1. Этапы создания Web-приложений. (28 билет)

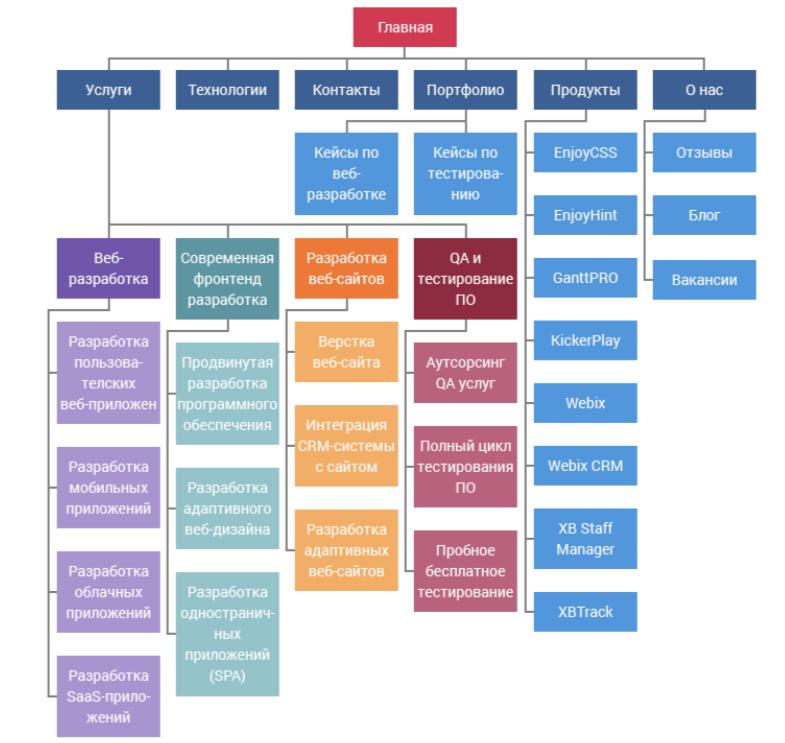
**1. Анализ целевой аудитории**

* **Целевая группа** (целевая аудитория) — люди, объединённые общими признаками, или объединённые ради общей цели или задачи.
* Определение целевого рынка ( б2б или б2с ( бизнес для потребителя и бизнес для бизнеса)
* Подбор инструментов анализа.
* Определение каналов коммуникации (инструменты онлайн- маркетинга, серии мероприятий ...)
* **Сегментирование целевой аудитории** – это разделение аудитории на группы, где они объединены по признаку схожих потребностей (запросов).

**2. Формирование технического задания**

Описывает:

* цели создания сайта и его целевую аудиторию;
* структуру веб-сайта и количество страниц в каждом разделе;
* работу динамических модулей;
* пожелания по дизайну (цвета, использоание фирменного стиля, соотношение графика/текст и т.д.
* используемые технологии (HTML5, CSS3, PHP, Python и проч.);
* порядок предоставления, обработки или создания графической и текстовой информации;
* технические требования к сайту.

 3.**Планирование. Карта сайта**

4. **Проектирование сайта**

* **Проектирование** — это один из первых этапов разработки сайта, на котором разрабатывается архитектура будущего сайта, с учетом специфики бизнеса и целей веб-проекта.
* На этапе проектирования создается прототип сайта.
* ПО для разработки прототипов: Mockingbird, Axure, Moqups, Figma

5. **Юзабилити сайта**

* **Юзабилити** – это качественный показатель простоты и удобства использования сайта.

*Проблемы*:

* сложная или непонятная структура;
* отсутствие нормальной навигации;
* неуместное использование рекламы;
* неудачное расположение или дизайн элементов; слишком медленная загрузка.

6. **Верстка сайта**

* **Верстка сайта** (веб-страниц)– это создание структуры гипертекстового документа на основе HTML-разметки и при использовании каскадных таблиц стилей CSS и клиентских сценариев, таким образом, чтобы элементы дизайна выглядели аналогично макету, созданному при проектировании сайта.
* Технологии: HTML5, CSS3, JavaScript

7. **Программирование**

* **Веб-программирование** – раздел программирования, ориентированный на разработку веб-приложений.
* Языки веб-программирования условно делятся на клиентские и серверные.
* **Фронтенд** (frontend)—это разработка интерфейса, с которым взаимодействуют пользователи (HTML, CSS и JavaScript)
* **Бэкенд** (backend)— все, что происходит на сервере, благодаря нему формируются API, использующиеся для функционирования фронтенда (PHP, Pytho, Node.js)

8. **Наполнение сайта контентом**

9**. Размещение сайта в сети**

10. **Продвижение сайта**

2. Валидность. Проверка на валидность.

***1.Проверка данных формы.***

* Проверка данных формы позволяет удостовериться в том, что пользователи заполняют форму в правильном формате, убедиться, что отправленные данные будут успешно работать с приложениями.
* **Валидация формы** — проверка данных, вводимых пользователем, на корректность.
* *Получать нужные данные в нужном формате* —приложения не будут работать должным образом, если данные пользователя хранятся в неправильном формате, если они вводят неправильную информацию или вообще не вводят информацию.
* *Защитить учетные записи* *пользователей* — заставляя пользователей вводить защищенные пароли, упрощается защита информации об их учетной записи.
* Безопасность — существует множество способов, которыми злоумышленники могут злоупотреблять незащищенными формами, чтобы повредить приложение, в которое они входят.

***2. Различные типы валидации формы***

1. **Проверка на стороне клиента**- это проверка, которая происходит в браузере, прежде чем данные будут отправлены на сервер:

* ***Javascript*** проверка выполняется с использованием JavaScript, полностью настраиваемая.
* ***Встроенная проверка формы,*** используя функции проверки формы HTML5. Для этого обычно не требуется JavaScript. Встроенная проверка формы имеет лучшую производительность, но она не такая настраиваемая, как с использованием JavaScript.

2. **Проверка на стороне сервера** - это проверка, которая возникает на сервере после отправки данных.

* • Серверный код используется для проверки данных перед их сохранением в базе данных.
* • Если данные не проходят проверку валидности, ответ отправляется обратно клиенту, чтобы сообщить пользователю, какие исправления должны быть сделаны.
* • Проверка на стороне сервера не выдает ошибок до тех пор, пока не будет отправлена вся форма.
* •Проверка на стороне сервера - это последний этап защиты приложения от неправильных или вредоносных данных.

3. HTML5. Спецификации.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

*Что появилось нового:*

* Упрощение (доктайп, кодировка)
* Новые теги
* Семантические теги
* Рисование в реальном времени на прямоугольном холсте <canvas> с помощью JavaScript
* Расширенные возможности проигрывания видео — <video> и аудио — <audio>
* Локальное хранилище — стало доступным хранить большие объемы информации прямо в браузере пользователя
* Многопоточность
* Геолокация — можно узнать местонахождение пользователя
* Управление историей браузера
* Drag-and-drop (например, для загрузки файла перетаскиванием его в браузер)
* Редактирование
* Поддержка MathML (математическая разметка для формул) и SVG (форма изображений, масштабируемая векторная графика)
* Работа с формами

4. HTML5. Структура документа.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

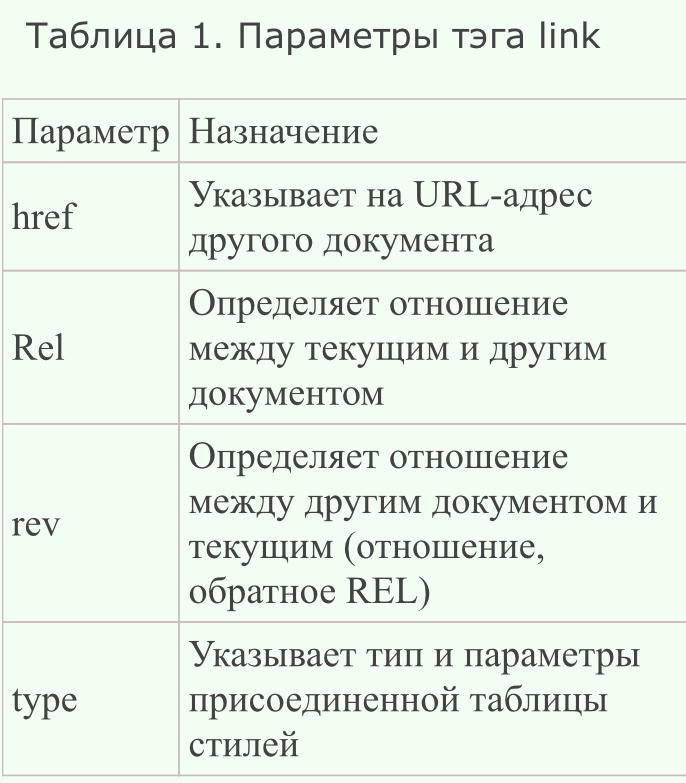
Основные составляющие документа:

