МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
БОРИСОГЛЕБСКИЙ ФИЛИАЛ
(БФ ФГБОУ ВО «ВГУ»)

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
Разработка и поддержка сайта работника сферы образования

Методические указания для обучающихся по освоению дисциплины

Приступая к изучению учебной дисциплины, прежде всего обучающиеся должны ознакомиться с учебной программой дисциплины. Электронный вариант рабочей программы размещён на сайте БФ ВГУ.

Знание основных положений, отраженных в рабочей программе дисциплины, поможет обучающимся ориентироваться в изучаемом курсе, осознавать место и роль изучаемой дисциплины в подготовке будущего педагога, строить свою работу в соответствии с требованиями, заложенными в программе.

Основными формами контактной работы по дисциплине являются лекции и лабораторные занятия, посещение которых обязательно для всех студентов (кроме студентов, обучающихся по индивидуальному плану).

В ходе лекционных занятий следует не только слушать излагаемый материал и кратко его конспектировать, но очень важно участвовать в анализе примеров, предлагаемых преподавателем, в рассмотрении и решении проблемных вопросов, выносимых на обсуждение. Необходимо критически осмысливать предлагаемый материал, задавать вопросы как уточняющего характера, помогающие уяснить отдельные излагаемые положения, так и вопросы продуктивного типа, направленные на расширение и углубление сведений по изучаемой теме, на выявление недостаточно освещенных вопросов, слабых мест в аргументации и т.п.

Не следует дословно записывать лекцию, лучше попытаться понять логику изложения и выделить наиболее важные положения лекции в виде опорного конспекта. Рекомендуется использовать различные формы выделения наиболее сложного, нового, непонятного материала, который требует дополнительной проработки: можно пометить его знаком вопроса (или записать на полях сам вопрос), цветом, размером букв и т.п. – это поможет быстро найти материал, вызвавший трудности, и в конце лекции (или сразу же, попутно) задать вопрос преподавателю (не следует оставлять непонятый материал без дополнительной проработки, без него иногда бывает невозможно понять последующие темы). Материал уже знакомый или понятный нуждается в меньшей детализации – это поможет сэкономить усилия во время конспектирования.

В ходе выполнения лабораторных работ необходимо не просто внимательно читать методические указания к работам и аккуратно выполнять все задания и упражнения, но и обращать внимание на сложные моменты (в тексте они выделены и снабжены отдельным примечанием), внимательно анализируя текст примечания и приведённые примеры, при необходимости экспериментируя и обращаясь к справочникам.

При подготовке к промежуточной аттестации необходимо повторить пройденный материал в соответствии с учебной программой, примерным перечнем вопросов, выносящихся на зачёт. Рекомендуется использовать конспекты лекций и источники, перечисленные в списке литературы в рабочей программе дисциплины, а также ресурсы электронно-библиотечных систем. Необходимо обратить особое внимание на темы учебных занятий, пропущенных по разным причинам. При необходимости можно обратиться за консультацией и методической помощью к преподавателю.

План лекций по дисциплине

План лекций по дисциплине «Разработка и поддержка сайта работника сферы образования»

Тема №1. Основы Web-конструирования

- Основные определения и понятия Web-конструирования и Web-программирования.

- Основные приемы создания и продвижения сайтов.

Тема №2. Основы проектирования и разработки Web-ресурсов

- Проектирование, разработка и маркетинг проблемно-ориентированных Web-ресурсов.
- Языки HTML, JavaScript, PHP, как средства создания информационных ресурсов Интернет.
- Web-публикация и дизайн, визуальные и семантические критерии качества

Методические материалы к лекционным занятиям по теме «Основы проектирования и разработки Web-ресурсов»

Условно процесс создания сайта (web-проекта) можно разделить на 3 этапа:

- Планирование
- Дизайн
- Разработка

Планирование

Данный этап можно разделить на несколько подэтапов:

- Создание идеи
- Разработка структуры проекта
- Проработка макета проекта

Создание идеи

На данном этапе нам необходимо определиться с тематикой проекта (сайта, сервиса). Далее, в соответствии с выбранной темой, необходимо собрать соответствующие материалы: текстовые, графические.

