ВОПРОСЫ К ЭКЗАМЕНУ ПО ДИСЦИПЛИНЕ  
«КОМПЬЮТЕРНЫЕ ЯЗЫКИ РАЗМЕТКИ»  
1. Представление о макете веб-страницы.

Существуют различные способы создания макетов в HTML.

**Макеты на основе таблиц**

В недалеком прошлом таблицы были самым простым и самым популярным способом создания многоколоночных макетов в HTML. Вся веб-страница помещалась внутри большой таблицы (**<table>**). Однако элемент **<table>** разработан не для создания макетов страниц, его цель — отображение табличных данных.   
Следующий макет создается с помощью таблицы, состоящей из 3-х строк и 2 столбцов, где строки содержащие контент заголовка и футера охватывают обе колонки, используя атрибут *colspan*:

## Макеты на основе DIV-элементов

В течение долгого времени веб-дизайнеры используют элементы **<div>** для группировки элементов на странице (например, оформирующих шапку сайта, статьи, футер или боковую панель). Таким образом разработчики, как правило, помещают эти основные разделы страницы внутри элементов **<div>** и используют атрибуты class или id, чтобы указать назначение конкретной части страницы.

Спецификация HTML5 предоставляет новые семантические теги, описывающие содержащийся в них контент. Так как современные сайты включают в свои страницы боковые панели, заголовки и секции, в спецификацию HTML5 были включены новые теги, предназначенные для деления страницы на логические области. Семантическая разметка предназначена для описания контента. Семантическая разметка упрощает понимание смысла и контекста информации, размещенной на странице — как для поисковиков, так и для разработчиков. Теперь программы экранного доступа могут позволить пользователям проигнорировать заголовки и нижние колонтитулы и сразу перейти к основному контенту. Кроме того, поисковые машины могут придать большее значение содержимому элементов **<article>**, чем содержимому **<header>** или **<footer>**.



Зачастую основными элементами страницы являются: содержащий блок (wrapper, container), логотип, навигация, контент, футер (нижний колонтитул), свободное пространство (по сути свободное пространство — это не элемент дизайна, но понятие, помня о котором при составлении макета страницы, наш проект не будет выглядеть как нагромождение блоков).

**Содержащий блок (контейнер)**

Роль контейнера на странице может выполнять непосредственно элемент body или же div. Ширина содержащего блока может быть резиновой (fluid), а может быть фиксированной (fixed).

#### Логотип

Текстовая или графическая составляющая проекта и выделяющая его среди других. Логотип чаще всего располагается в верхнем левом углу страницы или же посередине (в зависимости от идеи, макета).

#### Навигация

Основная навигационная панель содержит ссылки на основные разделы сайта. Навигационная панель часто располагается в верхней части страницы (в независимости от того вертикально или горизонтально располагаются элементы навигации).

#### Контент

Контент – это основная составляющая веб-страницы. Он занимает главенствующую роль в дизайне страницы, поэтому занимает большее пространство, подкреплён, помимо текста, графикой.

#### Нижний колонтитул (footer) Данный элемент располагается внизу страницы и обычно содержит информацию о правообладателе, контактные и юридические данные, ссылки на основные разделы сайта (зачастую дублирует основную навигацию), ссылки на социальные сети, форму обратной связи и пр.

2. Рекомендации для составления веб-страницы.

**1. Нужно определиться с задачами и целями, которым будет служить сайт.**

Перед тем как взяться за непосредственное написание кода нужно определиться – зачем вообще нужен сайт, какие именно задачи он будет решать, какую пользу вы желаете извлечь из вашего ресурса сами и чем он будет полезен другим пользователям? Нужно отдавать себе отчет, что сайт - это продукт, развивающийся и улучшающийся со временем, поэтому перед его созданием нужно определиться в тех точках и аспектах, на которые будет сделан основной упор. От того, насколько будут проработаны основные моменты, будет зависеть насколько успешно станет развиваться сайт в дальнейшем.

**2. Полезно разработать схему будущего сайта.**

Не малой пользой при разработке индивидуального проекта будет служить создание схемы будущего сайт. Можно воспользоваться графическим редактором, для того, чтобы представить, как будет выглядеть планируемый сайт. В каких местах и какие именно блоки займут свое место. Какие сервисы понадобятся и на каких страницах будут расположены. Определиться с элементами, которые будут повторяться, а также с теми, которые станут изменяться динамически. Разделять основной блок принято на двенадцать колонок и все элементы располагать сообразно данным колонкам. Чтобы облегчить задачу, нужно изучить уже готовые сайты, проанализировать их структуру, обратить внимание на сильные и слабые моменты. Достоинства несомненно стоит реализовать на своем ресурсе. Тщательный анализ и подготовка перед программированием убережет от значительных переделок и сэкономит немало средств и времени.

**3. После составления схемы сайта можно приступить к воплощению его визуальной составляющей.**

Прежде чем нарисовать сайт рекомендуется изучить направления в современном веб-дизайне, благо на эту тему написано не мало статей. Начинающему веб-дизайнеру можно с уверенностью рекомендовать не изобретать велосипед заново и брать готовые, уже устоявшиеся решения у уже получивших значительный опыт мастеров. Рекомендуется не использовать большого обилия цветов, шрифтов, стилей оформлений изображений. В дизайне должно сохраняться единообразие.

**4. Разработчик создает сайт для людей.**

При написании своего ресурса в первую очередь необходимо думать о конечном пользователе, чтобы сайт был удобен в использовании и полезен. Не нужно забывать, что сайт должен быть настроен таким образом, чтобы посетители хотели возвращаться на него регулярно. Нужно использовать продуманную навигацию, как правило спокойные цвета в оформлении выглядят лучше, нежели яркие, броские, текст должен располагаться на светлом фоне и хорошо читаться. Не стоит использовать слишком вычурные шрифты.

Данные советы помогут организовать пространство вашего сайта таким образом, чтобы посетителю было комфортно пребывать на нем.

**5. Заранее подумайте о наполнении сайта.**

Помните, люди приходят в Интернет в поисках определенной информации, значит, у Вас на сайте, должно быть что-то интересное, причем интересное для посетителей сайта по выбранной Вами тематике. Подумайте, чем Вы будете наполнять свой сайт, какого рода Контент (содержание сайта) Вы будете размещать: текст, видео файлы, аудио файлы, софт и т.д. Подумайте, из каких источников Вы будете в дальнейшем черпать информацию для своего сайта.

Разбейте для себя Контент на первоочередной (то что Вы выложите сразу) и долгосрочный (чем будете наполнять сайт в дальнейшем).

**6. Подумайте, как Вы будете управлять сайтом в дальнейшем.**

По мере разрастания сайта перед Вами встанет вопрос обслуживания сайта, а именно: создавать новые страницы, редактировать или удалять существующие страницы, публиковать новости и т.д. Вы будете делать это сами или же поручите кому-то сопровождение Вашего сайта, но в таком случае будьте готовы к финансовым затратам, ведь никто не будет за спасибо постоянно обслуживать Ваш сайт.

Возможно, это будет просто небольшой сайт визитка (1-2 web-странички, может 10), не требующий постоянного сопровождения и обновления.

Если же Вы хотите сами, научиться делать сайты и тогда проблем по дальнейшему сопровождению и наполнению сайта у Вас не возникнет, тогда обратите внимание на Пошаговый Обучающий Курс: Сайт от А до Я

Перед запуском сайта в Интернете, обязательно проверьте работоспособность всех ссылок. Это важный момент.

**7. Протестируйте свой сайт сторонними посетителями.**

Что значит протестируйте? После размещения сайта в Интернете (хотя это можно сделать и локально на Вашем домашнем компьютере) и широкого рекламирования, попросите нескольких знакомых, побродить по сайту, посмотреть на структуру сайта, как у Вас все организовано и как все работает. И обязательно выясните мнение этих людей. Зачем? Помните Вы делаете сайт для людей и то что может быть понятно Вам как разработчику, может вызвать недоумение и как следствие отторжение других посетителей. Учтите мнение этих людей и обязательно внесите коррективы.

Всего 7 небольших пунктов, помогут Вам создать вполне достойный сайт. Все это не так сложно и трудно, как может показаться вначале, достаточно следовать этому простому плану. Конечно в каждом из пунктов есть свои нюансы, которые трудно постигнуть с ходу, но все приходит со временем и постепенно.

3. Инструменты для редактирования веб-страницы.----

4. Основы HTML5: историческая справка, стандарты, структура html-документа, назначение

**Основы HTML** содержат основные правила языка HTML, описание структуры HTML-страницы, отношения в структуре HTML-документа между HTML-элементами.

HTML-документ — это обычный текстовый документ, может быть создан как в обычном текстовом редакторе **(Блокнот)**, так и в специализированном, с подсветкой кода **(Notepad++, Visual Studio Code и т.п.)**. HTML-документ имеет расширение .html.

HTML-документ состоит из дерева HTML-элементов и текста. Каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегом (за редким исключением).

**Начальный тег** показывает, где начинается элемент, конечный — где заканчивается. **Закрывающий тег** образуется путем добавления слэша / перед именем элемента: <имя элемента>…</имя элемента>. Между начальным и закрывающим тегами находится содержимое элемента — контент.

Элементы, представленные одиночными тегами, не могут хранить в себе содержимого напрямую, оно прописывается как значение атрибута, например, элемент <input type="button" value="Кнопка"> создаст кнопку с текстом **Кнопка** внутри.

Элементы могут вкладываться друг в друга, например, <p><i>Текст</i></p>. При вложении следует соблюдать порядок их закрытия **(принцип «матрёшки»)**, например, следующая запись будет неверной: <p><i>Текст</p></i>.

HTML-элементы могут иметь атрибуты (глобальные, применяемые для всех HTML-элементов, и собственные). Атрибуты прописываются в открывающем теге элемента и содержат имя и значение, указываемые в формате имя атрибута="значение". Атрибуты позволяют изменять свойства и поведение элемента, для которого они заданы.

Каждому элементу можно присвоить несколько значений class и только одно значение id. Множественные значения class записываются через пробел, <div class="nav top">. Значения class и id должны состоять только из букв, цифр, дефисов и нижних подчеркиваний и должны начинаться только с букв или цифр.

Браузер просматривает (интерпретирует) HTML-документ, выстраивая его структуру (DOM) и отображая ее в соответствии с инструкциями, включенными в этот файл (таблицы стилей, скрипты). Если разметка правильная, то в окне браузера будет отображена HTML-страница, содержащая HTML-элементы — заголовки, таблицы, изображения и т.д.

Процесс интерпретации **(парсинг)** начинается прежде, чем веб-страница полностью загружена в браузер. Браузеры обрабатывают HTML-документы последовательно, с самого начала, при этом обрабатывая CSS и соотнося таблицы стилей с элементами страницы.

HTML-документ состоит из двух разделов — заголовка — содержимое элемента <head> и содержательной части — содержимое <body>.

### Структура HTML-документа

Язык HTML следует правилам, которые содержатся в файле объявления типа документа (Document Type Definition, или DTD). DTD представляет собой XML-документ, определяющий, какие элементы, атрибуты и их значения действительны для конкретного типа HTML. Для каждой версии HTML есть свой DTD.

**DOCTYPE** отвечает за корректное отображение веб-страницы браузером. DOCTYPE определяет не только версию HTML (например, html), но и соответствующий DTD-файл в Интернете.

Элементы, находящиеся внутри элемента <html>, образуют дерево документа, так называемую **объектную модель документа, DOM (document object model)**. При этом элемент <html> является корневым элементом.

Чтобы разобраться во взаимодействии элементов веб-страницы, необходимо рассмотреть так называемые **«родственные отношения»** между элементами. Отношения между множественными вложенными элементами подразделяются на родительские, дочерние и сестринские.

* **Предок** — элемент, который заключает в себе другие элементы. На рисунке 1 предком для всех элементов является <html>. В то же время элемент <body> является предком для всех содержащихся в нем элементов: <h1>, <p>, <span>, <nav> и т.д.
* **Потомок** — элемент, расположенный внутри одного или более типов элементов. Например, <body> является потомком <html>, а элемент <p> является потомком одновременно для <body> и <html>.
* **Родительский элемент** — элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их. На рисунке 1 <html> является родительским только для <head> и <body>. Элемент <p> является родительским только для <span>.
* **Дочерний элемент** — элемент, непосредственно подчиненный другому элементу более высокого уровня. На рисунке 1 только элементы <h1>, <h2>, <p> и <nav> являются дочерними по отношению к <body>.
* **Сестринский элемент** — элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня. На рисунке 1 <head> и <body> — элементы одного уровня, так же как и элементы <h1>, <h2> и <p> являются между собой сестринскими.

### Элемент <html>

Является корневым элементом документа. Все остальные элементы содержатся внутри <html>...</html>. Все, что находится за пределами элемента, не воспринимается браузером как HTML-код и никак им не обрабатывается.

Раздел <head>...</head> содержит техническую информацию о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т.д. Введенная в нем информация не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.

Необязательным элементом раздела <head> является элемент <meta>. С его помощью можно задать описание содержимого страницы и ключевые слова для поисковых машин, автора HTML-документа и прочие свойства метаданных.

Элемент <head> может содержать несколько элементов <meta>, потому что в зависимости от используемых атрибутов они несут различную информацию.

#### Элемент <style>

Внутри этого элемента задаются стили, которые используются на странице. Для задания стилей в HTML-документе используется язык CSS. Таких элементов на странице может быть несколько.

В разделе <body> располагается все содержимое документа.

5. Кодировки (ASCII, Windows-1252, UTF-8, ISO-8859-1)

**Кодировка символов** международное представление букв, цифр и символов в веб-странице или любом другом файле, который хранится на компьютере и может быть передан через Интернет.

**ASCII** (American Standard Code for Information Interchange, американская стандартная кодировочная таблица для печатных символов и некоторых специальных кодов)— одна из самых старых компьютерных кодировок, в которой каждому символу соответствует строго определенное число. Это 7-битная кодировка, содержащая 128 символов

**Windows-1251** — набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для всех русских версий Microsoft Windows. Данная кодировка пользуется довольно большой популярностью в восточно-европейских странах. Windows-1251 отличается от других 8-битных кириллических кодировок наличием практически всех символов, использующихся в традиционной русской типографике для обычного текста (отсутствует только знак ударения)

**ISO-8859-1** — кодировка по умолчанию в большинстве современных браузеров.

Первые 128 символов ISO-8859-1 закодированы так же, как и в ASCII (прописные и строчные латинские буквы, цифры от 1 до 9 и некоторые другие символы).