1. ***Заголовок*** (колонтитул). Обычно это большая полоса вверху страницы, с крупным заголовком и / или логотипом. Здесь указывается общая информация о веб-сайте, не меняющаяся от страницы к странице.
2. ***Навигационное меню***. Ссылки на основные разделы сайта; обычно в виде кнопок, ссылок или вкладок. Также как и заголовок, навигация *остаётся неизменной на всех страницах сайта*.
3. ***Основное содержимое.*** Большая область в центре страницы, содержащая, в основном, уникальный контент данной веб-страницы.
4. ***Боковая панель***. Как правило, содержит некоторую второстепенную информацию, ссылки, цитаты, рекламу и т.д.
5. ***Нижний колонтитул*** (футер). Полоса в нижней части страницы, которая обычно содержит уведомления об авторских правах или контактную информацию.

5. HTML5. Связь с другими документами.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

Ссылки могут быть как *абсолютные*, так и *относительные*. Оба вида связей могут нарушиться при переносе документа с одного компьютера на другой.

Для решения этой проблемы используются тэги <**base**> и <**link**>, которые включаются в заголовок для того, чтобы связь между документами не нарушалась.

1. Тэг <**base**> (ОТНОСИТЕЛЬНАЯ ССЫЛКА) служит для указания полного базового URL-адреса документа. С его помощью относительная ссылка продолжает работать, если документ переносится в другой каталог или даже на другой компьютер. (*Тэг <base> имеет один обязательный параметр href, после которого указывается полный URL-адрес документа.*)
2. Тэг <**link**> указывает на связь документа, содержащего данный тэг и другого документа или объекта. Он состоит из URL-адреса и параметров, конкретизирующих отношения документов. Заголовок документа может содержать любое количество тэгов <link>.
3. Абсолютные ссылки могут быть громоздкими и переставать работать, если перемещен младший по иерархии документ.
4. Относительные ссылки легче вводить и обновлять, но и эта связь обрывается, если перемещен старший по иерархии документ.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

*Ряд элементов html* ***предназначены для форматирования*** *текстового содержимого, например, для выделения жирным или курсивом и т.д.*

6. HTML5. Форматирование текста.

**Физическое форматирование html-документа** – это процесс форматирования html-кода при помощи набора соответствующих элементов разметки, которые используются в основном для определения внешнего вида своего содержимого при отображении его браузером.

1. <b> (от англ. bold – жирный) – элемент разметки «b», сформированный данным тегом, отображает свое содержимое полужирным шрифтом
2. <i> (от англ. italic – курсивный) – элемент разметки «i», сформированный данным тегом, отображает свое содержимое курсивом;
3. <u> (от англ. unarticulated – невнятный) – содержимое элемента разметки «u», сформированного данным тегом, отображается подчеркнутым;
4. <s> (от англ. strike out – зачеркнутый) – содержимое элемента разметки «s», сформированного данным тегом, отображается зачеркнутым;
5. <sub> (от англ. subscript – нижний индекс) – элемент разметки «sub», сформированный данным тегом, отображает свое содержимое в виде нижнего индекса;

.

*Все перечисленные элементы формируются соответствующими парными тегами и отображаются браузерами как строчные элементы.*

**Логическое форматирование html-документа** – это процесс форматирования html-кода при помощи набора соответствующих элементов разметки, которые предназначены главным образом для структурной, логической разметки документа, определяя степень важности своего содержимого и его отношение к тому или иному типу данных, а также выделяя смысловое отличие своего содержимого от окружающего контекста.

1. <em> (от англ. emphasis – акцент) – элемент разметки «em», сформированный данным тегом, предназначен для акцентирования внимания на фрагменте текста (обращает внимание на его важность) и отображает свое содержимое курсивом;
2. <strong> (от англ. strong – сильный) – элемент разметки «strong», сформированный данным тегом, предназначен для еще большего акцентирования текста (делает его еще более важным) и отображает свое содержимое полужирным шрифтом;
3. <cite> (от англ. cite – цитировать) – элемент разметки «cite», сформированный данным тегом, предназначен для выделения сносок на другой материал и отображает свое содержимое курсивом;
4. <code> (от англ. code – код) – элемент разметки «code», сформированный данным тегом, предназначен для выделения текста программного кода и отображает свое содержимое моноширинным шрифтом;
5. <kbd> (от англ. keyboard – клавиатура) – элемент разметки «kbd», сформированный данным тегом, предназначен для выделения текста, который должен быть введен с клавиатуры или используется для названия клавиш клавиатуры; элемент отображает свое содержимое моноширинным шрифтом;
6. <var> (от англ. variable – переменная) – элемент разметки «var», сформированный данным тегом, предназначен для выделения переменных компьютерных программ и отображает свое содержимое курсивом;
7. <q> (от англ. quote – цитата, кавычки) – элемент разметки «q», сформированный данным тегом, предназначен для выделения в тексте небольших цитат и отображает свое содержимое в кавычках;
8. <ins> (от англ. inserted – вставленный) – элемент разметки «ins», сформированный данным тегом, предназначен для выделения текста, который был добавлен в новую версию документа, и отображает свое содержимое подчеркнутым;
9. <del> (от англ. deleted – удаленный) – элемент разметки «del», сформированный данным тегом, предназначен для выделения текста, который был удален в новой версии документа, и отображает свое содержимое зачеркнутым;
10. <small> (от англ. small – маленький) – элемент разметки «small», сформированный данным тегом, предназначен для выделения текста, который можно отнести к надписям мелким шрифтом или пометкам, как, например, второстепенная информация в конце юридических документов об отказе от ответственности или же информации о лицензии; элемент отображает свое содержимое моноширинным шрифтом уменьшенного размера;

*Все перечисленные элементы также формируются соответствующими парными тегами и отображаются браузерами как строчные элементы.*

7. HTML5. Поддержка мультимедиа

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

**ВИДЕО**

Особую популярность это приобрело тогда, когда на формат HTML5 перешел крупнейший видеохостинг YouTube.com.

1. Для успешного интегрирования видео в страницу необходимо воспользоваться тегом ‹video›.
2. К сожалению между производителями браузеров нет единого мнения в каком формате должно быть представлено видео, поэтому каждый из них продвигает свой формат.
3. Если IE и Safari поддерживают видео в формате H.264 (которое поддерживалось Flash-плеерами), то Opera и Firefox продвигают opensource форматы Vorbis и Theora.
4. Chrome же может правильно отображать видео во всех форматах, в том числе WebM.
5. Не все браузеры могут отображать HTML5 видео, поэтому под тегом ‹source› можно указать ссылку для скачивания видео, либо интегрировать Flash плеер.
6. Атрибут preloadпозволяет браузеру автоматически начинать загрузку видео, что может быть полезно, если размер видео достаточно большой. Атрибут controlsпозволяет установить стандартные элементы управления видеопотоком.