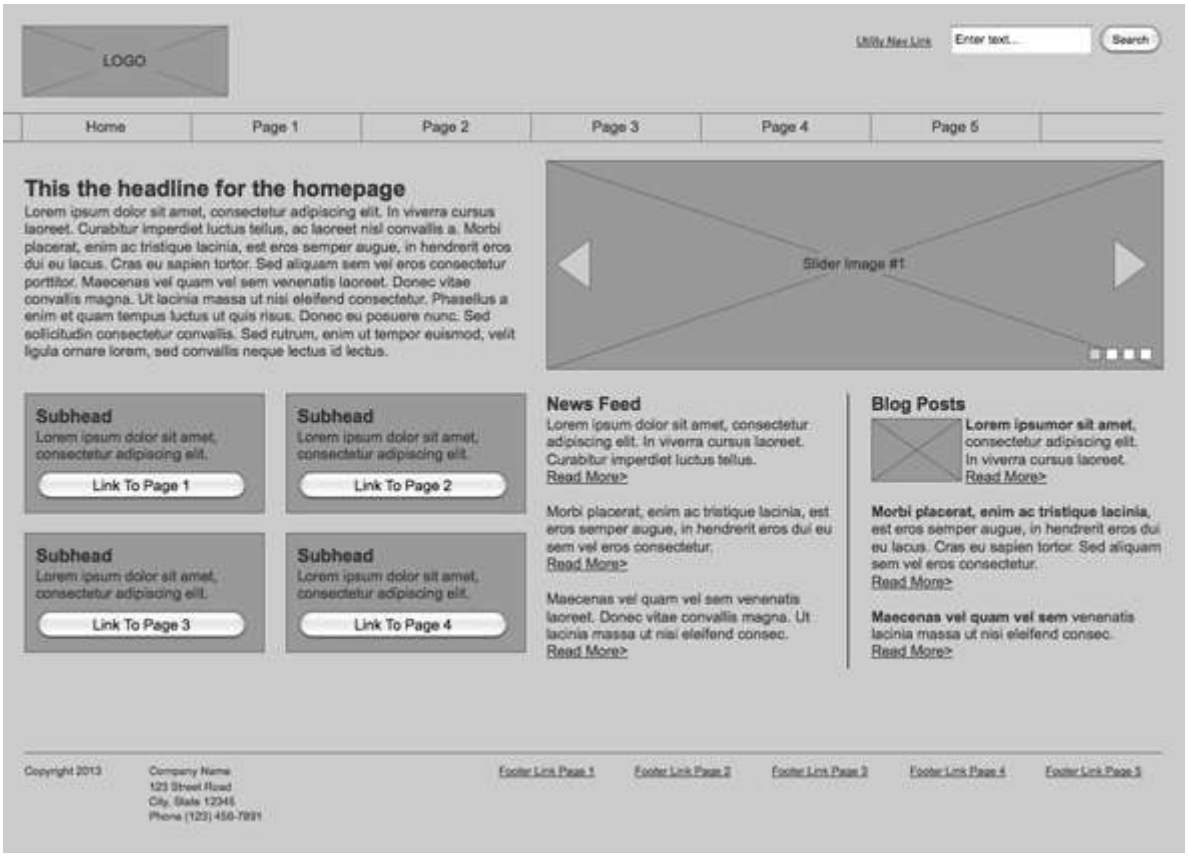
Разработка структуры проекта

Когда мы определились с темой проекта, подобрали необходимый материал, следующим этапом будет разработка структуры проекта. Структура проекта подразумевает под собой разделы сайта, в соответствии с которыми будет формироваться навигационное меню и строиться дизайн проекта. На данном этапе можно классифицировать материал по темам и разделам.

Проработка макета проекта

После того, как мы определились со структурой проекта можно составить макет проекта (схематично).

Для отрисовки наброска можно использовать бумагу и ручку, Photoshop, любой другой редактор графики (раньше часто использовали Adobe Fireworks). Важно отметить, что данный этап – это не отрисовка готового дизайн-макета, а всего лишь схематичный набросок, выполненный для понимания того, как на сайте будут располагаться основные информационные блоки, графика и прочие элементы дизайна.



Основные элементы страницы

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).