Коды от 160 до 255 содержат символы, используемые в западно-европейских странах и часто используемые спецсимволы.

**UTF-8 –** кодировка символов юникод в двоичном виде. Область UTF-8 с кодами от U+0000 до U+0500 — это базовая область символов кирилицы. Если Вы хотите какой-либо из этих знаков отобразить в HTML-документе, Вы можете использовать шестнадцатеричное **(Hex)** значение ( &#x + код + ;) или десятичное **(Dec)** значение ( &# + код + ;) из таблицы кодов

6. Служебные и структурные теги.

К служебным тегам относят:

* html;
* head;
* title;
* meta;
* style;
* body;
* script.

**Тег html**

Требуется в начале веб-страницы и с добавленным слешем в конце: </html>. Элемент html сообщает браузеру, что документ является программным кодом на языке HTML. Все содержимое страницы, включая остальные элементы, находится между открывающим и закрывающим тегами элемента html

**Атрибуты html:**

**Атрибут dir**

Определяет, в каком направлении браузер должен выводить текст в элементе, к котором применяется этот атрибут. По умолчанию выводит слева направо (значение **ltr**), для арабского и иврита применяется значение **rtl**

**Атрибут lang**

Определяет язык, который употребляется главным образом в документе. Использованный с другим тегом, атрибут lang назначает язык содержимого тега. Применяется, чтобы лучше отображать текст.

Принимает значения: en, en-US, fr, de, it, ul (голландский), el, es, pt, ar, he, ru, zh, hi и т.д. согласно ISO-639

**Тег head**

Служебный тег, который содержит в себе всю техническую информацию о веб-странице, а также задает ее заголовок. Эта информация не видна обычному пользователю, но помогает браузерам и поисковым системам работать с данными страницы

В нем могут содержаться такие элементы, как **title, base, meta, link, style**

**Атрибуты head:**

**Атрибут profile**

Указывает адрес URL профиля метаданных. Обычно браузер распознаёт значение этого атрибута и выполняет некоторые соглашения, связанные с указанным профилем. Загрузки самого документа по указанному адресу в реальности не происходит, более того, его может вообще не быть

**Тег title**

Используется для определения заголовка документа и отображается в заголовке окна браузера

**Тег link**

Используется для подключения к документу стилей, скриптов, изображений или значков из внешних файлов

**Атрибуты тега link:**

**Атрибут href**

от англ. "hypertext reference" ‒ «гипертекстовая ссылка»

**Указывает местоположение внешнего ресурса или файла.**

**Атрибут rel**

от англ. “relation" ‒ «отношение»

**Определяет, чем для нашего документа является подключаемый файл**

**<link href="style.css" rel=“stylesheet“>**

**Тег meta**

Содержит дополнительные данные о документе (метаданные). Эти данные используют браузеры для обработки страницы, а поисковые системы — для ее индексации.

**Атрибуты тега meta:**

**Атрибут charset**

Определяет кодировку документа

<meta charset="utf-8">

**Атрибут http-equiv**

Управляет действиями браузеров и используются для формирования информации, выдаваемой обычными http-заголовками

**Значения атрибута http-equiv**

**Content-Type** тип кодировки документа

**expires** устанавливает дату и время, после которой информация в документе будет считаться устаревшей

**pragma** способ кэширования документа

**refresh** указывает время в секундах до перезагрузки страницы или время до перенаправления на другую страницу

**Значения атрибута http-equiv**

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<meta http-equiv="expires" content="Sun, 01 Jan 2013 07:01:00 GMT">

**Обновление страницы через 10 секунд**

**<meta http-equiv="refresh" content="10">**

**Атрибут name**

Определяет название метаданных. Данный атрибут используется совместно с атрибутом content.

Не может быть использован одновременно с атрибутом **http-equiv**

**Значения атрибута name**

**application-name** указывает название веб-приложения, используемого на странице  
author используется для указания имени автора веб-страницы

**description**  информация, которую вы види-те на странице поисковика, когда ваша страница попадает в результаты запроса

**generator** указывает один из пакетов программного обеспечения, используемого для создания документа

**keywords** содержит список ключевых слов, разделенных запятыми, соответствующих содержимому страницы

**robots** показывает, должны ли поисковые системы включать данную страницу в результаты поиска

**viewport** позволяет разработчикам управлять размером исходной области просмотра на различных устройствах

Служебный тег style определяет стили элементов страницы. Например, стиль H1 (его размер, шрифт, цвет). Задается данный тег в контейнере head и может применяться неоднократно (для разных элементов).

Задав стиль в head, браузер использует его в body к соответствующему элементу.

Атрибуты:

* media — позволяет определить устройство вывода, для которого задана таблица стилей. Для каждого устройства можно задать свой стиль;
* type — задает синтаксис, который необходимо использовать для верной интерпретации таблиц стилей. По умолчанию значение равно «text/css».

Элемент body — это служебный тег, содержащий в себе все данные, которые отображаются на веб-странице (текст, медиафайлы, таблицы и т. д.). Иными словами, body отвечает за отображение контента документа.

Внутри контейнера body используются все элементы, предназначенные для дизайна страницы. Но иногда там могут быть расположены и обработчики событий. Например, onload, который выполняется после того, как документ завершит загрузку.

Внутри открывающего тега body можно разместить атрибуты, задающие определенные значения параметров для всей веб-страницы целиком. Например, ее фон, цвет текста, цвет гиперссылок и т. д.

Атрибуты:

* alink — цвет активных ссылок;
* background — фон страницы (изображение, картинка, видео);
* bgcolor — цвет фона (если это простая заливка);
* bgproperties — свойства фона (прокручиваться вместе с контентом, оставаться неподвижным, прокручиваться в режиме Paralax и т. д.);
* bottommargin, topmargin, rightmargin, leftmargin — отступы от соответствующих краев страницы до контента;
* link — цвет ссылок;
* scroll — отображение полос прокрутки;
* text — цвет текста на странице;
* vlink — цвет нажатых ссылок.

Кроме вышеописанных атрибутов в служебном теге body можно также использовать универсальные атрибуты и события.

Элемент script — служебный тег, предназначенный для описания скриптов. Он может содержать в себе ссылку на саму программу или ее текст, написанный на другом языке. Данный тег позволяет HTML-документам работать с программами, написанными на языке JavaScript и др.

Один и тот же скрипт может храниться во внешнем файле и использоваться одновременно для нескольких страниц. Это позволяет выполнять его быстрее при последующих загрузках, так как при первой загрузке файл кэшируется.

Атрибуты для тега script:

* async — позволяет загружать скрипт асинхронно;
* defer — не дает скрипту начать работу до тех пор, пока страница не будет загружена полностью;
* language — указывает язык программирования скрипта;
* src — адрес внешнего файла;
* type — определяет тип содержимого тега script.

Служебный тег script можно располагать в любом месте документа (как в контейнере head, так и в контейнере body).

**Тег base**

Тег <base> используется, если требуется выполнить локальное тестирование. Браузер ищет элемент <base>, определяет полный адрес документа и корректно загружает его.

Например, если адрес документа указан как <base href="http://www.megasite.ru/hzchd/">, то при добавлении рисунков достаточно использовать относительный адрес <img src="images/labuda.gif">. При этом полный путь к изображению будет http://www.megasite.ru/hzchd/images/labuda.gif, что позволяет браузеру всегда находить графический файл, независимо от того, где находится текущая веб-страница.

**Структурные теги**

## Тег section

Тег section группирует контент по тематике, или, другими словами, определяет раздел документа. Обычно предшествует header, может находиться в footer. Тег section может быть заключен внутрь другого тега section, если необходимо, и содержать любой объем разметки. В обычном html, как правило, вместо тега section используют div.

## Тег header

Тег header (шапка) обычно содержит заголовок или группирует заголовки, но также может содержать информацию о секции.

Другими словами тег header определяет верхний колонтитул сайта (шапку сайта).

## Тег footer

Тег footer (подвал) обычно содержит информацию о секции, например: автор статьи, дата, авторское право и т.д.

#### HTML

<footer>

<p>Copyright © 20011 All Rights</p>

</footer>

## Тег nav

Тег nav определяет навигационную область, обычно это список ссылок. Тег nav является родным братом основной section (секции), header (шапки) и footer (подвала).

#### HTML

<nav>

<ul>

<li><a href="#">Главная</a></li>

<li><a href="#">Скачать</a></li>

<li><a href="#">Архив</a></li>

<li><a href="#">О Нас</a></li>

</ul>

</nav>

Элемент nav можно расположить также, например, в теге header.

## Тег aside

Тег aside определяет контент, который расположен вокруг основного контента. Это может быть боковая панель (sidebar), содержащая ряд ссылок на статьи, на архив, на метки.

## Тег article

Тег article определяет любую независимую запись (статью) на сайте, или, например, это может быть резюме, описание товара в интернет магазине и так далее.

## Тег figure

Иногда бывает необходимо разместить изображение и прилагающийся к нему текст как единую иллюстрацию. Элемент figure указывает на то, что текст и изображение являются частями единого целого.

#### HTML

<figure>

<legend>

Тут я описываю изображение

</legend>

<img alt="" src="opa.png" />

</figure>

## Тег dialog

Тег dialog – это та область, где отображается комментарии (беседа, разговор). Тег dialog состоит из следующих частей:

* сам тег dialog, определяющий блок с комментариями
* тег dt определят того кто говорит
* тег dd определяет текст комментария

#### HTML

<dialog>

<dt>

Вася

</dt>

<dd>

я тут был

</dd>

<dt>

Петя

</dt>

<dd>

я подтверждаю, все что написал Вася правда

</dd>

</dialog>

7. Семантические теги.

***Семантика*** – это изучение значений слов и фраз на языке.

Семантические элементы = элементы с смыслом

**Семантические теги** — это теги, которые предназначены для того чтобы компьютерные программы (поисковые системы, сборщики информации, речевые браузеры и т.д.), понимали какой тип информации заложен в данных тегах.

**header** устанавливает верхний колонтитул («шапку») для страницы или раздела

**nav** предназначен для создания блока навигации веб-страницы или всего веб-сайта, при этом не обязательно должен находиться внутри

**footer** устанавливает нижний колонтитул («подвал») для страницы или раздела

**section** представляет собой общий раздел документа или приложения. Раздел в этом контексте представляет собой тематическую группу контента, обычно с заголовком

**article** представляет собой полную или автономную композицию в документе, странице, приложении или сайте и, в принципе, может независимо распространяться или повторно использоваться, например в синдикации. Это может быть сообщение на форуме, статья в журнале или газете, запись в блоге, комментарий пользователя, интерактивный виджет или гаджет или любой другой независимый элемент контента.

**main** определяет уникальный основной контент для веб-страницы, в нем не должно быть элементов, располагаемых на всех страницах сайта, например, заголовка, нижнего колонтитула и основной навигационной панели

**aside** предназначен для оформления боковых колонок, рекламных блоков, группирования навигационных элементов и прочего содержимого, которое лишь косвенно касается основного контента страницы.

**details** используется для информации, которую можно скрыть или показать по желанию пользователя

**summary** определяет видимый заголовок для элемента

**figure** заключает в себе потоковый контент, который может сопровождаться заглавием <figcaption>. Предназначен для контента, который можно удалить со страницы не изменяя смысла остальной информации.

**mark** представляет фрагмент текста в документе, помеченный цветом или выделенный подсветкой для справочных целей

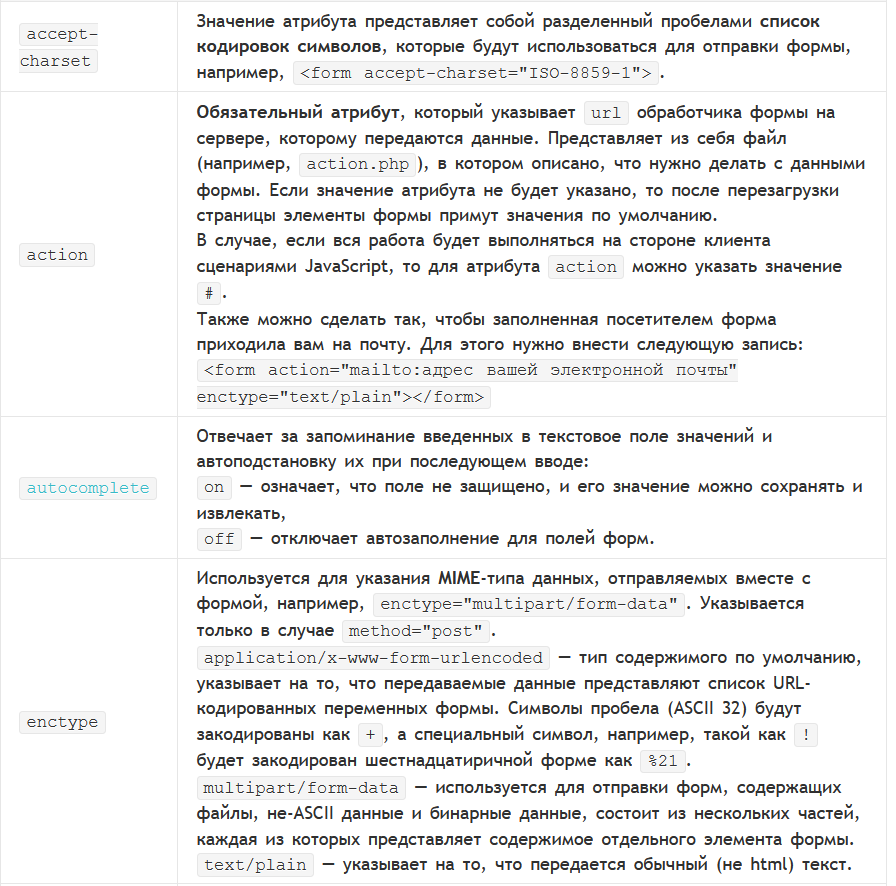
**time** используется для обозначения конкретной даты и времени

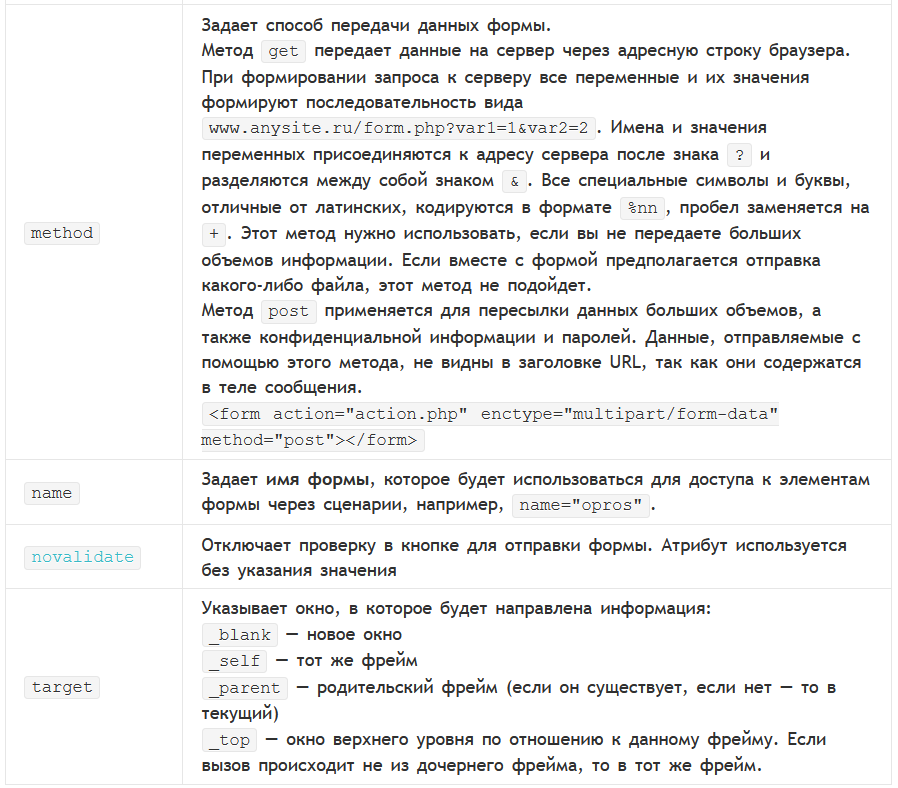
**address** контактная информация элемента <article> или <body>

8. Теги формы.

