3) Адаптивный веб-дизайн (RWD). Медиа-запросы (media queries). Формула ширины элемента. Размер текста

Адаптивный веб-дизайн (англ. Adaptive Web Design) — дизайн веб-страниц, обеспечивающий правильное отображение сайта на различных устройствах, подключённых к интернету, и динамически подстраивающийся под заданные размеры окна браузера. Целью адаптивного веб-дизайна является универсальность отображения содержимого веб-сайта для различных устройств.

Медиа запросы (media queries) - это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, это конструкции, которые позволяют определять на основании некоторых условий какие стили необходимо использовать на веб-странице, а какие нет. Основные типы устройств:

* all - все устройства (по умолчанию).
* print - принтеры и режим предварительного просмотра страницы перед печатью.
* screen - устройства с дисплеями.

Общая ширина и высота элементов представляет собой область окна браузера, которая состоит из ширины и высоты внутренних отступов, границ и указанных для них пользовательских значений.

Для изменения параметров в таблице стилей css существует специализированное свойство font-size, используя которое можно задать размер текста css-стилями в любых, удобных для вас единицах измерения. При этом можно использовать абсолютные и относительные единицу измерения.

4) Доступность веб-страниц (accessibility). Основные принципы

Веб-доступность предполагает проектирование и разработку веб-сайтов, приложений и технологических решений с учётом возможности их использования людьми с нарушениями здоровья. Иными словами, пользователи могут самостоятельно:

* получать, толковать, искать информацию и общаться посредством сети Интернет
* участвовать в работе Всемирной сети

Веб-доступность охватывает все виды нарушений здоровья, которые влияют на возможность использования Всемирной паутины, включая:

* Нарушения слуха
* Нарушения познавательной деятельности
* Неврологические нарушения
* Нарушения физического развития
* Нарушения речевой деятельности
* Нарушения зрения

Веб-доступность также отвечает интересам здоровых людей, в частности:

* обладателей мобильных телефонов, «умных часов», телевизоров с технологией Smart TV и других устройств, которые характеризуются наличием малого экрана, специфических режимов ввода данных и т.п.
* пожилых людей, испытывающих возрастные затруднения;
* людей с «временной недееспособностью», обусловленной, к примеру сломанной рукой или потерей очков;
* людей в условиях «ситуационных ограничений», таких как яркий солнечный свет или окружающая обстановка, запрещающая прослушивание аудиозаписей;
* пользователей, ограниченных низкоскоростным соединением, лимитированным или дорогостоящим Интернет-трафиком.
  1. CSS-фреймворки (библиотеки), преимущества и недостатки

CSS-фреймворк — фреймворк, созданный для упрощения работы верстальщика, быстроты разработки и исключения максимально возможного числа ошибок вёрстки (проблемы совместимости различных версий браузеров и т. д.). Как и библиотеки скриптовых языков программирования, CSS-библиотеки, обычно имеющие вид внешнего css-файла, «подключаются» к проекту (добавляются в заголовок веб-страницы).

Плюсы CSS-фреймворков:

* Кроссбраузерность
* Возможность создать корректный HTML макет даже не очень опытному специалисту
* Единообразие кода
* Увеличение скорости разработки

Минусы:

* Привязанность к стилю CSS библиотеки
* Избыточный код
  1. CSS-препроцессоры. Преимущества и недостатки

CSS препроцессор — это надстройка над CSS, которая расширяет возможности для стандартного CSS, с помощью новых синтаксических конструкций. Какие преимущества дают нам препроцессоры, для того чтобы сделать наш код более удобным, и поддерживаемым? Это, прежде всего, вложенность. Такая особенность дает нам возможность структурировать элементы, с легкостью найти родителя элемента, быстро писать псевдоклассы и псевдоэлементы и даже использовать БЭМ! Вторым, и не менее важным, преимуществом, являются переменные. Принцип работы их предельно прост: имя переменной начинается с символа $ , а дальше само имя. Через двоеточие пишем значение - это может быть цвет в любом удобном виде, значения отступа, ширины, размера шрифта и так далее. Третье, и самое большое, что может нам предложить CSS-препроцессоров - это использования миксинов. В привычном для нас понимании, миксины - это функции, которые можно использовать несколько раз, не повторяя одни и те же части кода. Кроме того препроцессоры хорошо совместимы с [разными языками программирования](https://merehead.com/ru/blog/popular-programming-languages-2020/).

* 1. Методология БЭМ. Базовые принципы и именование классов

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».

Блок

Функционально независимый компонент страницы, который может быть повторно использован. [Название блока](https://ru.bem.info/methodology/naming-convention/#%D0%98%D0%BC%D1%8F-%D0%B1%D0%BB%D0%BE%D0%BA%D0%B0) характеризует смысл («что это?» — «меню»: menu, «кнопка»: button), а не состояние («какой, как выглядит?» — «красный»: red, «большой»: big).

* Блоки можно вкладывать друг в друга.
* Допустима любая вложенность блоков.

Элемент

Составная часть блока, которая не может использоваться в отрыве от него.

