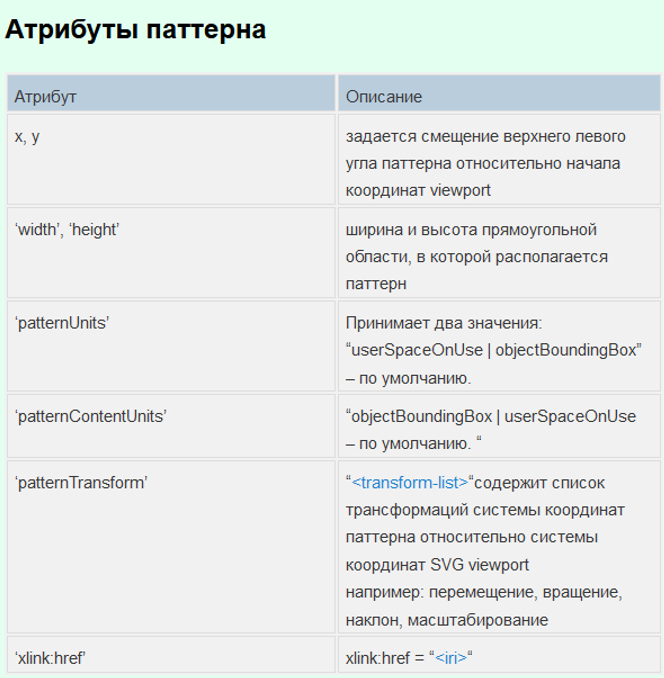
**Общие атрибуты:**

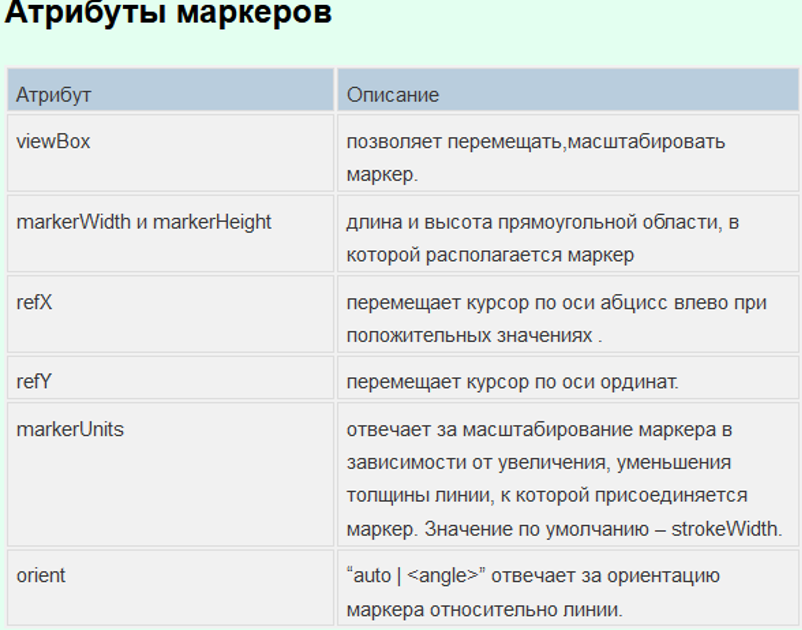
* stroke — определяет цвет линии или контура объекта
* stroke-width  — определяет толщину линии или контура
* stroke-linecap  — определяет стиль концов линии
* stroke-dasharray  — чередование штрихов и пробелов в пунктирной линии
* fill — определяет заливку объекта, ограниченного контуром
* fill-opacity  — определяет прозрачность заливки









65. Анимация SVG

<animate/> используется для анимации одного атрибута или свойства с течением времени

* Атрибут **attributeName** используется для указания имени атрибута, который вы анимируете
* Пространство имен также можно указать с помощью атрибута **attributeType**
* Атрибут **fill** определяет должен ли вернуться в начальное положение анимация
* Атрибут **restart** для перезапуска анимации в любой момент времени
* Атрибут **begin** используется для запуска анимации при обработке события
* Атрибут **repeatCount** для зацикливания анимации. Можно указать желаемое количество повторений либо использовать ключевое слово ***indefinite***, чтобы анимация повторялась бесконечно
* Атрибут **repeatDur** для ограничения времени повторения
* Чтобы указать ключевые кадры, используйте атрибут **keyTimes**. А чтобы задать значения анимируемого свойства, используйте атрибуты **values**.
* Для **set** вы можете указать целевой элемент, имя и тип атрибута, значение to и время анимации, которым можно управлять с помощью следующих атрибутов**: begin, dur, end, min, max, restart, repeatCount, repeatDur и fill**