**form** определяетHTML-форму для ввода пользователем данных, которые впоследствии можно принять и обработать на стороне сервера

**Атрибуты form:**

****

****

**input** создает пользовательские элементы формы

**Атрибуты:**

**type** определяет вид элемента формы

**аccept** устанавливает фильтр на типы файлов, которые можно отправить через поле загрузки файлов

**alt** альтернативный текст для кнопки с изображением

**autocomplete** устанавливает или отменяет автозаполнение полей формы

**autofocus** автоматически переводит фокус на элемент формы сразу после полной загрузки страницы

**checked** применяется для изначальной установки поля с флажком или радио-кнопкой в активное состояние

**dirname** указывает уникальное имя значения направления вывода текста

**disabled** блокирует элемент формы, то есть делает его неактивным

**autocomplete** устанавливает или отменяет автозаполнение полей формы

**form** указывает одну или несколько форм, к которой принадлежит элемент

**formaction** указывает расположение обработчика данных формы

**formenctype** указывает способ кодирования данных формы, отправляемых на сервер

**formmethod** указывает метод отправки данных формы на сервер

**formnovalidate** отключает проверку данных формы (перед отправкой на сервер) на корректность

**list** с помощью идентификатора присоединяет к элементу формы элемент **<datalist>** с вариантами автодополнения

**max** указывает максимальное значение элемента

**pattern** задаёт шаблон для значения текстового элемента формы

**placeholder** указывает краткую подсказку (слово или короткую фразу), предназначенную для помощи пользователю с вводом данных

**readonly** поле ввода предназначено только для чтения

**required** указывает что данный элемент формы (или группа элементов) обязателен для заполнения перед отправкой формы

**value** присваивает элементу значение

**label** устанавливает связь между определенной меткой, в качестве которой обычно выступает текст, и элементом формы

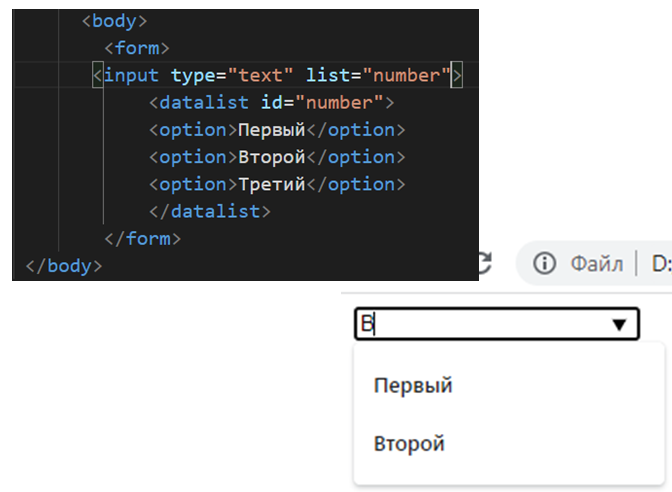
**textarea** определяет многострочное текстовое поле

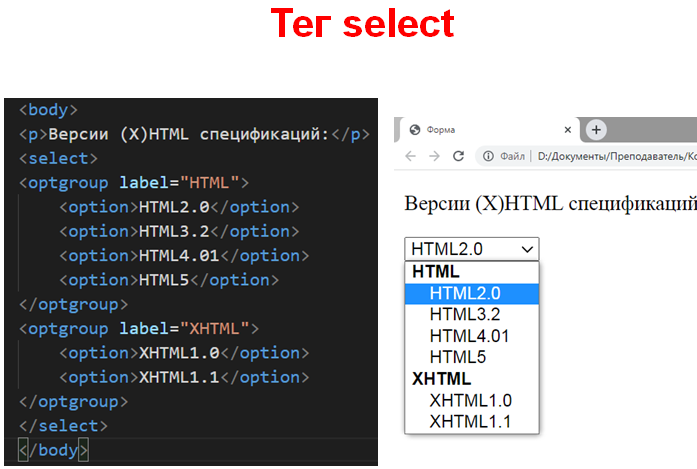
**datalist** используется для автоматического выбора значения элемента

**select** используется для создания раскрывающегося списка, который появляется при щелчке мышью по элементу формы

**ortgroup** группирует пункты в раскрывающемся списке

**option** создает элемент раскрывающегося списка





**button** cоздает в документе интерактивные кнопки

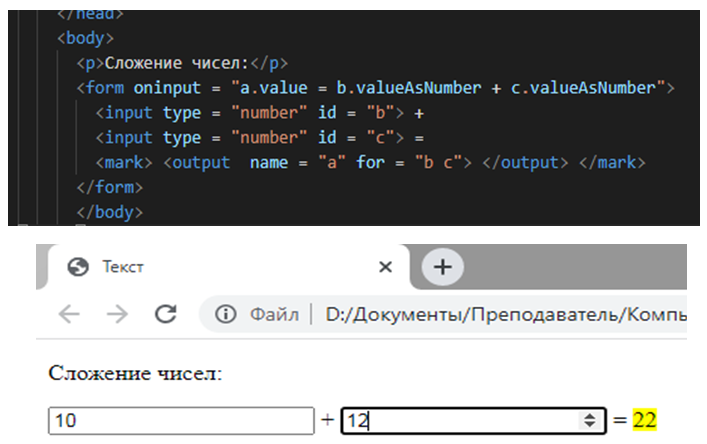
**fieldset** служит для группировки элементов, размещенных внутри формы

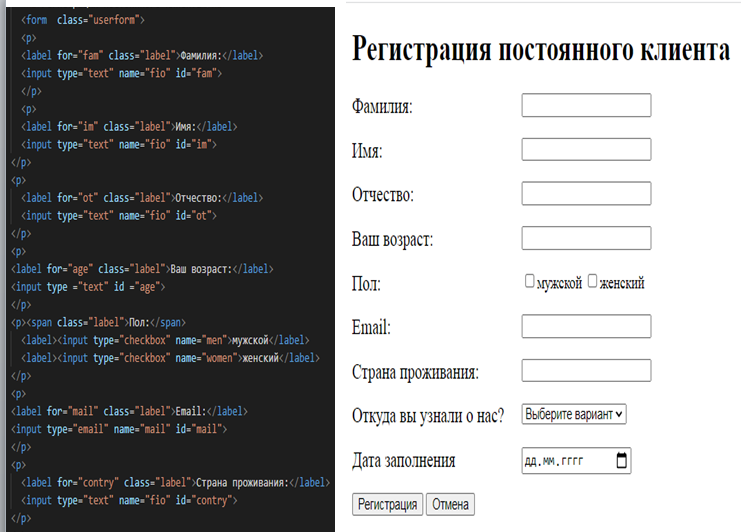
**keygen** используется для шифрования значения элемента формы

**progress** определяет индикатор прогресса выполнения задачи, отображающего, какой процент задачи уже выполнен

**meter** определяет скалярное измерение в пределах известного диапазона или дробное значение

**output** определяет область, в которую выводится результат вычислений или действий пользователя (обычно рассчитывается с помощью скриптов)значение





9. Теги списков, таблиц и изображений.

Теги списков:

<ul> маркированный список

**Атрибут type**-**Определяет вид маркера списка**

<ol> нумерованный список

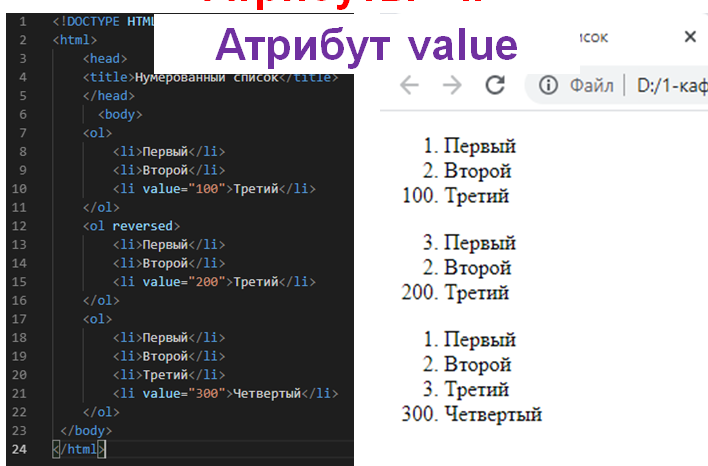
**Атрибуты:**

Атрибут start -Позволяет задать номер списка, с которого будет начинаться нумерация

Атрибут reversed-Нумерация в списке становиться по убыванию

Атрибут type-Устанавливает вид маркера списка

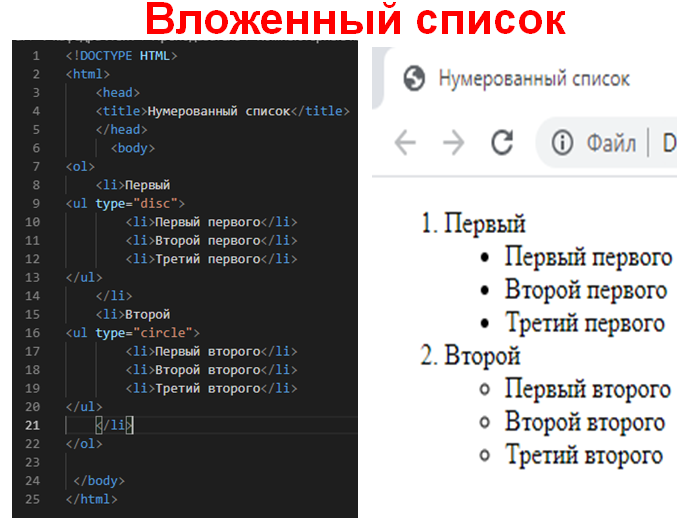
<li> пункт списка маркированного или нумерованного



<dl> список определений

<dt> для добавления термина определения

<dd> для вставки определения





**<table> выделяет элементы таблицы из содержимого документа**

**<tr> определяет строку**

**<th> определяет заголовок строки или столбца (необязательный)**

**<td> для создания ячейки**

**Атрибуты:**

**colspan** объединяет несколько ячеек в строке

**rowspan** объединяет несколько ячеек в столбце

**headers** позволяет связать ячейки таблицы с заголовками

**<caption> для добавления названия таблицы перед ней**

**<thead> для определения группы заголовочных строк**

**<tfoot> для вставки нижнего колонтитула таблицы**

**<tbody> для разбиения таблицы на разделы**

**<colgroup> определяет группу столбцов в таблице**

**<col> определяет столбец в группе стобцов**

**Атрибуты:**

**span** позволяет указать какое количество столбцов одновременно можно изменить

Атрибут **summary** предназначен для краткого описания таблицы или указания ее предназначения. В отличие от тега **<caption>** содержимое summary никак не отображается и в браузере не выводится, однако может использоваться поисковыми системами или речевыми браузерами

Теги изображений:

**img** используется для вставки в изображений

**Атрибуты:**

**src (от англ. Sorce источник)** указывает местоположение файла изображения

**alt** атрибут содержит текст, который браузер выводит вместо изображения, если графика не поддерживается, или показывает в виде подсказки, когда пользователь размещает над изображением мышиный курсор.

**height** высота изображения

**width** ширина изображения

**srcset** указывает список из одного или нескольких значений, разделённых запятыми, указывающих набор возможных изображений для отображения в браузере

**sizes** задает размеры изображения для разных макетов страницы

**usemap** определяет изображение в качестве клиентской карты-изображений

**ismap** указывает браузеру, что данное изображение является серверной картой-изображением

**map** создаёт карту-изображение.

**Атрибут name** присваивает уникальное имя элементу, через которое устанавливается связь с изображением

*Карта изображений* — это зоны обычного изображения, при нажатии на которые происходит активация ссылок связанных с конкретной зоной

**area** определяет активную зону карты-изображения

**Атрибуты:**

**alt** альтернативный текст для области изображения.

**coords** координаты активной области

**download** сообщает о том, что эта ссылка должна быть использована для скачивания файла, и, когда пользователь нажимает на ссылку, ему будет предложено сохранить файл как локальный.

**href** задает адрес документа, на который следует перейти

**shape** форма области.

**target** имя фрейма, куда браузер будет загружать документ

**canvas** определяет область, в которой можно рисовать при помощи скриптов

**В HTML5 была прекращена поддержка атрибутов**

***align, border, hspace, vspace***

10. Теги для встраиваемого контента.