**АУДИО**

1. Для внедрения на страницу звукового файла необходимо воспользоваться тегом ‹audio› с необходимыми атрибутами.
2. • Пример ниже выводит на странице блок управления аудиофайлом, содержащим ссылку на скачивание этого файла.
3. • Звук будет воспроизводиться автоматически.
4. У данного тега есть свои особенности поддержки в браузерах. Например, браузер MozillaFirefox работает с .ogg файлами, тогда как Webkit-браузеры работают с .mp3

8. HTML5. Форматирование HTML-документа. Разделение на абзацы. Перевод строки. Включение комментариев в документ.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

Применение специального тэга разбиения на **абзацы** <Р>. Перед началом каждого абзаца текста следует поместить тэг <Р>. Закрывающий тэг </Р> не обязателен. Браузеры обычно отделяют абзацы друг от друга пустой строкой.

Браузеры обычно интерпретируют несколько стоящих подряд тэгов абзаца <Р> как один. То же самое относится и к тэгу перевода **строки** <BR>. Поэтому соз­дать несколько пустых строк при помощи этих тэгов не удается.

Включение тэга <**BR**> в текст документа обеспечит раз­мещение последующего текста с начала новой строки. Например, такой подход может использоваться для создания структур типа списков без использования специальных тэгов разметки списка.

**Включение комментариев в документ**

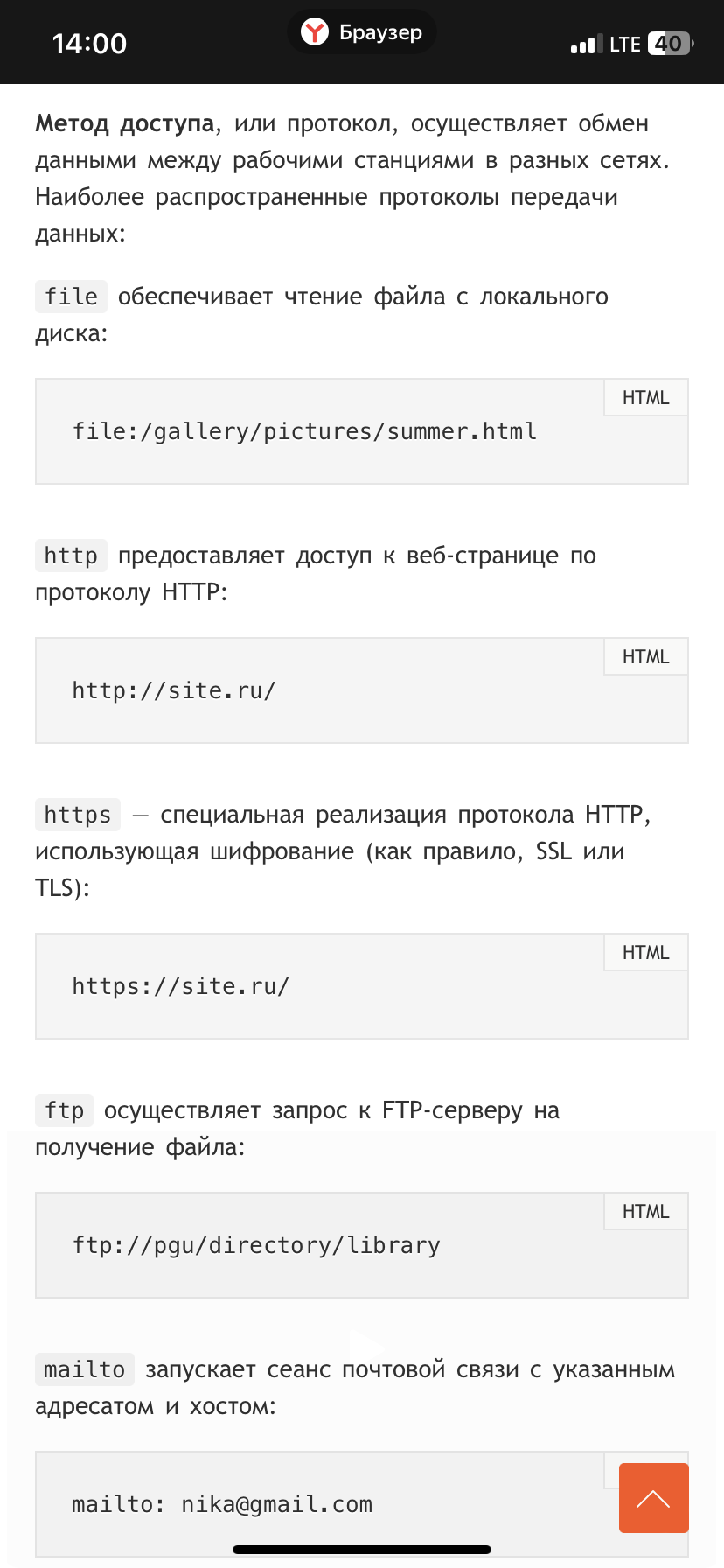
В HTML-документ можно включать комментарии, которые не будут видны читателю. Они могут состоять из произвольного числа строк и должны на­чинаться тэгом <!-- и заканчиваться тэгом -- >. Все, что заключено внутри этих тэгов, при просмотре страницы не будет отображаться на экране.(**зажать cntrl + /**)

9. HTML5. Форматирование HTML-документа. Ссылки на другие документы и файлы.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

HTML-ссылки создаются с помощью элементов <a>, <area> и <link>. Ссылки представляют собой связь между двумя ресурсами, одним из которых является текущий документ.

Ссылки можно поделить на две категории:



1. ссылки на **внешние** ресурсы — создаются с помощью элемента <link> и используются для расширения возможностей текущего документа при обработке браузером;
2. **гиперссылки** — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Ссылка состоит из двух частей — **указателя** и **адресной части**. Указатель ссылки представляет собой фрагмент текста или изображение, видимые для пользователя. Адресная часть ссылки пользователю не видна, она представляет собой адрес ресурса, к которому необходимо перейти.

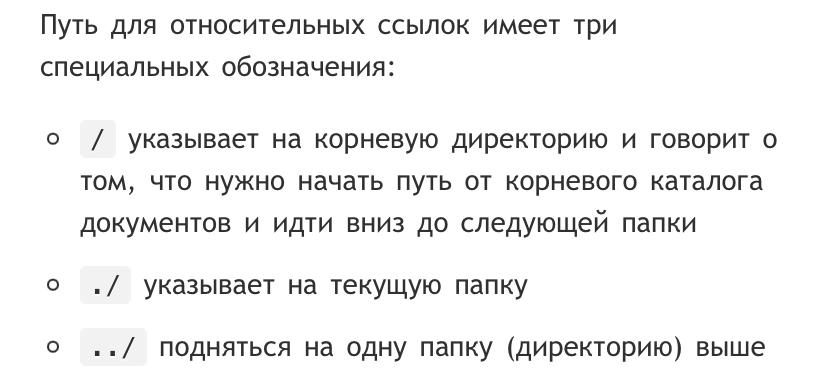
Когда в ссылке указывается только имя файла, браузер предполагает, что файл находится в той же папке, что и документ, содержащий гиперссылку.

Чтобы создать ссылку на файл, находящийся вне папки, содержащей текущий документ, необходимо указать расположение файла или путь. HTML поддерживает два вида пути: абсолютный и относительный.

1. ***Абсолютный*** путь указывает точное местоположение файла в пределах всей структуры папок на компьютере (сервере). Абсолютный путь к файлу даёт доступ к файлу со сторонних ресурсов и содержит следующие компоненты:

1. протокол, например, http (опционально);
2. домен (доменное имя или IP-адрес компьютера);
3. папка (имя папки, указывающей путь к файлу);
4. файл (имя файла).

2.***Относительный*** путь описывает путь к указанному документу относительно текущего. Путь определяется с учётом местоположения веб-страницы, на которой находится ссылка. Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте. Когда браузер не находит в ссылке протокол http://, он выполняет поиск указанного документа на том же сервере.

Относительный путь содержит следующие компоненты:

1. папка (имя папки, указывающей путь к файлу);
2. файл (имя файла).

Главное отличие относительного пути от абсолютного в том, что относительный путь не содержит имени корневой папки и родительских папок, что делает адрес короче, и в случае переезда с одного домена на другой не нужно прописывать новый абсолютный адрес.