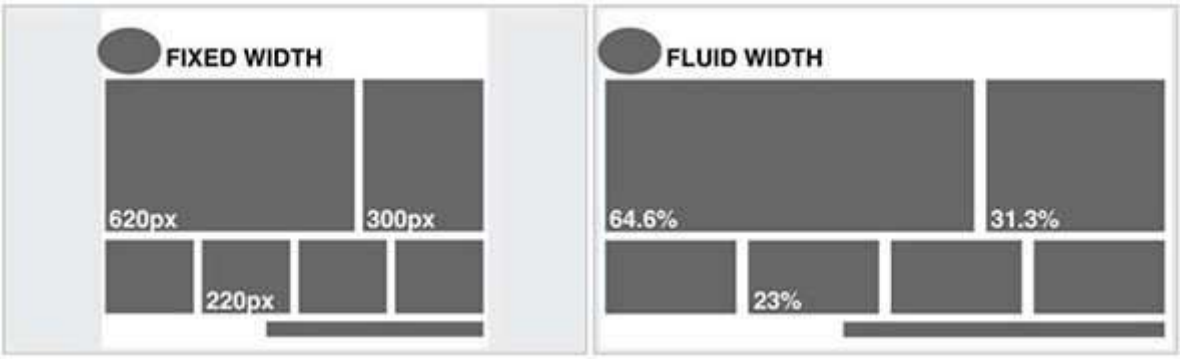
Контент

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.

Резиновый и фиксированный макет



Фиксированный макет

Фиксированный макет подразумевает под собой, что в независимости от разрешения экрана пользователя ваш сайт всегда будет занимать одинаковую ширину.

Резиновый макет

«Резиновый» макет подразумевает, что страница сайта будет стараться занять всё доступное ей пространство на экране пользователя, подстраиваясь под разрешение.

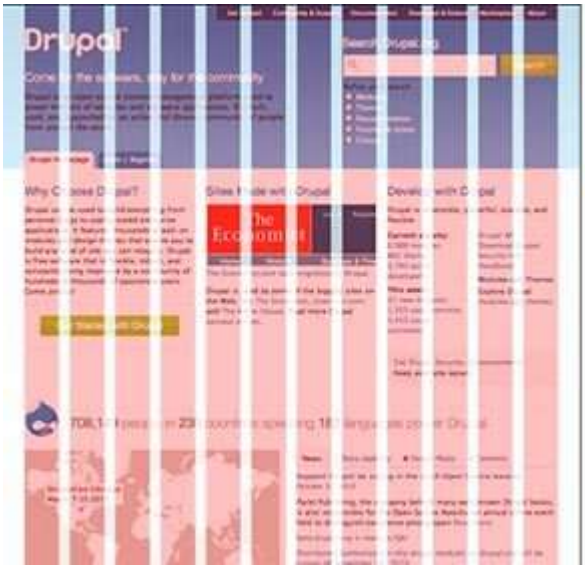
В данном контексте стоит уяснить такие понятия, как **отзывчивый веб-дизайн (Responsive Web Design aka. RWD)** и **адаптивный веб-дизайн (Adaptive Web Desing aka. AWD)**. Первое понятие укладывается в концепцию «резинового» и означает, что при изменении размера экрана ваш сайт подстраивается под него, второе понятие подразумевает, что при разработке вы определяете основные разрешения (размеры экрана), под которые будет подстраиваться (адаптироваться) ваш контент. В обоих случаях следует разрабатывать не один, а несколько макетов, которые будут соответствовать разным разрешениям экрана. Часто создаётся 3 макета под разрешения iPhone (Android Phone), iPad (Android Tablet) и Desktop.