**audio** добавляет, воспроизводит и управляет настройками аудиозаписи на веб-странице

**Атрибуты:**

**autoplay звук начинает играть сразу после загрузки страницы.**

**сontrols добавляет панель управления к аудиофайлу.**

**loop повторяет воспроизведение звука с начала после его завершения.**

**muted отключает звук при воспроизведении музыки.**

**preload указывает как аудио должно загружаться при загрузке страницы. Атрибут игнорируется, если присутствует атрибут autoplay.**

**src указывает путь к воспроизводимому файлу**

**video** вставляет видео файл в HTML-документ. В настоящее время поддерживается три видео формата — MP4, WEBM, OGG

**Атрибут poster** определяет изображение, которое будет показываться во время загрузки видеофайла или пока пользователь не нажмет кнопку воспроизведения видео

**source** используется для указания расположения медийных ресурсов (файлов) для медиа-элементов

**track** используется в качестве дочернего элемента для **<audio>** и **<video>**, определяя текстовые дорожки

**embed** используется для загрузки и отображения объектов (видеофайлов, флеш-роликов, некоторых звуковых файлов и т. д.), которые исходно браузер не понимает

**param** используется для определения параметров для плагинов, встроенных в элементы <**object>**

**object** используется для вставки в HTML-документ мультимедиа объектов (аудио и видео файлов, Java апплетов, ActiveX, PDF и Flash приложений)

**iframe** применяется для создания «плавающего» фрейма и вставки его в обычный HTML-документ

11. Теги для отображения и форматирования текста

**h1, h2, h3, h4, h5, h6**  создание заголовков раздела

**p** создание нового абзаца

**br** для принудительного разрыва строки

**hr** определяет тематический разрыв контента на HTML странице, отображается как горизонт. линия

**wbr**  вызывает переход на новую строку, если текущая строка вышла за пределы окна браузера

**code** для выделения части текста моноширинным шрифтом

**em** для выделения текста курсивом, на который необходимо обратить внимание

**kbd** используется для обозначения текста, который набирается на клавиатуре или для названия клавиш

**samp**  определяет текст, который является результатом вывода компьютерной программы или скрипта

**strong** предназначен для выделения семантически важного фрагмента текста **полужирным**, на который нужно обратить внимание пользователя

**var** используется для выделения переменных компьютерных программ

**pre**  определяет блок в который помещают предварительно отформатированный текст

**dfn** выделяет специальный термин впервые появившийся в тексте

**cite** представляет название произведения

**abbr** указывает, что заключенный в нем текст является сокращенной формой длинного слова или фразы

**del**  используется для выделения текста, который был удалeн

**s** определяет текст, который больше не является правильным или актуальным

**b** отображает вложенный в него текст полужирным шрифтом

**i** используется для выделения текста курсивом. Является элементом физического форматирования текста

**sup**  используется для выделения текста, который был удалeн

**sub** определяет текст, который больше не является правильным или актуальным

**u** создает подчеркивание текста

**small** уменьшает размер вложенного в него шрифта на одну единицу, кроме случаев когда исходный текст уже имеет наименьший допустимый размер

**bdi** гарантирует, что в обратном направлении будет написан только тот текст, который заключен в элементе

**bdо** используется для переопределения текущего направления текста

**ins** определяет текст, который был добавлен в новой версии документа

**<ins cite = "http://redactor.com/info.html" datetime = "2018-05-30T10:00:00+03:00">**

**ruby** предназначен для добавления небольшой аннотации сверху или снизу от заданного текста

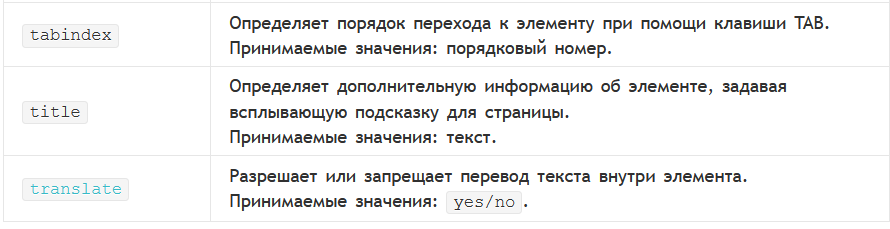
**rt** выводит аннотацию сверху или снизу от текста, заключенного в элемент <ruby>

**rp** выводит альтернативный текст для браузеров, не поддерживающих элемент <ruby>. В остальных браузерах текст в теге <rp> не отображается

12. Атрибуты HTML5. Специальные символы.

13. Глобальные атрибуты

**Глобальные атрибуты** — **это атрибуты общие для всех HTML-элементов**; они могут использоваться всеми элементами, хотя на некоторые элементы могут не оказывать влияния.

  
14. Атрибуты обработчиков событий

## События формы

События срабатывают на действия внутри HTML формы (могут применяться почти ко всем элементам HTML, но чаще всего применяются к элементам формы):

## События клавиатуры

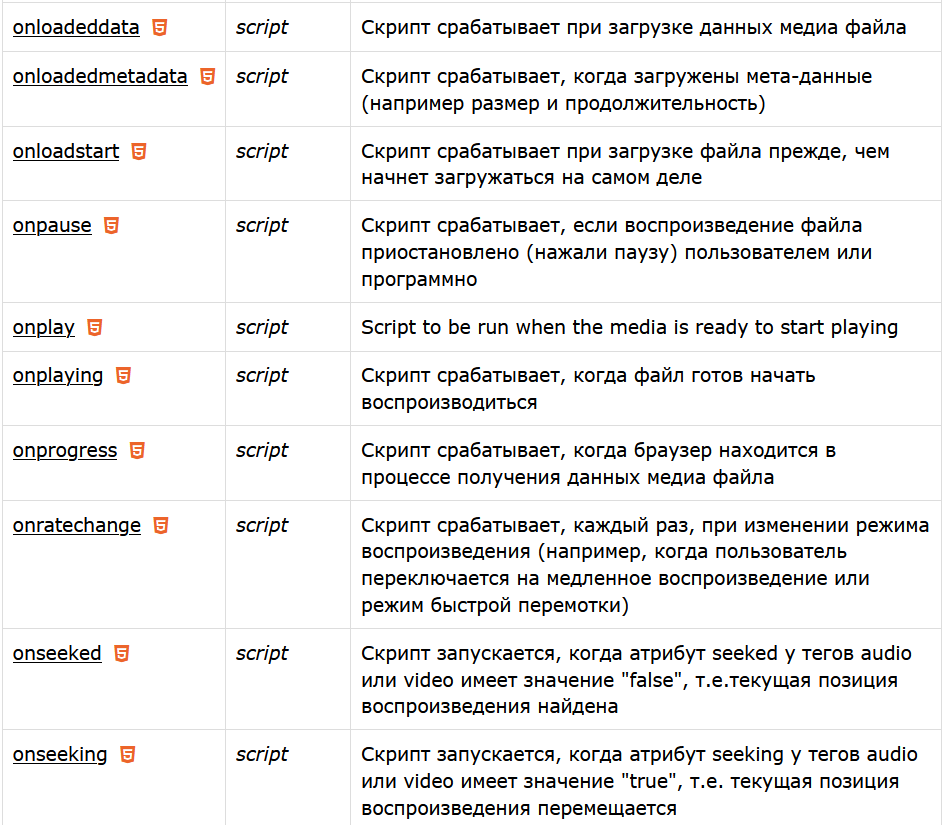
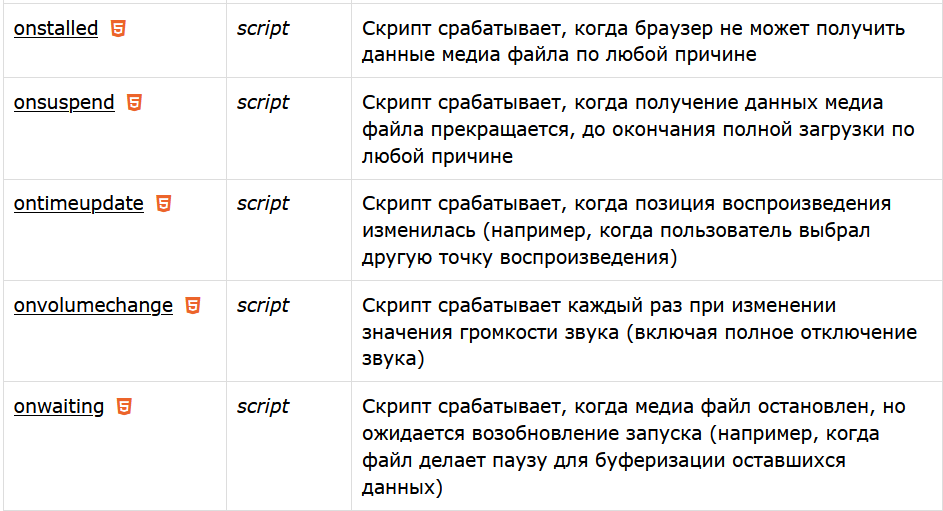
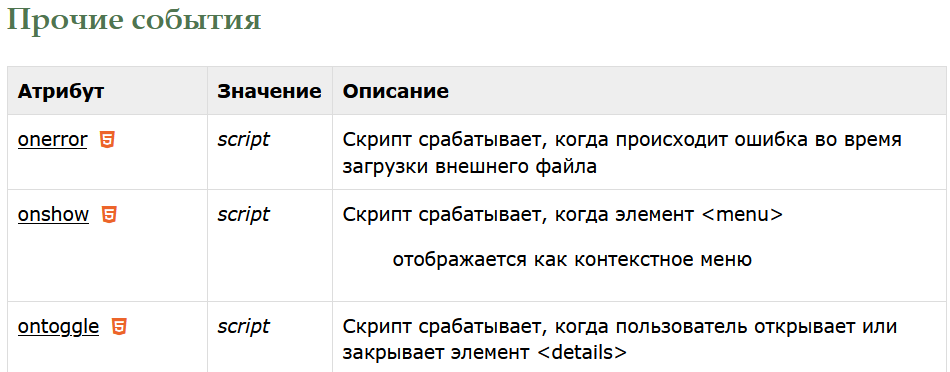
События, вызываемые клавиатурой (могут применяться ко всем элементам HTML):

## События мыши

События, вызываемые действиями мышкой или аналогичными действиями пользователя (могут применяться ко всем элементам HTML):

## События медиа-файлов

События, вызываемые для медиа файлов, таких как видео, изображения и аудио (могут применяться ко всем элементам HTML, но чаще всего применяются к таким элементам, как <audio>, <embed>, <img>, <object>, and <video>):

     
15. Атрибуты тегов формы-8 вопрос

16. Основы CSS3

CSS (Cascading Style Sheets)

Термин «каскадные таблицы стилей» был предложен Хокон Виум Ли в 1994 году

17 декабря 1996 г. CSS1

12 мая 1998 г. CSS2

22 июня 1999 г. CSS3

12 апреля 2011 CSS2.1

Каскадные таблицы стилей позволяют создавать правила, определяющие внешний вид контента

**CSS (Cascading Style Sheets)** — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML).

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

CSS поддерживает таблицы стилей для конкретных носителей, поэтому авторы могут адаптировать представление своих документов к визуальным браузерам, слуховым устройствам, принтерам, брайлевским устройствам, карманным устройствам и т.д.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: **селектора** и **объявления**. В HTML имена элементов нечувствительны к регистру, поэтому «h1» работает так же, как и «H1». Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

17. Создание стилей. Классификация селекторов.

1. С помощью тега <style> создаются внутренние таблицы стилей, который располагается в теге head. Данный тег закрывающимся.

2. Создание во внешнем файле и подключение с помощью тега link. Такие таблицы стилей называются внешними.

**<link rel="stylesheet" href="styles.css">**

3. С помощью атрибута **style**, создавая встроенные стили

**Селекторы — это один из фундаментальных механизмов CSS. Именно они определяют то, к каким элементам будут применены стили, указанные в фигурных скобках.**

Общий синтаксис CSS представляет собой следующий вид:

**СЕЛЕКТОР {СВОЙСТВО: ЗНАЧЕНИЕ;}**

В качестве базовых селекторов используются теги, классы и идентификаторы, но существуют другие типы. Примеры создания различных селекторов представлены в таблице 2.1

|  |  |  |
| --- | --- | --- |
| **Тип селектора** | **Пример создания** | **Использование** |
| Селектор тега | h1 {  font-family: Arial, sans-serif;  color: #CCCCFF;  } | <h1>Заголовок</h1> |
| Классы | .special {  color: #FF0000;  font-size: 12pt;  } | <p class=”special”>….</p> |
| Идентификатор | #banner {  background: #CC0000;  heigt: 300px;  } | <div id=”banner”>…</div> |
| Групповой селектор | h1, .copyr, #banner {color: red;} | <h1>….</h1>  <p class=”copyr”>….</p>  <div id=”banner”>…</div> |
| Универсальный селектор | \* {font-weigt: bold;} | ко всем элементам веб-страницы |
| Селектор потомков | li a {font-family: Arial;  .intro h2 {color: yellow;} | <div class=”intro”>  <h2>…</h2>  </div>  <li><a href=”…”> |
| Дочерние селекторы | body > h2 {color: green;} | <body>  <h2>…</h2>  <div>  <h2>…</h2>  </div>  </body> |
| Родственные селекторы | h2+p {color: green;}  h2 ~ p {color: green;} | + форматирует элемент **p,** который сразу следует за h2  ~ форматирует все элементы p родственные к h2 |

Селекторы, которые используются для применения стилей к определенным HTML-элементам называются селекторыми тегов (или селекторы элементов)

Групповой селектор позволяет применять стиль одновременно для нескольких элементов

Селекторы потомков используются для указания элемента по отношению к его родительскому элементу и применяется ко всем элементам потомка

Дочерний селектор позволяет определить конкретный дочерний и родительский элемент

Элемент, который следует сразу же за другим элементом, в HTML называется смежным родственным элемен­том того же уровня. Смежный родственный селектор использует знак + для соединения одного элемента с другим

Общий сборный родственный селектор позволяет используя ~ отформатировать все родственные элементы

Селектор класса используется, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега, но не связывать с конкретным HTML-элементом.

Для применения блока объявления класс используется атрибут class к элементу

Селектор идентификатора используется, когда необходимо определить стиль для уникального элемента и применения JavaScript-сценариев.