**Пример:**

<rect>

<animate attributeType="CSS" attributeName="opacity"

from="1" to="0" dur="5s" repeatCount="indefinite" />

</rect>

66. Основные элементы SVG

**Одиночные:**

* **<animate/>** используется для анимации одного атрибута или свойства с течением времени
* **<set/>** установить значение атрибута на заданный период времени
* **<animateTransform/>** анимирует атрибут преобразования на целевом элементе, тем самым позволяя анимации управлять перемещением, масштабированием, вращением и/или наклоном элемента
* **<animateMotion/>** заставляет элемент перемещаться по траектории движения
* **<animateColor/>** определяет изменение цвета с течением времени
* **<use/>** позволяет повторно использовать отображённую графику

**Парные:**

* **<defs> … </defs>** используется как элемент контейнера для элементов, которые предназначены для использования исключительно по ссылке и не отображаются непосредственно
* **<glyph>…</glyph>** определяет графику для заданного символа
* **<missing-glyph> … </missing-glyph>**
* **<g>…</g>** для объединения нескольких фигур в группу для последующих действий над ней
* **<symbol>…</symbol>** предоставляет возможность группировать элементы, однако их не выводит на экран
* **<marker>…</marker>** задается графическое отображение маркера внутри тега **<defs>**
* **<mask> … </mask>** комбинирование значений прозрачности и обрезки
* ***<pattern> … </pattern>*** определяет паттерн
* **<title>…..</title>** для краткого описания смыслового содержания файла
* **<desk>…</desk>** служит для более развернутого описания содержания файла

**Одиночные, которые могут быть и парными:**

* <line/> позволяет создавать прямые линии
* <polyline/> определяет набор соединенных отрезков прямых линий (ломанная линия)
* <polygon/> для создания многоугольных фигур
* <rest/> результатом является прямоугольник
* <circle/> отвечает за вывод окружности
* <ellipse/> отвечает за вывод эллипса
* <path/> для создания сложной траектории

**Особенности элемента <symbol>**

Наличие собственного **‘viewBox’** и **‘preserveAspectRatio’** позволяют дополнительно позиционировать и масштабировать изображения.

<symbol> – является одновременно контейнерным и структурным элементом. В качестве структурного элемента он может включать в себя в любом порядке и количестве: ‘defs’, ‘g’, svg’, ‘symbol’, ‘use’.

67. Работа с текстом SVG

Текст в элементе SVG определяется с помощью тега <text>. К специфическим атрибутам, используемым в text относятся:

1. х и y – базовая линия текста

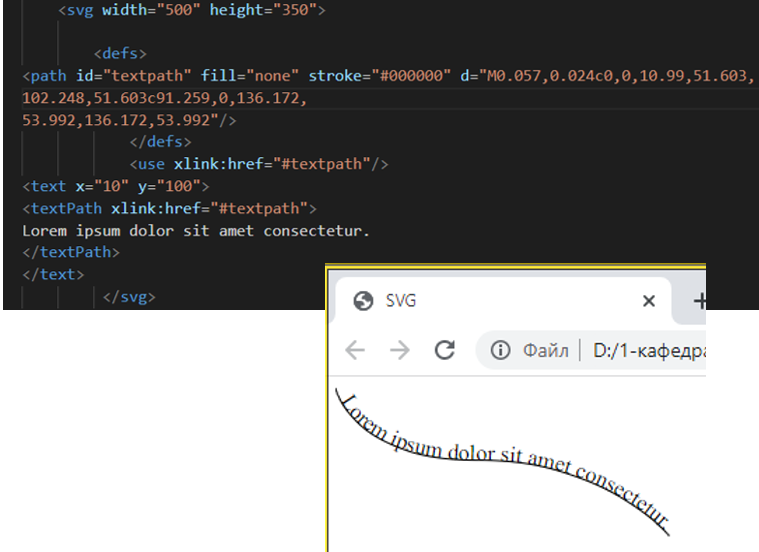
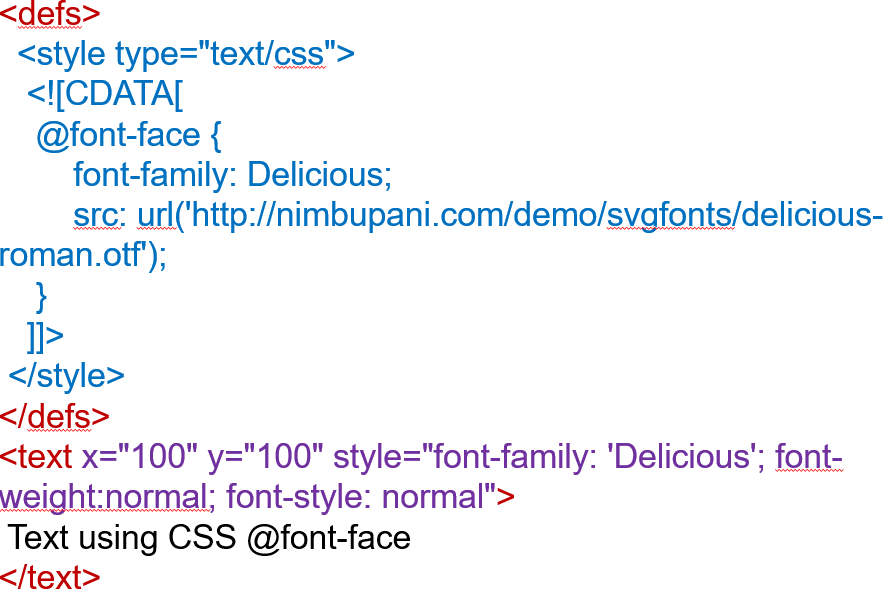
<text x=”0” y=”20”>Text</text>