10. HTM5L. Форматирование HTML-документа. Списки.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

HTML-списки используются для группировки связанных между собой фрагментов информации.

*Существует три вида списков:*

1. **маркированный** список — <ul> — каждый элемент списка <li> отмечается маркером,
2. **нумерованный** список — <ol> — каждый элемент списка <li> отмечается цифрой,
3. список **определений** — <dl> — состоит из пар термин <dt> — <dd> определение.

Для элемента <li> доступен атрибут **value**, который позволяет изменить номер по умолчанию для выбранного элемента списка.

Списки **определений** создаются с помощью элемента <dl>. Для добавления термина применяется элемент <dt>, а для вставки определения — элемент <dd>.

Для элементов <dl>, <dt> и <dd> доступны *‎глобальные атрибуты.*

11. HTML5. Геолокация. Управление историей браузера. Drag-and-drop.

**HTML5** (version 5) — язык для структурирования и представления содержимого всемирной паутины.

**ГЕОЛАКАЦИЯ**

**Географическое местоположение** в HTML5 используется для того, чтобы поделиться местоположением с некоторыми веб-сайтами и быть в курсе точного местоположения. В основном она используется для местных предприятий и ресторанов или для отображения местоположений на карте.

Возможность геолокации предельно проста. Она состоит из трех методов объекта **navigator.geolocation**: getCurrentPosition(), watchPosition() и clearWatch().

**УПРАВЛЕНИЕ ИСТОРИЕЙ**

**История просмотров** — это функциональность HTML5, которая расширяет возможности объекта history JavaScript.

Обьект **history** имеет только одно свойство и три основных метода. Это свойство **length** — содержит информацию о количестве элементов в списке истории просмотров (т.е. в поддерживаемом браузером списке недавно посещенных веб-страниц).

Наиболее полезным методом объекта history является метод **back()**. Этот метод позволяет переместить пользователя на один шаг назад в истории просмотров

Эффект этого метода равнозначен нажатию пользователем кнопки браузера "Назад". Подобным образом можно использовать метод **forward**() для перемещения на один шаг вперед или метод **go**() для перехода вперед или назад на определенное количество шагов.

**События Drag&Drop**

Ниже перечислены события Drag&Drop, с помощью которых можно контролировать процесс перетаскивания:

1. dragStart: пользователь начинает перетаскивание элемента.
2. dragEnter: перетаскиваемый элемент достигает конечного элемента.
3. dragOver: курсор мыши наведен на элемент при перетаскивании.
4. dragLeave: курсор мыши покидает пределы перетаскиваемого элемента.
5. drag: курсор двигается при перетаскивании.
6. drop: происходит drop элемента.
7. dragEnd: пользователь отпускает курсор мыши в процессе перетаскивания.

12. Валидация полей формы средствами HTML5.Регулярные выражения.

**ВАЛИДАЦИЯ ПОЛЕЙ**

Проверка данных формы позволяет удостовериться в том, что пользователи заполняют форму в правильном формате, убедиться, что отправленные данные будут успешно работать с приложениями.

* **Валидация формы** — проверка данных, вводимых пользователем, на корректность.
* *Получать нужные данные в нужном формате* —приложения не будут работать должным образом, если данные пользователя хранятся в неправильном формате, если они вводят неправильную информацию или вообще не вводят информацию.
* *Защитить учетные записи* *пользователей* — заставляя пользователей вводить защищенные пароли, упрощается защита информации об их учетной записи.
* Безопасность — существует множество способов, которыми злоумышленники могут злоупотреблять незащищенными формами, чтобы повредить приложение, в которое они входят.

*Самый простой, но не самый эффективный способ - это валидация полей формы через стандартные свойства полей HTML*:

1. Поле обязательно к заполнению (required)
2. Проверка с регулярными выражениями (aтрибут pattern, который ожидает Regular Expression в качестве значения). Регулярное выражение (regex) - это шаблон который используется для соответствия комбинаций символов в текстовых строках. Например, [0- 9] — соответствует любой цифре в диапазоне от 0 до 9
3. Ограничение длины записей. Все текстовые поля, созданные с помощью элементов (<input> или <textarea>) могут быть ограничены по размеру, используя атрибуты minlength и maxlength. Поле не валидно если его значение короче чем minlength или значение длиннее значения maxlength.
4. Для числовых полей (например <input type="number">), атрибуты min и max также обеспечивают ограничение валидации. Если значение поля меньше атрибута min или больше атрибута max, поле будет невалидным.

13. CSS-каскадные таблицы стилей.Верстка. Добавление стилей в документ.

**КАСКАДНЫЕ ТАБЛИЦЫ**

1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах.

**ДОБАВЛЕНИЕ СТИЛЕЙ**

**Технология - CSS** (каскадные таблицы стилей) позволяет хранить цвет, размеры текста и другие параметры в стилях.

**Стилем** называется набор правил форматирования, который применяется к элементу документа, чтобы быстро изменить его внешний вид.

**Подключение стилей**

Правило подключения глобальных и внешних стилей состоит из **селектора** и **объявлений** стиля.

Селектор, расположенный в левой части правила, определяет элемент (элементы), для которых установлено правило. Далее, в фигурных скобках перечисляются объявления стиля, разделенные точкой с запятой.

**4 способа добавление стилей в документ:**

1. Вложение Стилей в HTML-документ.
2. Ссылка из HTML-документа на внешний документ Стилей
3. Импортирование документа Стилей в HTML-документ.
4. Добавление Стилей непосредственно в строки HTML-документа.

1. **Внешние таблицы стилей представляют собой текстовый файл** с расширением .css. Внешняя таблица стилей подключается к веб-странице с помощью тега <**link**>, расположенного внутри раздела <head></head>

Файл **mystyles.css** будет содержать:

H1 { color: green; font-family: impact }

P { background: yellow; font-family: courier }

**2. Внутренние стили** встраиваются в раздел <head></head> HTML- документа и определяются внутри тега <style></style>

*Внутренние стили имеют приоритет над внешними, но уступают в приоритете встроенным стилям.*

**3. Встроенные стили** используются непосредственно внутри тега элемента с помощью атрибута style

**4. Импорт CSS.** В данном способе свойства стилей подключены

к веб-документу посредством директивы @import, располагающейся в этом документе между тегами <style> и </style>

14. CSS-каскадные таблицы стилей .Верстка. Селекторы. ( 9 экзаменационный билет)

**КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ**

1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы. Этот метод мало отличается от традиционного HTML.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах

**СЕЛЕКТОРЫ**

**Селектор**—это конструкция, позволяющая выбрать элемент, к которому будут применены данные стили.

Стиль CSS состоит из селектора, который всегда располагается слева, и блока объявления стилей, который заключается в фигурные скобки и следует непосредственно за селектором.

**Виды селекторов**:

1. Универсальный селектор (\*) соответствует любому HTML-элементу.
2. Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта.
3. Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта
4. Селектор идентификатора позволяет форматировать один конкретный элемент. Идентификатор (id) должен быть уникальным и на одной странице может встречаться только один раз.

15. CSS-каскадные таблицы стилей .Верстка. Псевдокласс.

**КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ**

1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы. Этот метод мало отличается от традиционного HTML.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах.

**ПСЕВДОКЛАСС**

**Псевдокласс в CSS — это ключевое слово, добавленное к селектору, которое определяет его особое состояние.**

*Псевдокласс позволяет учитывать различные условия или события при определении свойств HTM тэга.*

Можно использовать псевдоклассы для установки разных стилей посещённых и непосещённых ссылок.

Активные ссылки имеют псевдоклассa:active,и a:hover,когда указатель-надссылкой.

**Селектор:Псевдокласс { Описание правил стиля }**

1. **Псевдокласс :link** используется для ссылок на страницы, которые пользователь ещё не посещал.
2. **Псевдокласс :active** используется для активных ссылок.
3. **Псевдокласс :hover** используется для ссылок, над которыми находится указатель мыши.Это можно использовать для создания интересных эффектов.