Все имена идентификаторов начинаются с #

Для использования блока объявления идентификатора используется атрибут id

Используется для применения стилей ко всем элементам страницы вместо указания каждого по отдельности. Селектор обозначается \*

Веб-дизайнеры используют универсальный селектор как способ очистки всего пространства вокруг блочных элементов

\* {

margin: 0;

padding: 0;

}

18. Псевдоклассы и псевдоэлементы

**Псевдоклассы —** определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также в зависимости от положения в DOM

**Псевдоэлементы**— позволяют задать стиль определенной части не относящийся к элементам документа

**Псевдоклассы** – это селекторы, которые определяют **состояние** уже существующих элементов, которое может меняться при определенных условиях.

**Псевдоэлементы** – это селекторы, которые определяют **область** **элементов**, которая изначально отсутствует в дереве документа. Эта область создается искусственно с помощью CSS.

Ключевое отличие между ними в том, что псевдоклассы определяют именно состояние элементов, которые уже существуют на странице, а псевдоэлементы создают области (искусственные элементы), которых изначально на веб-странице не было. Но и те и другие отсутствуют в исходном коде документа.

**ПСЕВДОКЛАССЫ:**

**a:link**— обозначает стиль непосещенных гиперссылок

**a:visited**  — обозначает стиль посещенных ссылок

**a:hover** — позволяет изменять вид ссылки, на которую посетитель навел указа-тель мыши

**a:active** — позволяет определить, как будет выглядеть ссылка во время выбора ее посетителем веб-страницы

**:focus**  — фокусировка и изменение стиля элементов, которые получили фокус с помощью щелчка кнопкой мыши, касания экрана или нажатия клавиши *Tab*

**:disabled**  — указывает на элементы формы, которые отключены

**:enabled**  — указывает на элементы, которые включены

**:first-child**  — выбирает первый дочерний элемент

**:last-child**  — выбирает последний дочерний элемент

**:only-child**  — единственный дочерний элемент

**:nth-child(odd)**  — выбирает все нечетные дочерние элементы

**:nth-child(even)**  — выбирает все четные дочерние элементы

**:nth-child(5n+4)**  — применение стиля к каждому пятому дочернему элементу, начиная с четвертого дочернего элемента

**:nth-last-child()**  — работает аналогично псевдоклассу **:nth-child()**, только в обратном порядке, начиная с конца

**:first-of-type**  — выбирает элемент, который является первым потомком родителя заданного тега

**:last-of-type**  — выбирает элемент, который является последним потомком родителя заданного тега

**:nth-of-type**  — выбирает элемент, который является n-ным потомком заданного тега

**:target**  — соответствует элементу, на который указывает идентификатор в URL документа

**:any-link** — указывает на элементы, которые используют псевдоклассы **:link** и **:visited**

**:lang()**  — позволяет задать стиль в зависимости от языка

**:optional**  — указывает на элементы input, которые не обязательны к заполнению (без атрибута required)

**:required**  — указывает на элементы input, которые обязательны к заполнению (указан атрибут required)

**:checked**  — указывает на элементы **input** (type=”radio / checkbox”) и **option** (внутри тега select), которые выбраны

**:not()**— позволяет выбрать что-либо отличное от другого

Псевдоэлементы:

**:first-letter**  — применение стиля к первой букве первой строки блочного элемента

**:before**  — создает псевдоэлемент и добавляет до выбранного элемента

**:after**  — создает пседоэлемент и добавляет его после выбранного элемента

**::selection**  — позволяет применить некоторые стили к элементам, которые были выделены пользователем

**:first-line**  — применение стиля к первой строке блочного элемента

**::marker**  — позволяет применить некоторые стили к маркерам списка или элементам со значением display: list-item;

**::placeholder**  — позволяет применить некоторые стили к тексту атрибута **placeholder** в элементах **input** и **textarea**

19. Механизм наследования стилей.

Это механизм, с помощью которого стили применяются не только к указанным элементам, но также к их потомкам. Наследование также надо понимать в этом контексте — некоторые значения свойства CSS, установленные для родительских элементов наследуются их дочерними элементами, а некоторые нет.

Не наследуются:

1. Свойства, которые затрагивают размещение элементов на странице (поля, фоновый цвет, границы элементов)

2. Встроенные стили браузеров для форматирования различных элементов

Свойства, которые наследуются:

**list-style, list­-style-type, list-style-position и list-style-image**

**color, font, font-family, font-size, font-weight, font-variant, font-style, line-height, letter-spacing, text-align, text-indent, text-transform, white-space и word-spacing**

Существует два специальных значения, которые можно применять к любому свой­ству, чтобы управлять наследованием стилей: **inherit** и **initial**.

20. Единицы измерения значений свойств CSS  
**Пиксели (px)** это также наименьшая единица растрового изображения, получаемого с помощью графических систем вывода информации

1mm (мм) = 3.8px

1cm (см) = 38px

1pt (типографский пункт) = 4/3 px

1pc (типографская пика) = 16px

**Относительная единица шрифта (em)** является относительной единицей равной элементу font-size. Так что, если font-size задан 16px, это значение 1em

**Отностиельная единица (rem)**равная вычисленному значению свойства font-size для основного текста. Если свойство font-size установлено для основного текста, то rem единицы связываются с этим свойством

Проценты (%)

* Родителю (общее правило).
* Последнему родительскому элементу, для которого значение position установлено relative, когда значние элемента становлено absolute.
* Корневому элементу, когда для position устанавливается fixed.
* Во всех современных браузерах, исключая IE8-, поддерживаются новые единицы из черновика стандарта CSS Values and Units 3:
* vw – 1% ширины окна
* vh – 1% высоты окна
* vmin – наименьшее из (vw, vh), в IE9 обозначается vm
* vmax – наибольшее из (vw, vh)
* Эти значения были созданы, в первую очередь, для поддержки мобильных устройств.
* Их основное преимущество – в том, что любые размеры, которые в них заданы, автоматически масштабируются при изменении размеров окна.

21. Каскадность стилей. Приоритет стилей  
**Каскадность**—набор правил, которые определяют последовательность применения стилей и устраняют возникающие между ними конфликты.

Определяется тремя параметрами:

1. Источник стиля

2. Специфичность селекторов

3. Исходный порядок

По убыванию приоритета стиля в зависимости от источника располагаются следующим образом:

1. Стиль браузера
2. Стиль автора
3. Стиль пользователя
4. Стиль автора с добавлением !important
5. Стиль пользователя с добавлением !important

Условная величина определяющая приоритет селектора. Имеет следующие значения:

1. Селектор тега = 1
2. Селектор класса = 10
3. Селектор идентификатора = 100
4. Строковый стиль = 1000
5. Псевдокласс = 10
6. Псевдоэлемент = 1
7. Селектор атрибута = 10

22. Конфликты стилей. Устранение конфликтов стилей. ----

23. Использование селекторов потомков

Наиболее распространённым дочерним селектором является селектор потомка, который соответствует каждому элементу, который следует за определённым предком. Потомок не обязательно должен идти сразу после предка в дереве документа, вроде отношения родитель-ребёнок, но может находиться в любом месте внутри предка. Селекторы потомка создаются пробелом между отдельными элементами в селекторе, создавая новый уровень иерархии для каждого элемента списка.

Селектор article h2 является селектором потомка и выбирает только элементы <h2>, которые находятся внутри элемента <article>. Обратите внимание, независимо от того, где живёт элемент <h2>, пока он находится внутри элемента <article>, он всегда будет выбран. Кроме того, любой элемент <h2> за пределами элемента <article> не будет выбран.  
24. Форматирование текста на CSS3: абзацев, символов и слов

Свойство *font-style* со значением ***italic*** позволяет выделяет текст курсивным, а для того чтобы сделать полужирным используется свойство *font-weight* со значением ***bold***.

Чтобы изменить регистр текста необходимо свойство *text-transform* со значением ***uppercase***. Сделать буквы строчными можно с помощью значения ***lowercase***, а превратить первые буквы в прописные — с помощью слова ***capitalize***.Для создания стиля с малыми прописными буквами используйте следующий код:

font-variant: small-caps;

Свойство *text-shadow* требует задания трех параметров: горизонтального смещения (насколько левее или правее текста должна отображаться тень), вертикальное смещение (насколько выше или ниже текста должна появиться тень), степень размытости тени и цвет отбрасываемой тени.

Для установки размера межстрочного интервала с помощью свойства *line-height* в процентах. Чтобы увеличить межстрочный интервал, т. е. распределить строки дальше друг от друга, используйте значение больше 120%.

Используя свойство *text-align* можно расположить абзац в центре (***center***) веб-страницы, вдоль левого (***left***) или правого (***right***) края или выровнять по ширине (формату) (***justify***).

Чтобы полностью избавиться от верхнего и нижнего полей используется следующий код:

p {margin-top: 0;  
margin-bottom: 0;}

Для того чтобы добавить абзацный отступ (отступ первой строки) используется свойтсво *text-indent*.

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также в зависимости от положения в DOM. Псевдоэлементы позволяют задать стиль определенной части не относящийся к элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

К псевдоэлементам относятся следующее:

1. ***first-letter*** позволяет оформить первую букву указанного элемента;

2. ***first-line*** позволяет оформить первую строчку указанного элемента;

3. ***before, after*** позволяет вставлять произвольное содержимое до и после указанных элементов.

В спецификации CSS3 псевдоэлементы обозначаются двойным символом двоеточия (::), но для лучшей совместимости с более старыми версиями используйте один символ двоеточия (:).

Форматирование текста  — это процесс оформления страницы, абзаца, строки, символа.

Основная цель форматирования — сделать восприятие готового документа простым и приятным для читателя.

Свойства форматирования текста:

Свойство **font-size** для изменения размера кегля шрифта

Свойство **font-family** для изменения гарнитуры шрифта

Свойство **font-style** для изменения начертания на курсивное

Свойство **font-weight** для изменения начертания на полужирное

Свойство **text-indent** отступ первой строки (абзацный отступ).

Свойство **text-align** выравнивание текста относительно страницы

Свойство **text-align-last** для выравнивания последней строки

Свойство **text-decoration** для изменения внешнего вида текста с добавлением различных эффектов

Свойство **color** для форматирования текста цветом

Свойство **text-transform** для изменения регистра (на заглавные или строчные символы)

Свойство **line-height** для изменения междустрочного интервала (интерлиньяжа)

Свойство **font-variant** для преобразования текста в малые прописные

Свойство **letter-spacing** для изменения интервала между символами (трекинг)

Свойство **word-spacing** для изменения интервала между словами

Свойство **text-shadow** для добавления тени

Свойство **hyphens** добавляет автоматически переносы

Свойство **writing-mode** определяет направление текста

Свойство **text-orientation** устанавливает ориентацию текстовых символов в строке и применяется только к вертикальному тексту

Свойство **text-rendering** предоставляет информацию механизму визуализации о том, что нужно оптимизировать при рендеринге текста

Антиквенные шрифты (с засечками) идеальны для длинных фрагментов текста, так как засечки хорошо для глаз, делая текст более удобочитаемым

**font-family: "Times New Roman",** **serif;**

Рубленые шрифты (без засечек) часто используются для заголовков благодаря простому и четкому внешнему виду.

**font-family: Arial,** **sans-serif;**

Моноширинный шрифт часто используется для отображения компьютерного кода. Каждая буква в моноширинном шрифте имеет одинаковую ширину (как буквы в механических печатных машинках).

**font-family: "Courier New",** **monospace;**

Декоративные шрифты особые шрифты, которые применяются для визуального выделения заголовков, текстовых эффектов и никогда для основного текста.

**font-family: "Comic Sans MS",** **fantasy;**

25. Использование веб-шрифтов. Использование службы Google Fonts.

Сначала скопируйте шрифты в папку на компьютере, где хранятся файлы для вашего сайта. Для этого можно создать выделенную папку в корневом каталоге сайта, которая носит имя fonts.

Правило **@font-face** позволяет подключать шрифты. Эта команда присваивает шрифту имя и сообщает браузеру, где найти файл шрифта для его загрузки. Например, в папке fonts находиться шрифт формата Web Open Font c именем Lobster

@font-face   
{font-family: 'Lobster'; /\* Задаем имя шрифта \*/

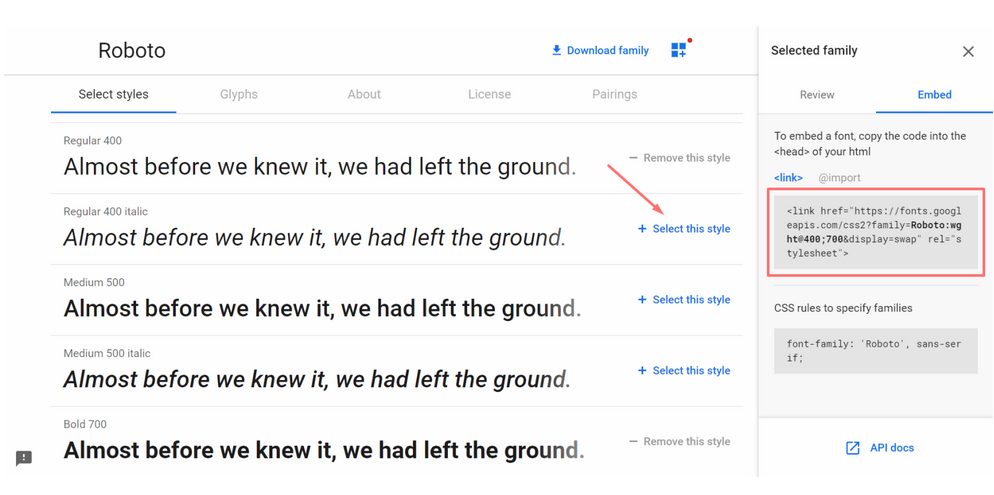
src:url('lobster.woff')/\* Указываем местонахождение нашего шрифта \*/ }

При использовании внутри правила **@font-face** свойство *font-family* присваивает шрифту имя. Затем это имя задействуется, когда соответствующий шрифт нужно применить к стилю. Например, нужно использовать шрифт Lobster для всех заголовков первого уровня на странице. Для этого можно будет указать следующий стиль:

p {font-family: 'Lobster';}

Если необходимо использовать несколько шрифтов, то потребуется несколько правил **@font-face**. WOFF в большинстве случаев пред-почтительнее, файлы формата SVG намного больше по размеру и поддерживаются только браузером Safari.