Особенности:

* [Название элемента](https://ru.bem.info/methodology/naming-convention/#%D0%98%D0%BC%D1%8F-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0) характеризует смысл («что это?» — «пункт»: item, «текст»: text), а не состояние («какой, как выглядит?» — «красный»: red, «большой»: big).
* Структура полного имени элемента соответствует схеме: имя-блока\_\_имя-элемента. Имя элемента отделяется от имени блока двумя подчеркиваниями (\_\_).
* Элементы можно вкладывать друг в друга.
* Допустима любая вложенность элементов.
* Элемент — всегда часть блока, а не другого элемента. Это означает, что в названии элементов нельзя прописывать иерархию вида block\_\_elem1\_\_elem2.

Элемент — **всегда часть блока** и не должен использоваться отдельно от него.

Элемент — необязательный компонент блока. Не у всех блоков должны быть элементы.

Создавайте блок

Если фрагмент кода может использоваться повторно и не зависит от реализации других компонентов страницы.

Создавайте элемент

Если фрагмент кода не может использоваться самостоятельно, без родительской сущности (блока).

Модификатор

Cущность, определяющая внешний вид, состояние или поведение блока либо элемента.

Особенности:

* [Название модификатора](https://ru.bem.info/methodology/naming-convention/#%D0%98%D0%BC%D1%8F-%D0%BC%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D0%B0-%D0%B1%D0%BB%D0%BE%D0%BA%D0%B0) характеризует внешний вид («какой размер?», «какая тема?» и т. п. — «размер»: size\_s, «тема»: theme\_islands), состояние («чем отличается от прочих?» — «отключен»: disabled, «фокусированный»: focused) и поведение («как ведет себя?», «как взаимодействует с пользователем?» — «направление»: directions\_left-top).
* Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (\_).

С точки зрения БЭМ-методологии модификатор не может использоваться в отрыве от модифицируемого блока или элемента. Модификатор должен изменять вид, поведение или состояние сущности, а не заменять ее.

* 1. Особенности формата SVG. Плюсы и минусы векторной графики

SVG (Scalable Vector Graphics) — язык разметки векторной графики, созданный W3C, подмножество языка XML.

SVG легко модифицируется (причём, как в графическом редакторе, так и на самой странице средствами CSS). Поменять расположение, форму, размер, пропорции, цвет, заливку и все прочие свойства составляющих частей изображения проще, чем в случае с растровой графикой.

Для экранов с повышенной плотностью пикселей достаточно одного изображения. В случае с растровой графикой на данный момент (весна 2015 г.) нужно, минимум, три (!) версии картинки: 100% макетного размера, 200% и 300%. В случае использования SVG, достаточно одной версии — как и для любого векторного формата, пиксели, нужные для показа изображения, «возникают» непосредственно перед показом, исходя из необходимого пиксельного размера.  
  
SVG быстро загружается. Во-первых, SVG-файлы, как правило, имеют меньший размер файла, чем их растровые версии (исключения составляют изображения малого пиксельного размера и сложные векторные изображения со множеством форм). Во-вторых, в один SVG-файл можно добавить несколько версий изображения и показывать их по определённым условиям (уменьшение количества запросов к серверу). В-третьих, в SVG можно использовать «клонирование» — единожды прописать форму (градиент, текстуру) и многократно использовать её, ссылаясь на оригинал. Есть, правда, и минус: отрисовка SVG в браузере чуть медленнее, чем показ растрового изображения, но чтобы заметить эту разницу, нужно сравнивать большие и сложные изображения.

SVG легко сделать адаптивным. В том случае, когда необходимо на узких вьюпортах показывать логотип сайта в [упрощённом варианте](http://tympanus.net/Tutorials/ResponsiveSVGs/index6.html), SVG позволяет добиться этого всего за один запрос к серверу.  
  
SVG может быть интерактивным. Внутри изображения могут быть ссылки, скрипты, интерактивные части могут реагировать на наведение и другие действия пользователя, можно [добавить анимацию](http://www.polygon.com/a/ps4-review).  
  
Свободно распространяемое ПО. Для работы с SVG не обязательно использовать Adobe Illustrator (как стандарт в мире векторной графики), достаточно свободно распространяемого [Inkscape](https://inkscape.org/ru/). Есть и [ряд других инструментов](http://www.3dnews.ru/774197).

Основная проблема — совместимость с разными браузерами. Такие картинки зачастую просто не отображаются и нужно много «танцев с бубнами». нужен простой и универсальный формат для отображения векторных изображений на страницах. Вполне возможно, что SVG скоро станет стандартом.

* 1. Документ и объекты страницы. Окружение: DOM, BOM
* Объектная модель документа (Document Object Model, DOM) – это интерфейс программирования приложений (Application Programming Interface, API) для XML, который был расширен также для работы с HTML.
* В DOM всё содержимое страницы (элементы и текст) представляется как иерархия узлов.
* При описании древовидной структуры DOM используется терминология, которая была заимствована у генеалогических деревьев.
* Так, узел, расположенный непосредственно над данным узлом, называется родительским по отношению к данному узлу. Узлы, расположенные на один уровень ниже данного узла, называются дочерними по отношению к данному узлу. Узлы, находящиеся на одном уровне и имеющие одного и того же родителя, называются сестринскими или братьями. Узлы, расположенные на любое число уровней ниже данного узла, называются его потомками. Родительские, прародительские и любые другие узлы, расположенные на любое число уровней выше данного узла, называются его предками.