16. CSS-каскадные таблицы стилей.Верстка. Модель форматирования.

**КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ**

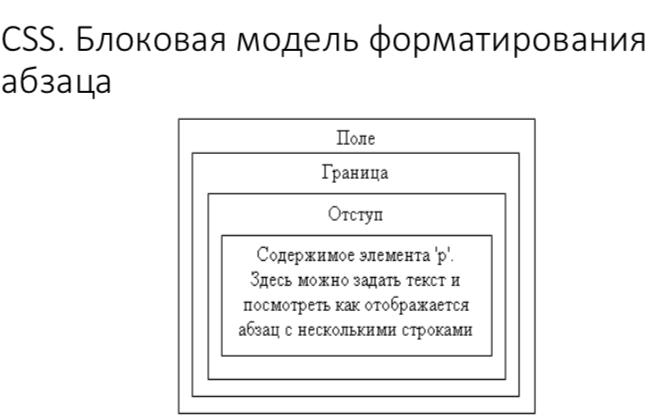
1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы. Этот метод мало отличается от традиционного HTML.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах.

**МОДЕЛЬ ФОРМАТИРОВАНИЯ**

Модель форматирования CSS ориентирована на представление любого элемента html в окружении вложенных прямоугольных блоков.

1. Блок содержимого элемента отделен от границы отступами.
2. Самым внешним блоком является поле.
3. Свойства таблицы стилей позволяют устанавливать размеры и цвета всех блоков, составляющих в сумме отображаемый элемент.
4. "Поле" всегда прозрачно, поэтому его цвет наследует цвет родителя элемента (для <p> - это <body> ).
5. Отступ всегда имеет цвет фона самого элемента.
6. Все перечисленные блоки в совокупности составляют блок форматирования или отображения элемента, т.е. видимое в окне браузера изображение элемента.
7. Размеры блока форматирования элемента складываются из размеров самого элемента и размеров отступов границы и полей.

17. CSS-каскадные таблицы стилей. Верстка. Позиционирование в CSS. Box-модель в CSS и типы позиционирования.

**КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ**

1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы. Этот метод мало отличается от традиционного HTML.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах.

**ПОЗИЦИОНИРОВАНИЕ В CSS**

CSS рассматривает макет html-документа как **дерево** элементов. Уникальный элемент, у которого нет родительского элемента, называется **корневым** элементом.

*(Модуль CSS-позиционирование описывает, как любой из элементов может быть размещен независимо от порядка документа)*

В CSS2 каждый элемент в дереве документа генерирует ноль или более блоков в соответствии с блочной моделью. Модуль CSS3 дополняет и расширяет схему позиционирования. Расположение этих блоков регулируется:

1. размерами и типом элемента,
2. схемой позиционирования (нормальный поток, обтекание и абсолютное позиционирование),
3. отношениями между элементами в дереве документа,
4. внешней информацией (например, размер области просмотра, внутренними размерами изображений и т.д.).

**Схемы позиционирования**

1. **Нормальный поток.** Нормальный поток включает блочный контекст форматирования (элементы с display block, list-item или table), строчный (встроенный) контекст форматирования (элементы с display inline, inline-block или inline-table), и относительное и «липкое» позиционирование элементов уровня блока и строки.
2. **Обтекание**. В обтекающей модели блок удаляется из нормального потока и позиционируется влево или вправо. Содержимое обтекает правую сторону элемента с float: left и левую сторону элемента с float: right.
3. **Абсолютное позиционирование.** В модели абсолютного позиционирования блок полностью удаляется из нормального потока и ему присваивается позиция относительно содержащего блока. Абсолютное позиционирование реализуется с помощью значений position: absolute; и position: fixed;.

**Box-модель в CSS и типы позиционирования.**

Каждый элемент в CSS заключён в блок, и понимание поведения этих блоков — это ключ к умению задавать раскладку с помощью CSS, то есть выстраивать одни элементы относительно других элементов.

Если элемент определён как блочный, то он будет вести себя следующим образом:

1. Начнётся с новой строки.
2. Будет расширяться вдоль строки таким образом, чтобы заполнить всё пространство, доступное в её контейнере. В большинстве случаев это означает, что блок станет такой же ширины, как и его контейнер, заполняя 100% доступного пространства.
3. Будут применяться свойства width и height.
4. Внешние и внутренние отступы, рамка будут отодвигать от него другие элементы.
5. Если не изменить намеренно тип отображения на строчный, то такие элементы, как заголовки (например, <h1>) и <p>, все используют block как свой внешний тип отображения по умолчанию.

**ПОЗИЦИОНИРОВАНИЕ**

Можно выделить четыре типа позиционирования: нормальное, абсолютное, относительное и фиксированное. В зависимости от выбранного типа, установленного через свойство position, будет изменяться и система координат.

18. CSS-каскадные таблицы стилей.Верстка. Цвет и фон.

**КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ**

1. **Внутренние стили**. Внутренние стили определяются атрибутом style конкретных тегов. Внутренний стиль действует только на определенные такими тегами элементы. Этот метод мало отличается от традиционного HTML.
2. **Глобальные стили.** Глобальные стили CSS располагаются в контейнере <style>...</style>, расположенном в свою очередь в контейнере <head>...</head>.Глобальные стили являются универсальным средством, позволяющим не только оперативно изменять внешний вид Web-страницы, но и и бороться с перегруженностью документа оформительскими тегами.
3. **Внешние (связанные) стили**. Внешние (связанные) стили определяются в отдельном файле с расширением css. Внешние стили позволяют всем страницам сайта выглядеть единообразно. Для связи с файлом, в котором описаны стили, используется тег <link>, расположенный в контейнере <head>...</head>. В этом теге должны быть заданы два атрибута: rel="stylesheet" и href, определяющиЙ адрес файла стилей.

**ВЕРСТКА**

**Виды верстки сайтов**

1. **Фиксированная** верстка или **статическая**. Вне зависимости от размеров окна браузера или устройства ширина страниц будет постоянной. Все элементы занимают строго определенную ширину в пикселях на странице.
2. **Резиновая верстка**. Страница занимает всю ширину браузера, каких бы размеров она не была и на каком бы устройстве вы не открыли эту страницу. Ширина элементов страницы задается в процентах от ширины окна, и поэтому занимают всю доступную область.
3. **Табличная верстка** или верстка таблицами. Сетка страницы строится с помощью таблиц.
4. **Блочная верстка.** Сетка страниц конструируется из множества блоков <div> и <span>, которые вложены друг в друга. Для доступа к ним используются атрибуты id и class.
5. **Адаптивная верстка**. Страницы хорошо адаптируются под любое разрешение экрана пользователя. Это достигается тем, что используют несколько таблиц стилей под разные разрешения.
6. **Гибкая верстка или flex верстка**. Вначале применяется известная всем блочная верстка, а потом нужные блоки превращают во флекс-контейнеры (флексбоксы). В стилях элемента указывают display: flex; Элементу можно указать направление главной оси и выравнивание.
7. **Семантическая верстка**. Она явилась логичным продолжением блочной верстки и стала доступна в HTML5. Новые теги делают станицу более структурированной и высоко поднимаются при поисковых запросах.
8. **Валидная верстка.** Это верстка, выполненная в соответствии со стандартами W3C. Проверить свою HTML- страницу на корректность можно с помощью специального валидатора W3C.
9. **Кроссбраузерная верстка**. Страницы выглядят одинаково в разных браузерах.

**ЦВЕТ И ФОН**

**Цвет в стилях можно задавать тремя способами:**

1. 1. По шестнадцатеричному значению: обозначение цвета разбивается на три составляющие #rrggbb.Допускается использовать сокращенную форму вида #rgb: #fe0 эквивалентна #ffee00
2. По названию: red, black, gray и т.д.
3. 3. С помощью RGB: можно определить цвет, используя значения красной, зеленой и синей составляющей в десятичном исчислении:

Помимо обычных цветовых значений свойство **background-**

**color** может принимать (установленное по умолчанию).

также значение **transparent**

В этом случае **фон** текста элемента цвета собственного не имеет, взамен используется подложка элемента уровнем выше (если он есть).