Есть простой и удобный способ подключить нестандартный шрифт — использовать [Google Fonts](https://fonts.google.com/). Это бесплатный сервис, с помощью которого можно подключать шрифты, но не хранить их на своём сервере. Чтобы им воспользоваться, необходимо выбрать шрифт и добавить одно или несколько нужных начертаний, а затем вставить в <head> ссылку, которую сгенерирует Google Fonts во вкладке Embed.



Этого достаточно, чтобы шрифт подключился. Теперь его можно использовать в свойстве font-family без каких-либо дополнительных действий.

body {

font-family: "Roboto", "Arial", sans-serif;

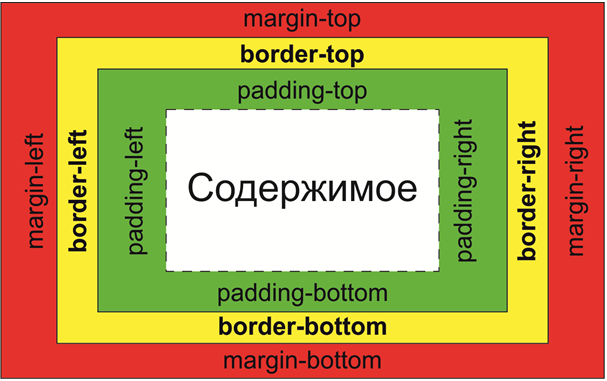
}

Стоит отметить, что после основного шрифта важно указать веб-безопасный. В случае, если нестандартный шрифт не загрузится, браузер воспользуется альтернативным. Его нужно [подобрать](https://meowni.ca/font-style-matcher/) максимально похожим на основной шрифт.

Плюсы этого способа — простота использования и всегда актуальные версии шрифтов, Google Fonts их часто обновляет. Но у него есть и минус — запросы к сторонним серверам могут негативно повлиять на скорость загрузки страницы. Если выбираете этот способ, стоит обратить внимание на [оптимизацию](https://csswizardry.com/2020/05/the-fastest-google-fonts/).

26. Понятие блочной модели.

**Блочная модель** структура HTML-элемента, которая состоит из области содержимого с отступами, границами и полями вокруг него



**кБлочные элементы** создают разрыв строки перед элементом и после него

**Строчные элементы** не создают разрывов строк ни до, ни после себя

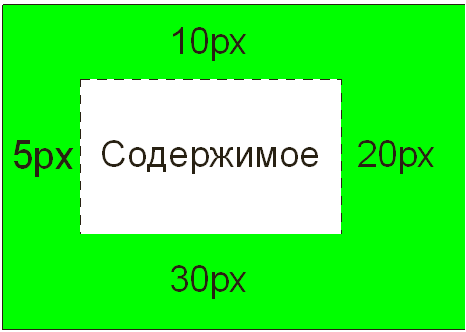
**padding** — отступ между содержимым и границей;

**border**  — граница элемента;

**margin**  — поля после границы и до следующего элемента;

Сокращенная запись padding

**padding:** **10px 20px 30px 5px;**



**Свойства границ**

Можно управлять тремя различными свойствами любой из границ: *color* (цвет), *width* (ширина) и *style* (стиль). Для ширины границы width используются любые единицы измерения каскадных таблиц стилей (кроме процентов) или ключевые слова ***thin*** (тонкая линия), ***medium*** (средняя) и ***thick*** (толстая). Самые распространенные единицы измерения для данного свойства — пикселы.

Свойство *style* управляет типом линии границы. В каскадных таблицах стилей для границ имеются следующие стили: ***solid, dotted, dashed, double, groove, ridge, inset, outset, noneи hidden***. Ключевые слова none и hidden работают одинаково: они полностью удаляют границы. Но значение none удобно использовать для удаления границы с одной стороны элемента. Для установки границ можно использовать сокращенную запись или расширенную следующим образом:

border: 2px double #FFCC33;

или

border-width: 2px;

border-style: double;

border-color: #FFCC33;

Следует отметить, что каждая сторона имеет свой набор из трех свойств, которые удобно использовать для отмены одного. Правая граница *border-right-width*, *border-right-style* и *border-right-color*. Левая, верхняя и нижняя границы имеют похожие свойства: border-left-width, border-left-style и т. д. Однако можно задать собственные значения сразу для каждой стороны границы, используя сокращенную запись свойства *border-width*, *border-style* и *border-color*. Например, правило ***border-width: 10px 5px 15px 13px;*** применит четыре различных значения ширины для каждой из сторон (верхней, правой, нижней и левой).

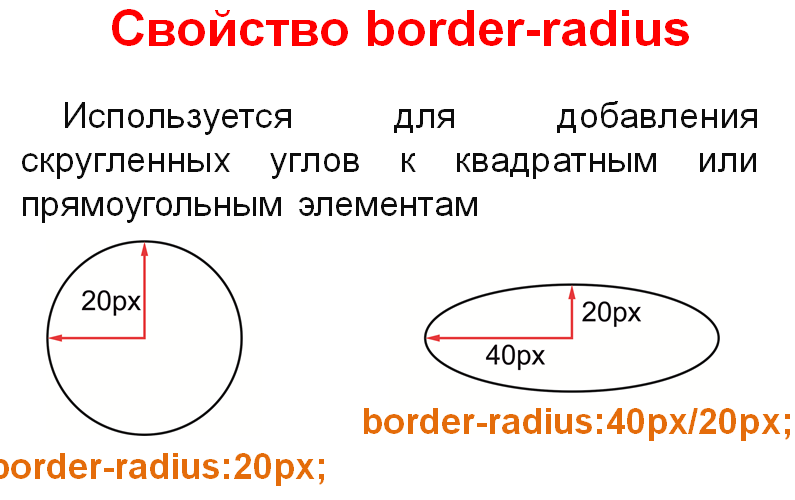
В языке CSS существует свойство *border-radius*, позволяющее дизайнерам добавлять скругления к одному или нескольким углам элемента. Для каждого угла можно предоставить отдельные значения, задав четыре параметра. Объявление свойства имеет следующий вид:

border-radius: 0 30px 10px 5px;

Сначала задается числовое значение для левого верхнего угла блока, а затем по часовой стрелке для всех остальных углов.

Чтобы добавить эллиптические углы, нужно создать следующее объявление:

border-radius: 40px/20px;

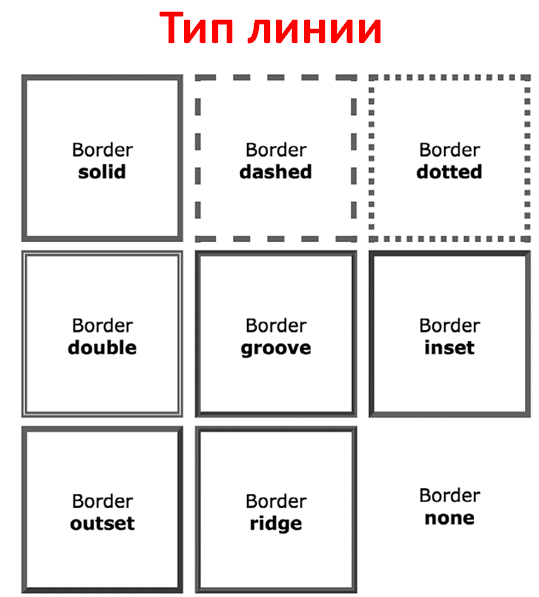


**border: 1px solid green;**

**1px** — толщина границы;

**solid** — графический тип линии;

**green** — цвет границы.



**Ширина блочных элементов** = **width + padding\*2 + margin\*2 + border\*2**

**.box {**

**width: 100px;**

**padding: 15px;**

**border: 5px;**

**margin: 10px;**

**}**

Ширина блока составляет 160px

**Размеры блочных элементов**

Браузеры вычисляют ширину блочного элемента, складывая значения свойств *border*, *padding* и *width*. Таким образом, необходимо производить математические вычисления для определения занимаемого экранного пространства. На этот случай каскадные таблицы стилей предлагают свойство *box-sizing*, позволяющее изменить порядок вычисления браузером экранной ширины (и высоты) элемента. Свойство *box-sizing* имеет следующие значения:

1. ***content-box*** добавляет ширину границ и толщину отступа к значениям, установленным для свойств ширины и высоты, чтобы определить экранную ширину и высоту заданного элемента.

2. ***padding-box*** сообщает браузеру, что при установке для стиля свойства ширины или высоты должны включать отступы как часть своего значения

3. ***border-box*** сообщает браузеру о необходимости включения в качестве состав-ляющей части значений свойств *width* и *height* толщину как отступа, так и границ. Это значение можно использовать для универсального селектора.

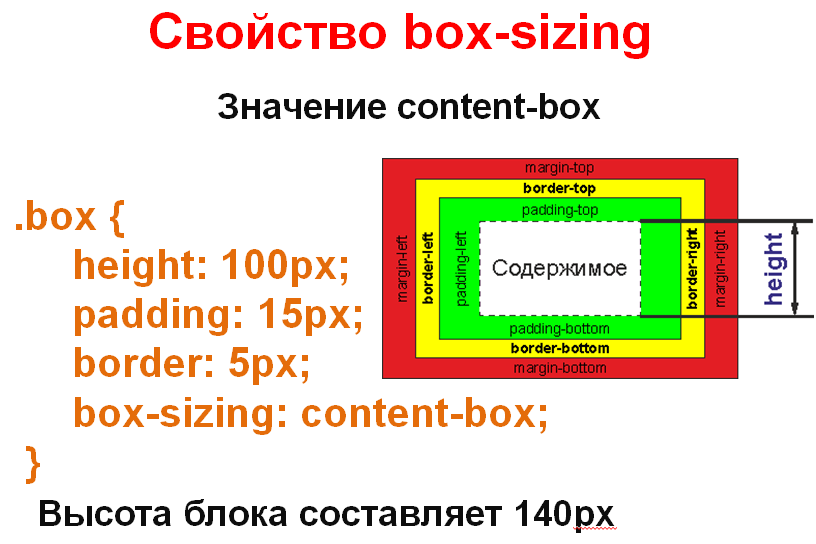
Когда содержимое форматируемого элемента имеет размеры больше определенных свойствами *width* и *height*, то используется свойство *overflow* со следующими значениями:

– ***visible*** имеет тот же эффект, что отсутствие установки свойства;

– ***scroll*** позволяет добавить полосы прокрутки.

– ***auto*** выполняет ту же функцию, что и ***scroll***, но полосы прокрутки в данном случае появляются только при необходимости.

– ***hidden*** скрывает любое содержимое, выходящее за пределы блочного элемента.







**Единицы измерения:**

**width: 300px;**

**width: 30%;**

**height: 20em;**

Поля: Отступы:

**margin-top padding-top**

**margin-right padding-right**

**margin-bottom padding-bottom**

**margin-left padding-left**

Свойство **display**:

Значение **block** позволяет изменить свойства строчного элемента на блочные

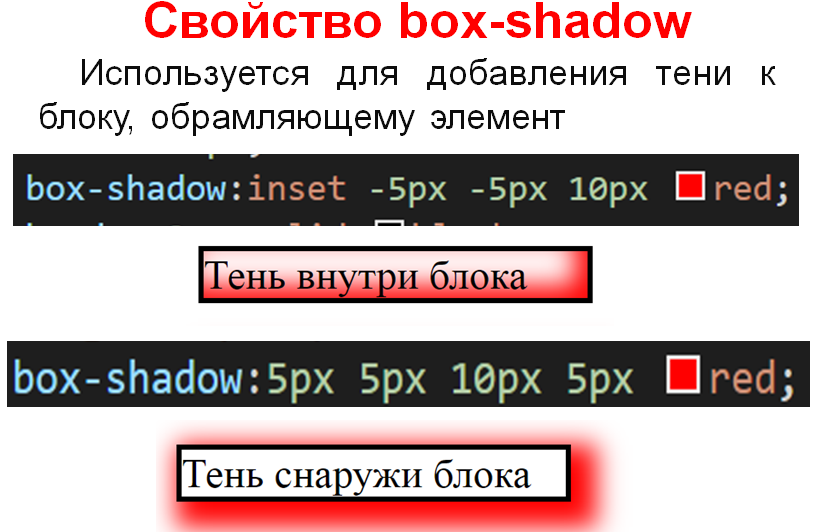
Значение **inline** позволяет изменить свойства блочных элементов на строчные

Значение **inline-block** позволяет придать элементу свойства строчных и блочных элементов

Плавающие элементы:

Свойство **float** перемещает любой элемент в нужную позицию, выравнивая его по левому или правому краю веб-страницы.

В процессе перемещения содержимое, находящееся ниже позиционируемого плавающего элемента, смещается вверх и плавно обтекает сам плавающий элемент



Свойство overflow управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров.  
27. Форматирование таблиц

Для форматирования таблиц можно использовать следующие псевдоклассы:

1. ***first-child*** позволяет оформить первый дочерний элемент;

2. ***last-child*** позволяет оформить последний дочерний элемент;

3. ***nth-child(odd)*** позволяет оформить чередующиеся четные дочерние элементы;

4. ***nth-child(even)*** позволяет оформить чередующиеся нечетные дочерние элементы.

Например, если нужно предоставить один фоновый цвет для каждой четной строки таблицы и другой фоновый цвет для каждой нечетной, можно создать два следующих правила:

tr:nth-child(odd) { background-color: #000000; }

tr:nth-child(even) { background-color: #FFFFFF; }

**Свойства форматирования таблиц**

Текстовая информация таблицы располагается в ячейках, которые имеют границы. Для изменения отступа от границ до содержимого ячеек используется свойство *padding.* Например, чтобы задать отступ размером 10 пикселов для всех ячеек таблицы, следует воспользоваться следующим стилем

td, th { padding: 10px; }

Можно также контролировать отступы от каждой стороны ячейки или использовать сокращенную запись:

td {

padding-top: 10px;

padding-right: 5px;

padding-bottom: 3px;

padding-left: 5px;

}

td {

padding: 10px 5px 3px 5px;

}

Чтобы настроить расположение содержимого внутри ячейки, используются свойства *text-align* (для выравнивания по горизонтали) и *vertical-align* (для выравнивания по вертикали). Свойство *vertical-align* принимает значения **top** (выравнивание по верхнему краю), **baseline**, **middle** (выравнивание по центру) или **bottom** (выравнивание по нижнему краю). При указании значения **baseline** выравнивание происходит так же, как и при установке значения **top**, за исключением того, что браузер выравнивает первую строку текста в каждой ячейке заданной строки таблицы.