2. dx и dy – размещение текстовых областей относительно друг друга

3. text-anchor – выравнивание текстовой строки относительно точки (x, y). Может принимать значения start, middle, end

4. rotate – поворот текста transform=”rotate(30 20, 40)”

5. textLength – ширина текстовой области

68. Обзор и версии языков разметки веб-страниц

HTML (от англ. HyperText Markup Language - "язык разметки гипертекста") - стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме.

HTML является приложением ("частным случаем") SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. XHTML же является приложением XML.

Язык HTML был разработан британским учёным [Тимом Бернерсом-Ли](http://bourabai.kz/dhtml/berners.htm) приблизительно в 1989-1991 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов - дескрипторов. Дескрипторы также часто называют "тегами". С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. [Мультимедийные возможности](http://bourabai.kz/mmt/index.htm) были добавлены позже.

HTML - теговый язык разметки документов. Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначается специальными пометками - тегами. Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных (например, тег перевода строки <br>). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, размер шрифта для элемента font). Атрибуты указываются в открывающем теге.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <TABLE>, несколько раз использованный для форматирования страницы, которую вы на даный момент читаете, предназначен для создания в документах самых обычных таблиц, но, как можно убедиться, здесь нет ни одной таблицы. С течением времени, основная идея платформонезависимости языка HTML была отдана в своеобразную жертву современным потребностям в мультимедийном и графическом оформлении.

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые "браузерами" или "интернет-обозревателями", обычно предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome и Opera.

## Версии HTML

**HTML 1.0** - множество нестандартных языков, существовавших в сети до 1995 г.

**HTML 2.0**, стандарт [RFC 1866](http://bourabai.kz/dhtml/rfc1866.htm), одобренный 22 сентября 1995 года;

**HTML 3.0** была предложена Консорциумом всемирной паутины (W3C) в марте 1995 года и обеспечивала много новых возможностей, таких как создание таблиц, "обтекание" изображений текстом и отображение сложных математических формул. Даже при том, что этот стандарт был совместим со второй версией, реализация его была сложна для браузеров того времени.

**HTML 3.1** официально никогда не предлагалась, и следующей версией стандарта HTML стала 3.2, в которой были опущены многие нововведения версии 3.0, но добавлены нестандартные элементы, поддерживаемые браузерами Netscape Navigator и Mosaic.

**HTML 3.2** - введена 14 января 1997 года;

**HTML 4.0** - введена 18 декабря 1997 года. В версии HTML 4.0 произошла некоторая "очистка" стандарта. Многие элементы были отмечены как устаревшие и нерекомендованные (англ. deprecated). В частности, элемент font, используемый для изменения свойств шрифта, был помечен как устаревший (вместо него рекомендуется использовать таблицы стилей CSS).

**HTML 4.01** (изменения, причём более значительные, чем кажется на первый взгляд) - 24 декабря 1999 года;

**ISO/IEC 15445:2000** (так называемый ISO HTML, основан на HTML 4.01 Strict) - 15 мая 2000 года.

**HTML 5** - в настоящее время находится разработке. Черновой вариант спецификации языка появился в Интернете 20 ноября 2007 года. Конец разработки запланирован на 2014 год.

**XHTML 1.0**. В 1998 году консорциум Всемирной паутины начал работу над новым языком разметки, основанном на HTML 4, но соответствующим синтаксису XML. Впоследствии новый язык получил название XHTML. Первая версия XHTML 1.0 одобрена в качестве Рекомендации консорциума Всемирной паутины 26 января 2000 года.

**XHTML 2.0**. Планируемая версия XHTML 2.0 должна была разорвать совместимость со старыми версиями HTML и XHTML, но 2 июля 2009 года консорциум Всемирной паутины объявил, что полномочия рабочей группы XHTML2 истекают в конце 2009 года. Таким образом, была приостановлена вся дальнейшая разработка стандарта XHTML 2.0