19. CSS анимация.

Что такое анимация CSS?

• Анимация позволяет элементу постепенно переходить от одного стиля к другому.

• Вы можете изменить любое количество свойств CSS, сколько угодно раз.

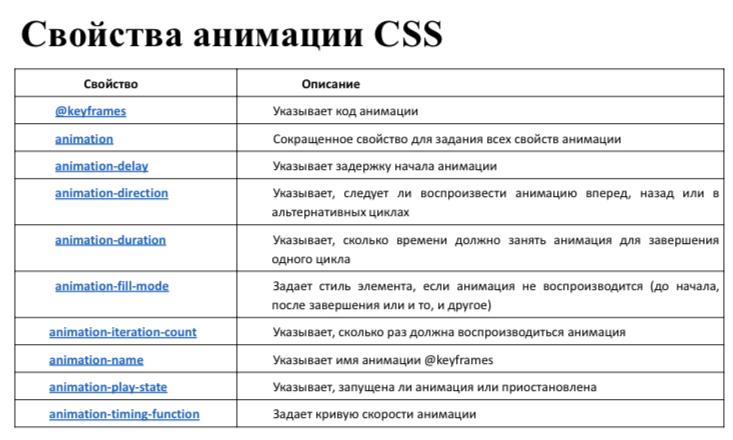
• Чтобы использовать анимацию CSS, необходимо сначала указать некоторые ключевые кадры для анимации

Преобразования CSS позволяют переводить, вращать, масштабировать и наклонять элементы.

• **Преобразование** — это эффект, который позволяет элементу изменять форму, размер и положение.

• CSS поддерживает 2D и 3D преобразования.

1. Метод **rotate**(). Метод поворота - вращает элемент по часовой стрелке или против часовой стрелки в зависимости от заданного градуса.
2. Метод **Scale** () Метод scale - увеличивает или уменьшает размер элемента (в соответствии с параметрами, заданными для ширины и высоты).
3. Метод **skewX**()- skew(), skewX(), skewY() — угол (deg, grad, rad), наклоняет элемент вдоль оси X заданным углом.
4. Метод **Translate** ()Метод translate- перемещает элемент из текущей позиции (в соответствии с параметрами, заданными для оси X и оси Y).

**СВОЙВСТВА**

1. **Задержка анимации.** Свойство animation-delay указывает задержку начала анимации.
2. **Повтор анимации.** Свойство animation-iteration-count указывает, сколько раз должна выполняться анимация.
3. **Запуск анимации в обратном направлении или альтернативные циклы.**

20. SVG графика

**Что таное и зачем нужно SVG**

**SVG** расшифровывается как «масштабируемая векторная графика». Это значит, перед нами векторное изображение, которое можно масштабировать без потери

качества.

**SVG** - это не единственный формат векторной графики, есть много других. Но именно SVG используют на веб-страницах и в HTML-коде. Его инструкции оптимизированы для веб- разработки, и отрисовка происходит прямо в браузере. И, самое главное, все команды SVG представлены в виде текста и координат, а не в виде отдельных файлов. Это значит, что SVG- изображение можно зашить прямо в код веб-страницы, что иногда бывает незаменимо.

SVG используется в вебе для отрисовки:

1. • геометрических фигур и линий,
2. • ЛОГоТиПоВ,
3. • элементов интерфейса на странице,
4. • простых картинок и иллюстраций.

21. Верстка Flex

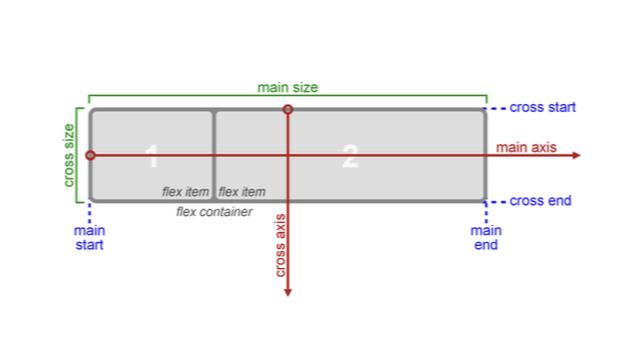
• **Flexbox**— модуль макета гибкого контейнера — представляет собой способ компоновки элементов.

• Flexbox состоит из flex-контейнера — родительского контейнера и flex-элементов — дочерних блоков. Дочерние элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

**В основе flexbox лежит идея оси.**

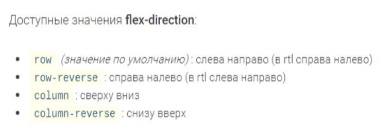
• Flexbox является инструментом двумерной компоновки и использует для работы две оси — горизонтальную (главную ось) и поперечную.

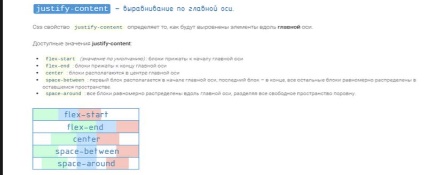
**Модуль** flexbox позволяет решать **следующие задачи**:

1. • Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх;
2. • Переопределять порядок отображения элементов;
3. • Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство;
4. • Решать проблему с горизонтальным и вертикальным центрированием;
5. • Переносить элементы внутри контейнера, не допуская их переполнения;
6. • Создавать колонки одинаковой высоты;
7. • Создавать прижатый к низу страницы подвал сайта.

**СВОЙСТВА**

1.Выравнивание





2.Порядок отображения элементов ORDER

1. Свойство определяет порядок, в котором flex- элементы отображаются внутри flex-контейнера.
2. По умолчанию для всех flex-элементов задан порядок order: 0; и они следуют друг за другом по мере добавления во flex-контейнер.
3. Самый первый flex-элемент по умолчанию расположен слева.
4. Чтобы поставить любой flex-элемент в начало строки, ему нужно назначить order: -1;, в конец строки — order: 1;.
5. Свойство не наследуется.

3.Размеры

4. Выравнивание

5.Расположение

6.Порядок

22. Верстка Grid

**CSS GridLayout** - самая мощная система компоновки из доступных на данный момент в CSS.

1. • Это двумерная система, которая может обрабатывать как колонки, так и строки, в отличии от Flexbox, который в значительной степени является одномерной системой.
2. • При работе с CSS Grid, вы применяете CSS правила и родительским элементам, которые становятся Grid контейнерами) и к дочерним элементам (которые становятся Grid элементами).

CSS GridLayout (aka "Grid") — это двумерная система компоновки, основанная на сетке, цель которой заключается в том, чтобы полностью изменить способ проектирования пользовательских интерфейсов, основанных на сетке.

(С помощью Grid CSS вы можете создавать ряды и столбцы, определяя их размеры, расположение и поведение.)

Grid контейнер - Это элемент, который содержит все элементы сетки. Обычно это может быть элемент <div> или другой блочный элемент.

Grid item - Это элементы, которые размещаются внутри контейнера сетки.

Грид-трек – это промежуток между любыми двумя линиями грида. (Размер треков может быть задан с помощью любой единицы длины. Спецификация также вводит дополнительную единицу длины, позволяющую создавать гибкие (flexible) грид-треки. Новая единица длины «fr» представляет собой долю (fraction) доступного пространства в грид-контейнере)

Явный и неявный грид - при создании грида в примере выше мы специально объявляли треки-колонки при помощи свойства grid-template-columns, но грид также самостоятельно создавал строки. Эти строки - часть «неявного» грида. В отличие от него, «явный» грид состоит из строк и колонок, заданных с помощью grid-template-columns или grid-template-rows.

23. JavaScript. Обзор языка. Область применения.

**JavaScript** - объектно-ориентированный язык сценариев, позволяющий включать в веб-страницы исполняемое содержимое (скрипты).

• JavaScript выполняется на стороне клиента.

• С помощью JavaScript можно создавать статические HTML страницы и страницы с динамически изменяемым содержанием.