Для создания границ используется свойство *border*, свойство   
*border-spacing* для управления размером этого промежутка между ячейками, который образуется при создании таблицы.

Если убрать промежутки между ячейками, границы, заданные для ячеек, будут удваиваться. Чтобы избавится от этого (а также от промежутков между ячейками) необходимо использовать свойство *border-collapse*. Значение **separate** эквивалентно тому, как обычно и отображаются таблицы: с промежутками между ячейками и двойными границами. Значение **collapse** позволяет избавиться удвоения границ.

Чтобы добавить к ячейкам таблиц (но не к самим таблицам) скругленные углы необходимо свойство *border-radius*.

Для того, чтобы скрыть пустые ячейки используется свойство ***empty-cells*: hide** к стилю, примененному к таблице:

table {

empty-cells: hide;

}

Если свойству *border-collapse* присвоить значение **collapse**, браузеры проигнорируют свойство *empty-cells* и *border-radius*.

Свойство **padding** добавление отступов до границы ячейки

Свойство **text-align** выравнивание текста в ячейке по горизонтали

Свойство **vertical-align** выравнивание текста по вертикали

Свойство **border-spacing** управление расстоянием между ячейками

Свойство **border-collapse** позволяет удалить интервалы удвоение границ между ячейками

Свойство **border-radius** для добавления скругления углов к ячейкам

Свойство **empty-cells** скрывает пустые ячейки

28. Форматирование веб-форм

**Свойства форматирования веб-форм**

По умолчанию элементы **label** и **span** — строчные, игнорирующие многие настройки, доступные блочным элементам, включая свойства *width*, *height* и *text-align*. Но если превратить метки в блочные элементы (с помощью значения **inline-block** свойства *display*), то они по-прежнему будут располагаться рядом с элементами веб-формы (оправдывая свое строчное происхождение) и не будут игнорировать свойства блочных элементов.

Для того, чтобы выровнять текст метки по правому краю, чтобы каждая метка отображалась рядом с соответствующим элементом веб-формы и увеличить поле справа от метки можно использовать следующие свойства:

.label {

display: inline-block;

width: 20em;

vertical-align: top;

text-align: right;

margin-right; 15px;

}

Псевдокласс **:focus** позволяет создать селектор, изменяющий внешний вид текстового поля при щелчке на нем кнопкой мыши или при переходе на него нажатием клавиши Tab (это называется фокусировкой на элементе веб-формы).

Чтобы выбрать только текстовые поля в форме веб-страницы, используйте следующее выражение: **input[type="text"]**, которое представляет собой селектор атрибутов.

1. Псевдоклассы :focus, :checked, :enabled, :optional,:disabled, :required, :read-write, :read-only

2. Свойство **display** для добавления к элементам **label** свойств блочных элементов

3. Псевдоклассы :valid и :invalid для определения валидности вводимой информации, например для email

29. Добавление фоновых изображений. Использование градиентных фонов.

**Фоновые изображения**

Для того чтобы установить фоном цвет используется свойство *background-color*, а сделать изображение фоном применяется свойство *background-image*. Свойство принимает единственное значение: ключевое слово **url**, за которым следует путь к графическому файлу, заключенный в круглые скобки. Можно использовать абсолютный URL-адрес, например, url:(http://www.cosmofar-mer.com/image/bg.gif) или относительный путь от документа или корневого каталога сайта.

url(../images/1.jpg) /\* относительный путь от документа \*/

url(/images/1.jpg) /\* относительный путь от корневого каталога \*/

Современные браузеры работают только с четырьмя графическими форматами: GIF, JPEG, PNG и SVG.

Если свойство *background-image* используется без указания дополнительных свойств, то фоновое изображение многократно повторяется заполняя всю веб-страницу. На этот случай можно воспользоваться свойством *background-repeat*, чтобы определить, каким образом будет повторяться фоновое изображение. Свойство может принимать четыре значения:

**repeat** — параметр по умолчанию, обеспечивает повторное отображение фонового изображения слева направо и сверху вниз до полного заполнения всего пространства веб-страницы;

**no-repeat** — отображает фоновое изображение без повторения;

**repeat-x** — вызывает повторение фонового изображения горизонтально вдоль оси X;

**repeat-y** — повторяет фоновое изображение вертикально вдоль оси Y;

**round** — фоновое изображение, как и предыдущие параметры, но не обрезает его.

**space** — повторяет фоновое изображение, но предотвращает искажение и обрезание изображений вверху или внизу.

Для управления расположением графики используется свойство позиционирования фонового изображения *background-position*. Свойство должно принимать два значения: одно из них для горизонтального позиционирования: left, center, right, а другое — вертикального: top, center, bottom.

background-position: center center;

background-position: right top;

Позиционировать фоновые изображения можно, используя точные значения в пикселах или единицах em.

Начальную позицию изображения можно изменить, воспользовавшись свойством *background-origin*. Этому свойству присваивается одно из трех значений:

**border-box** — изображение помещается в верхний левый угол области границы;

**padding-box** — изображение помещается в верхний левый угол области отступа, это место по умолчанию;

**content-box** — изображение помещается в левый верхний угол области содержимого.

Кроме того, свойство *background-origin* может эффективно использоваться с другим свойством — *background-clip*, которое ограничивает область появления фонового изображения. Используется три значения:

**border-box** — позволяет рисунку отображаться позади содержимого и любых границ;

**padding-box** — дает возможность ограничить любое фоновое изображение областью отступов и содержимого элемента;

**content-box** — позволяет ограничить фоновое изображение областью содержимого элемента.

Чтобы установить размер фонового изображения используется свойство *background-size*.

С помощью свойства *background-attachment*, которое может принимать два значения: scroll и fixed. Значение по умолчанию **scroll** определяет такое поведение браузера, при котором фоновое изображение прокручивается вместе с текстом и другим контентом. Значение **fixed** предотвращает перемещение, жестко фиксируя его на заднем плане.

Следует отметить, что для использования изображения как типа маркера можно применить свойство background-image, например, следующим образом:

.spisok li {

list-style: none;

background-image: url(images/2.png);

background-repeat: no-repeat;

background-position: 0 4px;

padding-left: 25px;

margin-bottom: 10px;

}

**Градиентные фоны**

Градиент — это плавный переход цвета, к примеру, от синего к красному или от черного к белому.

Линейный градиент распространяется по прямой от одного конца элемента к другому, демонстрируя плавный переход от одного цвета к другому. Необходимо только указать его направление в градусах или с помощью значений top, bottom, right, left либо их комбинаций. Напри-мер, чтобы нарисовать градиент, изменяющий цвет от черного к белому от левого угла к правому, нужно воспользоваться следующим кодом:

background-image: linear-gradient(to right, black, white);

или

background-image: linear-gradient(90deg, black, white);

Значения в углах повторяют часовую стрелку на циферблате, поэтому значение 90deg соответствует правому краю элемента (right), 180deg — нижнему краю элемента (bottom), а 270deg — левому краю элемента (left) и 0deg — значению top.

Также можно составлять составной градиент в процентах из нескольких цветов.

background-image: linear-gradient(to right, red 20%, orange 30%, orange 70%, red 80%);

Это означает, что первые 20 % протяженности элемента (слева направо) будут иметь фон из сплошного красного цвета. Затем, от 20%-ной до 30%-ной точки градиент плавно перейдет от красного к оранжевому цвету.

Также можно использовать transparent для любого цвета в градиенте, чтобы видеть область, расположенную позади него, например фоновый цвет элемента или другой линейный градиент.

background-image: linear-gradient(cyan, transparent),

linear-gradient(225deg, magenta, transparent),

Чтобы получить повторяющийся линейный градиент следует написать такой код:

background-image: repeating-linear-gradient(45deg, #900 20px, #FC0 30px, #900 40px)

Радиальный градиент распространяется наружу по круговой или эллиптической схеме. Необходимо только задать начальный цвет (цвет в середине градиента) и конечный (цвет в конце).

background-image: radial-gradient(red, blue);

Чтобы создать круговые градиенты, добавить перед указанием цветов circle. Значения позиционирования центра указываются перед значениями, определяющими форму и цвета. Например, ***background-image: radial-gradient(circle at 20% 40%, red, blue)***.

C помощью свойства background-blend-mode можно применять режимы смешивания, которые используются в Adobe Photoshop.

Если градиенты не поддерживаются браузером требуется добавить префиксы: для IE10+ требуется префикс ***-ms***, для Chrome и Safari префикс ***-webkit***, для Opera префикс ***-o*** и для Firefox префикс ***–moz***, например, -ms-radial-gradient.

Свойства background:

Значение **linear-gradient**  создает линейный градиент

Значение **repeating-linear-gradient** для создания повторяющегося линейного градиента

Значение **radial-gradient** создает радиальный градиент

Значение **repeating-radial-gradient** для создания повторяющегося радиального градиента

30. Построение навигационной системы сайта.

**Вертикальная панель навигации**

Для создания вертикальной панели навигации можно использовать маркированный или нумерованный список и заключается в следующих этапах:

1. Создать список со ссылками следующим образом:

<ul class="nav">

<li><a href="first.html">Первая страница</a></li>

<li><a href="second.html">Вторая страница</a></li>

</ul>

2. Удалить маркеры списка.

ul.nav {

list-style-type: none;

}

3. Изменить ссылку до блочного элемента, так как элемент **a** является строчным элементом.

ul.nav a {

display: block;

}

Это позволяет для ссылок определить границы, отступы, поля.

4. Ограничить ширину и высоту элементов, т.к. их ширина ссылки равна ширине окна браузера

**Горизонтальная панель навигации**

Создание горизонтальной панели навигации можно осуществлять двумя методами. Первый заключается в применении свойства *float* и создании из ссылок плавающих элементов, которые обтекают друг друга с слева. Второй способ состоит в создании строки ссылок с помощью установки свойства ***display: inline-block***.

1. Создать маркированный спискок:

2. Преобразовать пункты списка в строчные элементы

.nav li { display: inline; }

или в плавающие элементы

.nav li { float: left; }

3. Для ссылок установить *display: inline-block* и другие свойства например:

.nav a {

display: inline-block; /\*для плавающих элементов значение block\*/

border: 1px solid black;

padding: 5px 15px 5px 15px;

background-color: yellow;

text-decoration: none;

color: blue;

}

**Панель навигации с выпадающим меню**

Чтобы создать панель навигации с выпадающим меню при наведении мыши можно использовать следующий СSS:

.nav ul {

margin:0;

padding:0;

list-style-type:none;

}

.nav>ul>li {

float:left;

width:180px;

position: relative;

}

.nav li a {

display: block;

background: yellow;

border: 1px solid black;

padding: 8px;

text-decoration: none;

}

.nav li .second {

display: none;

position: absolute;

top: 100%;

left:0;

}

.nav li:hover .second{

display: block;

}

.nav li li {

width:180px;

}

Следует отметить, что для данного набора стилей используется следующий вложенный список:

<nav class="nav">

<ul>

<li><a href="1.html">Первая страница</a>

<ul class="second">

<li><a>1.1</a></li>

<li><a>1.2</a></li>

<li><a>1.3</a></li>

</ul>

</li>

<li><a href="2.html">Вторая страница</a></li>

</ul>

</nav>

Для вложенного списка задается абсолютное позиционирование, а для основного списка задается относительное позиционирование. Относительное позиционирование необходимо для того, чтобы меню размещалось прямо под кнопкой раскрывающегося списка.

Чтобы выпадающее меню скрывалось используется свойство *display* со значением **none**. При наведении ссылки появляется благодаря значению **block**. Также можно использовать свойство ***visibility:hidden;*** для скрытия меню и ***visibility:visible;*** для отображения.

31. Форматирование ссылок

**Форматирование гиперссылок**

Для форматирования гиперссылок используются псевдоклассы, учитывающие их состояние:

1. ***:link*** — непосещенная ссылка;

2. ***:visited*** — посещенная ссылка;

3. ***:hover*** — ссылка, на которой находится указатель;

4. ***:active*** — ссылка в момент нажатия.

5. ***:focus*** — состояние ссылки при нажатии клавиши Tab.

Чтобы убрать стандартное подчеркивание используется свойство ***text-decoration*** cо значением **none**: a {text-decoration: none;}.

32. Использование ролловеров. Метод CSS-спрайт  
**Использование ролловеров**

Для смены одной графической ссылки на другую при наведении на нее указателя мыши применяется метод CSS­спрайт, который использует единственное изображение для представления различных состояний одной навигационной кнопки. Реализация метода заключается в следующем:

1. В программе редактирования изображений создается один рисунок с различными вариантами.

2. Измеряется расстояние от верхнего края получившегося комбинированного изображения до верхнего края каждого следующего изображения.

3. Создается стиль для ссылки в обычном ненажатом состоянии

a {background: url(facultet.png) no-repeat 0 0;}

4. Создается стиль с псевдоклассом ***:hover***

a:hover {background-position: 0 -39px;} /\*второе значение соответствует расстоянию измеренному в пункте 2\*/

Чтобы для отдельного элемента панели навигации, созданного из списка появлялось свое отображение, то необходимо использовать следующий вид записи:

/\*CSS\*/

.nav li a {background: url(facultet.png) no-repeat 0 0;}

.nav .pim a:hover {background-position: 0 -90px;}

/!-- HTML--/

<nav class="nav">

<ul>

<li class="pim"><a href="3.html">ХТиТ</a></li>

33. CSS-переходы.  
**Переходы**

Переход представляет собой анимацию смены одного набора свойств CSS другим за определенный промежуток времени. Переходы задаются с помощью свойств *transition*. Свойство *transition-property* указывает, в каких свойствах должен присутствовать переход. Значение **all** означает, что переход задается для всех свойств, которые изменяются. Свойство *transition-duration* указывает продолжительность перехода к конечному значению, например, 0.5s.

Для замедления или ускорения перехода используется свойство *transition-timing-function* со значениями linear, ease-in и ease-out:

1. **linear** — переход изменяется с постоянной скоростью;

2. **ease-in** — изменение вначале протекает медленно, но ускоряется до самого конца перехода;

3. **ease-out** — изменение начинается быстро, но к концу перехода замедляется.

Также можно использовать значение **cubic-bezier**, которое определяет график изменения перехода по кривой Безье.