BOM (Browser Object Model в переводе с анг. - Объектная Модель Браузера) обеспечивает доступ к окну браузера и позволяет манипулировать им и его элементами.

BOM-объекты обеспечивают доступ к функционалу браузера независимо от контента веб-страницы. BOM — это объекты для работы с чем угодно, кроме документа. Доступ к фреймам, запросы к серверу, функции alert/confirm/prompt — все это BOM. Большинство возможностей BOM стандартизированы в HTML5, но браузеры любят изобрести что-нибудь своё, особенное.

10) Регулярные выражения. Примеры

Регулярные выражения (regular expressions, RegExp) — наборы символов, применяемых для поиска текстовых строк, соответствующих требуемым условиям. Результат применения регулярного выражения — подмножество данных, отобранное согласно логике, заложенной в выражении. Регулярные выражения применяются в любых задачах по поиску в множестве данных, для которых нужно получать выжимку по определенным правилам.

Большинство символов в регулярных выражениях представляют сами себя, за исключением группы специальных символов «[ ] \ / ^ $ . | ? \* + ( ) { }». Если эти символы нужно представить в качестве символов текста, их следует экранировать обратной косой чертой «\».

Если эти спецсимволы встречаются без обратной косой черты, значит у них особенные значения в регулярных выражениях:

* «^» — каретка, циркумфлекс или просто галочка. Начало строки;
* «$» — знак доллара. Конец строки;
* «.» — точка. Любой символ;
* «\*» – знак умножения, звездочка. Любое количество предыдущих символов;
* «+» – плюс. 1 или более предыдущих символов;
* «?» – вопросительный знак. 0 или 1 предыдущих символов;
* «( )» – круглые скобки. Группировка конструкций;
* «|» – вертикальная линия. Оператор «ИЛИ»;
* «[ ]» – квадратные скобки. Любой из перечисленных символов, диапазон. Если первый символ в этой конструкции – «^», то массив работает наоборот – проверяемый символ не должен совпадать с тем, что перечислено в скобках;
* «{ }» – фигурные скобки. Повторение символа несколько раз;
* «\» – обратный слеш. Экранирование служебных символов.

Регулярные выражения для проверки знаний:

* выбрать все страницы;
  + (.\*) — любое количество любых символов;
* выбрать все страницы с https;
  + ^https.\* — все URL, начинающиеся с https;

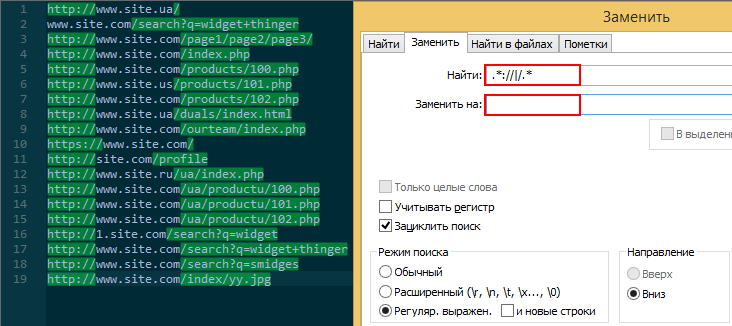
SEO-специалисты прибегают к регулярным выражениям при работе с Google Analytics, Яндекс.Метрикой, RewriteRule в .htaccess, в текстовых редакторах

Расскажу о нескольких регулярных выражених, которые часто мне помогают.

1. Выделить всё, кроме домена:

.\*://|/.\*

Использую, когда есть большой список URL-адресов (например, внешних ссылок) и для анализа нужно вычленить только домен. В NotePad++ с помощью функции замены меняю на пустую строку и получаю чистый список доменов:



Выделить URL заданной вложенности:

.\*://site.com/.\*?/.\*?/.\*?/

Здесь конструкция (/.\*?/) обозначает один уровень вложенности.

11) Обработка событий в JavaScript

Обработка событий в JavaScript - это, пожалуй, самое "вкусное блюдо", т.к. именно обработка событий выдает ответную реакцию на действия пользователей.

**События в JS** разделяются на **системные**: загрузка (load) и выгрузка (unload) страницы,**события мыши** ([click](http://html-plus.in.ua/obrabotka-sobytiya-onclick/" \t "_blank), [mouseover](http://html-plus.in.ua/obrabotka-sobytiy-onmouseover-i-onmouseout/" \t "_blank) и т.д.), mousemove и **события клавиатуры** ([keypress](http://html-plus.in.ua/obrabotka-sobytiya-onkeypress/" \t "_blank), [keydown](http://html-plus.in.ua/obrabotka-sobytiya-onkeydown/" \t "_blank), [keyup](http://html-plus.in.ua/obrabotka-sobytiya-onkeyup/" \t "_blank)).

Для того чтобы написать ответную реакцию на событие, создают **обработчик события** (event handler), который, как правило, **представляет собой функцию**.