**ОБЗОР**

1. Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.
2. • Возможности JavaScript сильно зависят от окружения, в котором он работает.
3. • В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером

• Например, в браузере JavaScript может:

1. – Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
2. – Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
3. – Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
4. – Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
5. – Запоминать данные на стороне клиента

**ОБЛАСТЬ ПРИМЕНЕНИЯ**

**Основные области использования** JavaScript при создании интерактивных HTML- страниц:

1. • Динамического создания содержимого страницы во время ее загрузки или уже после того, как она полностью загружена;
2. • Отображения диалоговых панелей и сообщений в статусной строке
3. браузера;
4. • Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;
5. • Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа (DHTML);

**Особенности JavaScript:**

1. • Полная интеграция с HTML/CSS.
2. • Простая реализация.
3. • Поддерживается всеми основными браузерами и включён по умолчанию.

24. JavaScript. Подключение и выполнение.

**JavaScript** - объектно-ориентированный язык сценариев, позволяющий включать в веб-страницы исполняемое содержимое (скрипты).

• JavaScript выполняется на стороне клиента.

• С помощью JavaScript можно создавать статические HTML страницы и страницы с динамически изменяемым содержанием.

**ПОДКЛЮЧЕНИЕ И ВЫПОЛНЕНИЕ**

1. • Подключение в любом месте
2. • Внешние скрипты

**Чтобы код Javascript сработал в браузере пользователя, его необходимо вставить между тегами <script> и </script>.**

**Функции и события JavaScript**

1. **•** Функция JavaScript — это блок кода, который выполняется по "вызову".

• Например, функция может вызываться при возникновении какого-нибудь события, вроде нажатия пользователем на кнопку мыши.

**JavaScript в теге <head> и <body>**

1. • В HTML документ можно вставлять любое число скриптов.

• На HTML странице скрипты можно размещать внутри

секции <body> или <head>, либо в обоих сразу. (<script src="main.js" defer></script>)

defer – подходит для тех скриптов, которые требуют взаимодействия с DOM или другими скриптами, так как defer гарантирует, что выполнение скрипта не начнётся до полной загрузки страницы, что актуально и в старых версиях браузеров.

25. JavaScript. Основы программирования. Константы. Числа. Строки. Логические

литералы. Переменные. Массивы.

**JavaScript** - объектно-ориентированный язык сценариев, позволяющий включать в веб-страницы исполняемое содержимое (скрипты).

• JavaScript выполняется на стороне клиента.

• С помощью JavaScript можно создавать статические HTML страницы и страницы с динамически изменяемым содержанием.

**КОНСТАНТЫ**

Константы Простейшими выражениями языка JavaScript являются константы (называемые также литералами ) и переменные. B JavaScript- программе можно использовать следующие

типы констант:

1. • числа
2. • строки
3. • логические литералы true и false
4. • литерал null

**СТРОКИ**

• Константы-строки записываются в

апострофах или в кавычках:

• 'Hello world!' "Привет!"

**Строки**

К строкам применимы следующие операции:

1. • сцепление строк
2. • определение длины строки
3. • выделение фрагмента строки или одного символа • поиск заданного фрагмента слева направо или справа налево • замена больших букв на маленькие или маленьких на большие
4. • добавление в строку HTML-тегов
5. • замена специальных символов на их ASCII-коды
6. • преобразование строки в число

**ЧИСЛА**

Числа B языке JavaScript применяются следующие формы

записи чисел:

1. • **десятичные числа**, например:

• 5 137 1.3 6.67e-11

1. **восьмеричные** числа, например: • 01234 -077 0312

*Признаком восьмеричного числа является 0 (ноль в начале записи числа (а для*

*отрицательных чисел 0 после знака минус).*

1. • **шестнадцатеричные** числа, например:

• OxFF -OxCCFF Oxabcdef

*Признаком шестнадцатеричного числа является Ох в начале записи числа (а для отрицательных*

*чисел Ох после знака минус).*

**Логические литералы**

• Логические литералы "истинно" и "ложно"

в JavaScript имеют следующий вид:

**True**//**False**

**Переменные. Типы переменных**

• B отличие от большинства языков программирования, B JavaScript переменные не имеют определенного типа (строка, целое число, логическое значение И т.Д.)

• "Тип" переменной определяется типом значения, которое в данный момент хранится в этой переменной.

**МАССИВЫ**

1.Одномерные массивы

Для создания массивов используется функция-

конструктор Array(). И может иметь следующий вид:

имя массива = new Array0

имя массива = new Array (длина)

имя массива = new Array (элемент,..элемент)

Или можно использовать запись следующего

вида:

var имя массива = [];

имя массива=Ґʼэлемент',..,'элемент'];

2.Многомерные массивы

26. JavaScript. Библиотека jQuery

**JavaScript** - объектно-ориентированный язык сценариев, позволяющий включать в веб-страницы исполняемое содержимое (скрипты).

• JavaScript выполняется на стороне клиента.

• С помощью JavaScript можно создавать статические HTML страницы и страницы с динамически изменяемым содержанием.

**БИБЛИОТЕКА**

**jQuery**—набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML (Википедия).

• Разработка jQuery ведётся командой добровольцев на пожертвования.

**Библиотеки JavaScript используются во множестве проектов**. Основные причины использования библиотек:

1. • cинтаксис библиотек более простой по сравнению с javascript
2. • библиотеки кроссбраузерны, что упрощает разработку сайтов для разных браузеров и не требует тестирование кода для разных браузеров
3. • в библиотеках реализованы популярные функции, что экономит время при написании кода
4. • для библиотек существует множество плагинов
5. • библиотеки поддерживают сложные визуальные эффекты

**Преимущества jQuery:**

1. • простой синтаксис и компактный код;
2. • простое и понятное обращение и манипулирование элементами веб‐страницы;
3. • множество готовых плагинов для решения разных задач;
4. • доступная документация;
5. • AJAX-дополнения;
6. • библиотека постоянно развивается добавляя новые возможности и упрощая работу с кодом

27. Фреймворк Bootstrap 5

**Bootstrap** - свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

**Bootstrap состоит из**:

1. • инструментов для создания макета (обёрточных контейнеров, мощной системы сеток, гибких медиа-обьектов, адаптивных утилитных классов);
2. • классов для стилизации базового контента: текста, изображений, кода, таблиц и figure;
3. готовых компонентов: •КНОПОК, форм, горизонтальных И вертикальных навигационных панелей, слайдеров, выпадающих списков, аккордеонов, модальных окон, всплывающих подсказок и др.,
4. • утилитных классов для решения традиционных задач наиболее часто возникающими перед веб-разработчиками: выравнивание текста, отображение и скрытие элементов, задания цвета, фона, margin и padding отступов, и т.д.

**Bootstrap5. Область применения**

1. • это фронтенд разработка сайтов и интерфейсов панелей администрирования среди аналогичных систем (Foundation, Ulkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным.

**Преимущества Bootstrap5**

1. • **высокая скорость создания качественной адаптивной вёрстки** даже начинающими веб- разработчиками достигается это благодаря использованию готовых классов и компонентов, созданных профессионалами);
2. • **кроссбраузерность и кроссплатформеннось** (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);
3. • **наличие большого количество готовЫх хорошо продуманных компонентов**, протестированных огромным сообществом веб-разработчиков на различных устройствах;
4. • **возможность настройки под свой проект**, достигается это посредством изменения SCSS переменных и использования миксинов (можно изменить количество колонок, цвета, радиус скруглений, отступы между колонками и т.д.);
5. • **низкий порог вхождения**: для работы с фреймворком не обязательно иметь «глубокие» знания по HTML, CSS, JavaScript и Query (достаточно знать только основы этих технологий);
6. • **однородность дизайна и его согласованность** между различными компонентами (в Bootstrap все компоненты выполнены в едином стиле)
7. • **наличие огромного количества сообществ и обучающих материалов**; при желании это поможет не только хорошо разобраться в фреймворке, но и найти ответы практически на любые возникающие у вас вопросы.

29. Методология БЭМ

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип

разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой

сложности и повторно использовать существующий код, избегая «Copy-Paste».

Понятия. Блок

● Функционально независимый компонент страницы, который может быть повторно использован. В HTML блоки

представлены атрибутом class.