Для задержки перехода используется свойство *transition-delay* со значением, например, 0.5s. Для создания анимации при наведении мыши на элемент блока плавного перехода изменения цвета с зеленого на синий:

.div {

background-color: rgb(255,0,0);

transition: background-color 1s linear;

}

.div:hover {

background-color: rgb(0,0,255);

}

34. CSS-преобразования.

**Преобразования**

К свойствам преобразованиям объекта относится вращение, масштабирование, смещение и наклон. Преобразование осуществляется с использованием свойства *transform*. Свойство имеет следующие значения:

1. **scale(0.5)** — увеличить или уменьшить в размерах элемент (например, для значения 0.5 в два раза);

2. **rotate(45deg)** — повернуть элемент на определенный угол заданный в deg;

3. **translate(40px, 60px)** — смещение элемента из его текущей позиции на некоторое расстояние вправо на 40px и вниз на 60px;

4. **skew(15deg)** — изменяет форму элемента, смещая верхний край элемента в одну сторону, а нижний — в противоположную.

По умолчанию точкой трансформации является центр элемента, но ее можно изменить с помощью свойства *transform-origin*.

Вращения и перемещения могут производиться во всех трех измерениях — X, Y и Z. Значение *rotateZ* эквивалентно *rotate*, так как выполняет вращение вокруг оси Z. Значение *rotateX* и *rotateY* вращают элемент вокруг горизонтальной оси X (наклоняя элемент вперед или назад) и вертикальной оси Y (поворачивая элемент вправо и влево) соответственно.

Также можно задать значение перспективы используя трансформацию **perspective()** или же свойство *perspective*. В качестве значения указывается расстояние от плоскости монитора до точки сходимости линий. Чем меньше значение, тем более выраженной выглядит перспектива. По умолчанию перспектива отображается так, как если бы наблюдатель находился прямо по центру элемента. Свойство *perspective-origin* смещает вправо или влево, вверх или вниз отображение перспективы.

Свойство *transform-style* позволяет располагать элементы в трехмерном пространстве, задавая значение **preserve-3d**.

### 35. Функции 3D-преобразований.

36. CSS-анимация.  
**Создание анимации**

Создание анимации проходит в два этапа.

1. Определение анимации. Включает настройку ключевых кадров со списком анимируемых CSS-свойств.

2. Применение анимации к элементу.

Кадры определяются с помощью правила @keyframes, структура которого имеет следующий вид:

@keyframes имяАнимации {

from {

/\* здесь перечисляются свойства CSS \*/

}

to {

/\* здесь перечисляются свойства CSS \*/

}

}

В блоке объявления селектора элемента, чтобы анимация к нему применялась используются следующая расширенная запись:

.anime {

animation-name: имяАнимации;

animation-duration: 2s;

animation-timing-function: ease-in-out;

animation-delay: 5s;

animation-iteration-count: 2;

animation-direction: alternate;

animation-fill-mode: forwards; }

Свойство *animation-name* задает имя анимации, определяемое правилом @keyframes, *animation-duration* задает длительность анимации, *animation-timing-function* задает временную функцию, описывающую ускорение и/или замедление воспроизведения анимации, *animation-iteration-count* задает количество повторений анимации, *animation-direction* и задает обратное воспроизведение анимации при повторном воспроизведении.

Чтобы анимация была бесконечной необходимо использовать значение infinite для свойства *animation-iteration-count.* Следует отметить, что для работы анимации необходимо задавать продолжительность и имя анимации.

Для того, чтобы использовать следующую запись, необходимо изначально для элемента задать свойство *position* со значением **absolute** или **relative**.

@keyframes around {

0% { left: 0; top: 0; }

25% { left: 200px; top: 0; }

50% { left: 200px; top: 200px; }

75% { left: 0; top: 200px; }

100% { left: 0; top: 0; }

}

p {

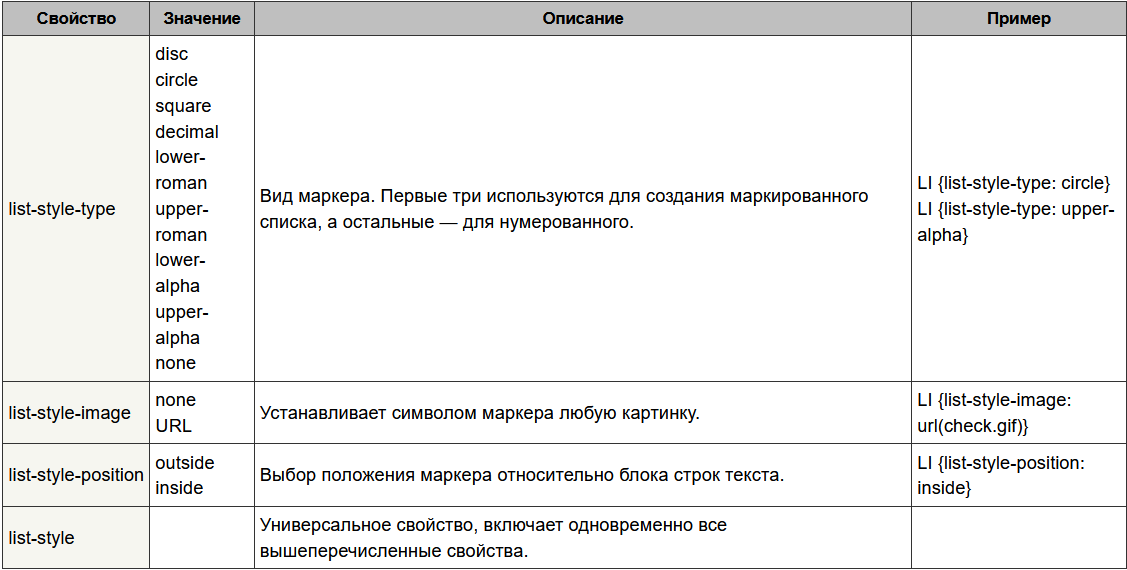
position: relative;

animation: around 4s linear infinite; }

37. Форматирование списков перечисления.  
Свойство **list-style-type** для изменения маркера списка

Свойство **list-style-image** для использования изображения как маркера

Свойство **list-style-position** для позиционирования маркера внутри или снаружи текста списка



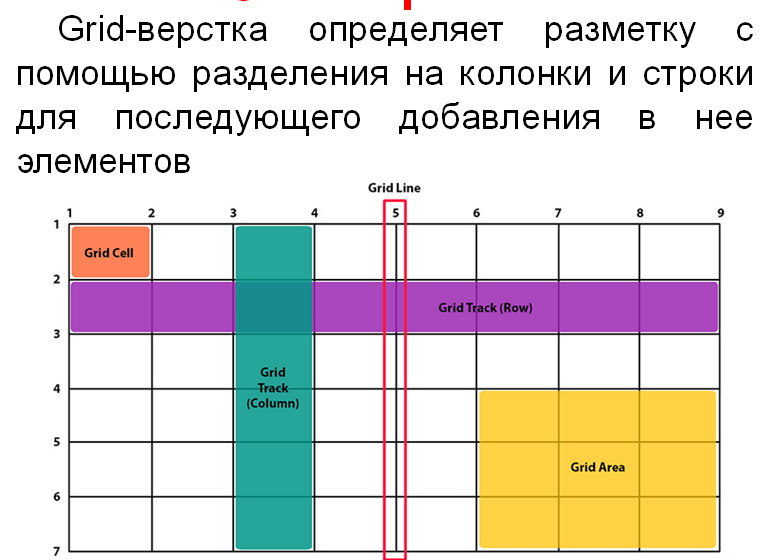
38. Системы модульной верстки.

Системы модульных сеток:

1. **Bootstrap**
2. **Skeleton**
3. **Foundation**
4. **Pure**
5. **Simple Grid**

39. Система модульной верстки Skeleton.

40. Структурирование HTML-кода под модульную сетку и ее использо-  
вание.  
41. Основы Grid-верстки.



Свойство **display:grid;** преобразует HTML-элемент в контейнер сетки

Свойство **grid-template-columns** устанавливает размер каждой колонки

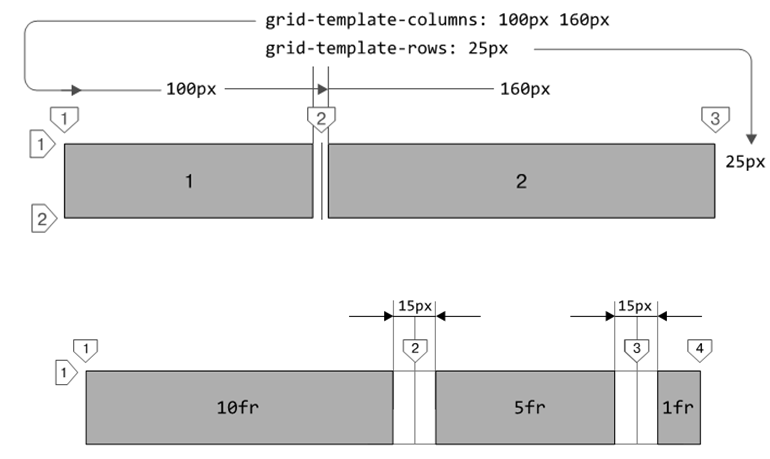
Свойство **grid-template-rows** устанавливает размер каждой строки

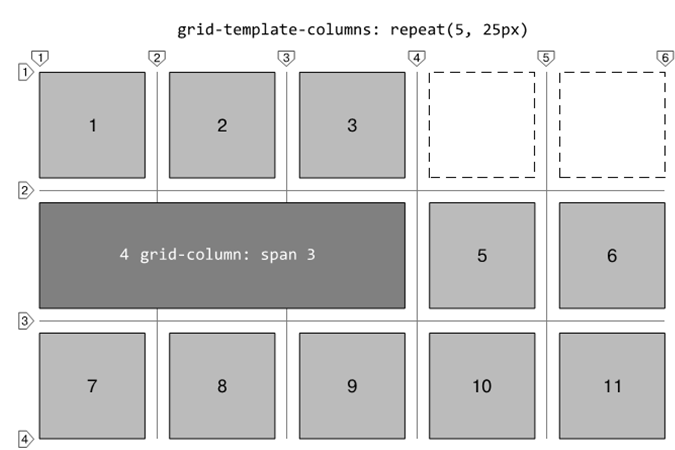
Свойство **grid-gap** устанавливает промежуток между ячейками сетки

Свойство **grid-auto-rows**   
устанавливает автоматически размер каждой строки

Свойство **grid-auto-columns** устанавливает автоматически размер каждой колонки

Свойство **grid-auto –flow** определяет автоматически распределение элементов в сетке





Свойство **grid-template-areas** составляет сетку в зависимости от имени каждой ячейки

Свойство **grid-area** устанавливает имя для каждой ячейки каждой сетки

Свойство **grid-column** определяет местоположение элемента по номеру колонки

Свойство **grid-row** определяет местоположение элемента по номеру строки

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

Свойство **justify-self** определяет местоположение контента ячейки по горизонтали

42. Медиазапросы. Создание адаптивного дизайна

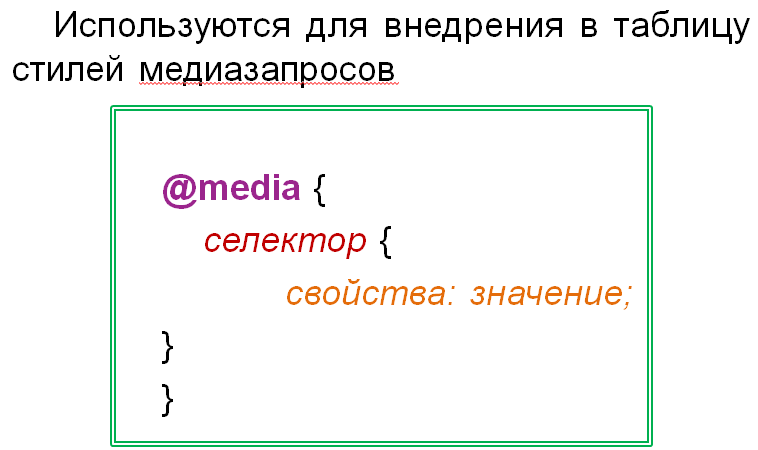
Медиазапросы  — правила, которые позволяют назначать стили в зависимости от размера окна браузера.

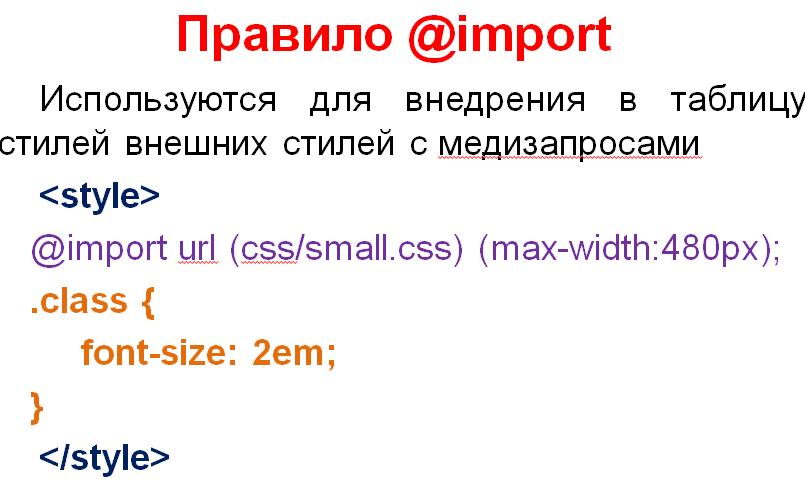
Они подключаются следующим образом:

1. Атрибут media тега link

1. Правило @media

2. Правило @import





43. Основы Flexbox-верстки.

44. Основы Sass  
45. Основы Sass. Наследование свойств  
46. Установка Sass  
47. Примеси Sass  
48. Переменные Sass  
49. Операторы Sass  
50. Основы Sass. Вложенные правила  
51. Основы XML  
52. Правила документов XML.  
53. Элементы и атрибуты XML  
54. Стандарты XML  
55. Cхема DTD  
56. XML Schema  
57. Сущности XML  
58. Язык преобразований XSLT  
59. Сортировка данных в XML  
60. Язык переадресации XPath  
61. Язык запросов XQuery  
62. Введение в XLinks. Применение XPointer  
63. Основы SVG  
64. Атрибуты SVG  
65. Анимация SVG  
66. Основные элементы SVG  
67. Работа с текстом SVG  
68. Обзор и версии языков разметки веб-страниц