● Название блока характеризует смысл («что это?» — «меню»: menu, «кнопка»: button), а не состояние

(«какой, как выглядит?» — «красный»: red, «большой»: big).

● Блок не должен влиять на свое окружение, т. е. блоку не следует задавать внешнюю геометрию (в виде

отступов, границ, влияющих на размеры) и позиционирование.

● В CSS по БЭМ также не рекомендуется использовать селекторы по тегам или id.

Таким образом обеспечивается независимость, при которой возможно повторное использование или перенос

блоков с места на место.

## Основные понятия БЭМ

Эффективность БЭМ-верстки достигается благодаря разделению кода:

* на независимые блоки;
* элементы (дочерние компоненты) блока;
* модификаторы блоков и элементов.

Рассмотрим блоки, элементы и модификаторы подробнее.

## Блок

Блок — логически и функционально независимый компонент страницы: меню, боковая панель, карусель, шапка сайта и так далее. Блок включает в себя шаблоны (Pug, Handlebars) и CSS-стили, скрипты JavaScript, документацию в формате [XML](https://blog.skillfactory.ru/glossary/xml/) или Markdown, другие необходимые для реализации технологии.

Независимость блоков позволяет свободно перемещать их в пределах страницы и всего проекта. Составные части блока, например формы авторизации и поиска, можно поменять местами: они будут корректно работать и сохранят внешний вид. Внесение изменений в CSS- или JavaScript-скрипты форм не потребуется.

Элемент — неотделимая составная часть блока. Особенности элементов:

* они не существуют и не используются вне блока;
* принадлежат только одному блоку;
* могут вкладываться друг в друга.

Это необязательные компоненты: небольшие блоки могут обходиться без вложенных элементов. Методология не устанавливает жестких правил в отношении того, когда разработчику следует использовать блок, а когда — элемент. Блоки, как правило, нужно создавать в том случае, если фрагмент кода может использоваться самостоятельно. Зависимым фрагментам кода (элементам) необходима родительская сущность — блок.

Модификатор — это сущность, которая определяет внешний вид, состояние и поведение элемента или блока. Один и тот же блок, например меню, будет выглядеть по-разному в зависимости от применяемых модификаторов.

Модификаторы делятся на два типа:

**Булевые.** Применяют, когда факт наличия модификатора важнее, чем его значение. Например, если элемент имеет модификатор disabled («отключен»), значение по умолчанию равно true. Структура имени при использовании булевого модификатора выглядит так: **имя-блока\_\_имя-элемента\_имя-модификатора.**

**Ключ-значение.** Используют в тех случаях, когда значение модификатора важно. Например, если применяется блок с определенной темой оформления: **search-form\_theme\_forest.** Структура наименования модификатора элемента соответствует схеме: **имя-блока\_\_имя-элемента\_имя-модификатора\_значение-модификатора.**

Модификаторы могут изменяться в процессе работы блока в зависимости от запросов из других блоков или как реакция на [DOM-события](https://blog.skillfactory.ru/glossary/dom/). Например, если пользователь введет неверные данные в форму авторизации и нажмет кнопку «Войти», DOM-событие click («Нажатие») изменит свойство блока с сообщением об ошибке с невидимого на видимое.

Наличие модификаторов у элементов и блоков — опционально, количество — не ограничено. Следует помнить, что блоку и его элементу нельзя присваивать разные значения одного и того же модификатора. По правилам БЭМ, название модификатора должно характеризовать: внешний вид блока или элемента; состояние компонента; поведение.

Миксом в терминологии БЭМ называется способ использования разных БЭМ-сущностей на одном DOM-узле таким образом, чтобы не возникало дублирования кода. С помощью миксов можно: задавать одинаковое форматирование для различных элементов дизайна; создавать новые компоненты интерфейса на основе имеющихся; совмещать поведение и стили нескольких компонентов.

## Файловая структура

Для организации проектов методология БЭМ предусматривает компонентный подход: все блоки, элементы и модификаторы реализуются в независимых файлах-технологиях — например, footer.css, footer.js. Это позволяет: подключать компоненты опционально; разрабатывать блоки проекта независимо друг от друга; переносить блоки между проектами.

Каждый блок хранится в отдельной директории; в поддиректориях располагаются элементы, модификаторы и все дополнительные ассеты —изображения, видео, шрифты, скрипты.

БЭМ предоставляет фронтенд-разработчикам комплексное решение для создания архитектуры проекта и организации рабочего процесса. Однако БЭМ не единственная методология — профессиональные веб-разработчики используют и другие решения для оптимизации CSS и переиспользования компонентов:

OOCSS; SMACSS; SUIT CSS; Systematic CSS. Каждая из этих концепций обладает набором собственных преимуществ и подходит для реализации определенных проектов.

30. Web API

API (Application Programming Interface - интерфейс программных

приложений) — это установка функций и правил позволяющая

взаимодействовать между программным обеспечением,

которое предоставляет API и другими программными

компонентами.

● В Веб разработке, под API обычно подразумевают набор

стандартных методов, свойств, событий и URL ссылок для

взаимодействия с Веб контентом.

● Они "скрывают" более сложный код от программиста,

обеспечивая простоту использования.

**API браузера** встроены в веб-браузер и способны использовать данные

браузера и компьютерной среды для осуществления более сложных

действий с этими данными.

**Сторонние API** не встроены в браузер по умолчанию.

Такие API и информацию о них обычно необходимо искать в

интернете.

Как работает API?

● Они основаны на объектах

● У них узнаваемые точки входа

● Они используют события для управления состоянием

● У них есть дополнительные средства безопасности там, где

это необходимо

Функциональность WebAPI подвержена тем же соображениям

безопасности, что и JavaScript или другие веб-технологии, но иногда они

содержат дополнительные механизмы защиты.

● К примеру, некоторые из наиболее современных WebAPI работают только со

страницами, обслуживаемыми через HTTPS в связи с передачей

конфиденциальных данных (примеры: Service Workers и Push).

31. LocalStorage

Веб-хранилище — одна из функций HTML5. Она позволяет веб-

приложениям сохранять данные локально, в браузере пользователя.

● Веб-хранилище имеет две основные формы — локальное и сессионное

хранилище (соотв. LocalStorage и Session Storage).

● Ключевая разница между ними состоит в том, что данные, сохраненные

в LocalStorage, остаются в браузере до тех пор, пока их оттуда не удалят

намеренно.

● Данные, сохраненные в SessionStorage, удаляются после закрытия

вкладки или окна браузера.

Локальное хранилище — одна из форм веб-хранилища, предоставляемая

браузером. LocalStorage позволяет веб-приложениям хранить данные

локально, в браузере пользователя, причем без ограничения по времени. В

результате сохраненные данные остаются доступными, даже если закрыть

вкладку или окно браузера.

● Обратите внимание, что локальное хранилище привязано к браузеру

пользователя на конкретном устройстве, с которого был совершен вход на

сайт. Это означает, что тот же пользователь, зашедший на тот же сайт через

другой браузер или с другого устройства, не будет иметь доступа к данным,

сохраненным в LocalStorage первого браузера.

JavaScript localStorage – это основное хранилище данных, которое находится в

объекте Window браузера. Вы можете сохранять любую информацию в localStorage, и

она будет сохраняться даже при перезагрузке страницы или закрытии и повторном

открытии браузера. localStorage и связанное с ним sessionStorage являются частью

API веб-хранилища.

LocalStorage – это часть API веб-хранилища. Он позволяет сохранять постоянные

данные (данные сохраняются при перезагрузке браузера и даже при закрытии и

повторном открытии браузера) в объекте Window браузера в виде пар ключ/значение

строк.

Существует 5 методов для работы с LocalStorage:

● setItem – устанавливает значение для указанного ключа,

● getItem – возвращает значение, связанное с указанным ключом,

● removeItem – удаляет пару ключ/значение по указанному ключу,

● clear – удаляет все пары ключ/значение из LocalStorage,

● key – возвращает имя ключа по указанному индексу.