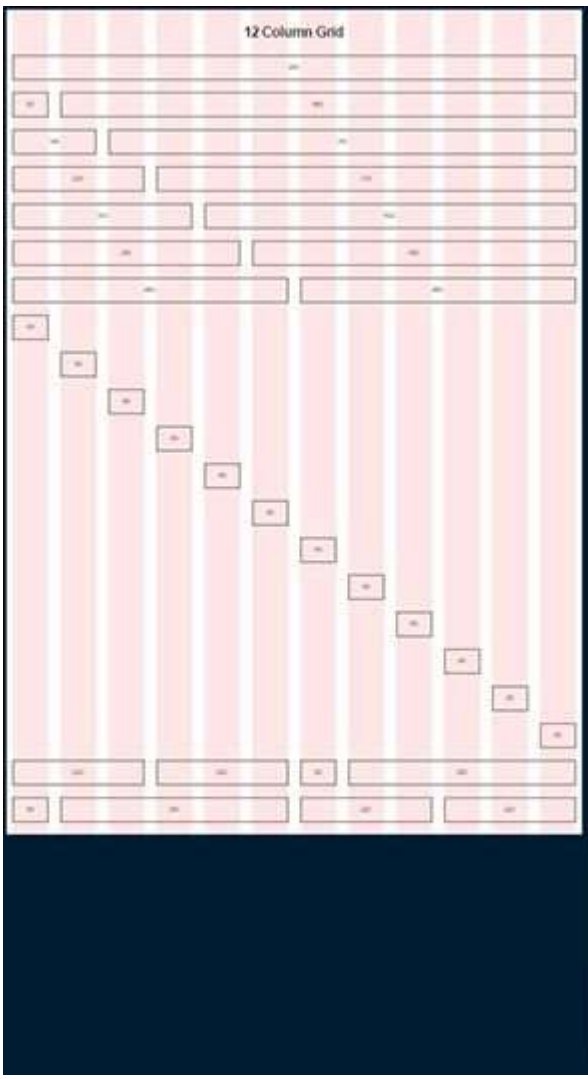
При разработке макета мобильной версии сайта стараются на первый план выносить основной контент, поэтому навигационное меню часто прячется, скрываются большие баннеры и декоративные элементы, блоки контента обычно располагают друг под другом. На заранее составленном макете как раз можно определиться какие элементы мы оставляем на мобильном, а какие прячем.

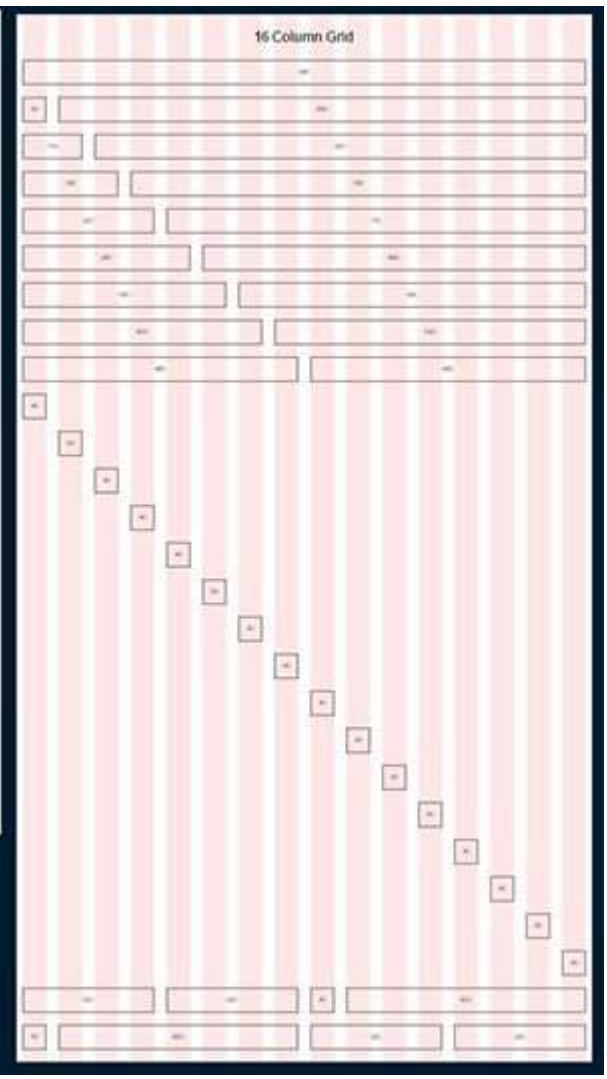
Модульная сетка

Перед составлением схемы проекта так же необходимо уяснить понятие модульной сетки. Модульная сетка подразумевает под собой разделение страницы на отдельные колонки по вертикали и выстраивание контента, при разработке дизайн макета, именно по этой сетке.

Наиболее популярной системой является модульная сетка 960 Grid System (http://960.gs), которая максимально делит страницу на 12, 16 и 24 колонки. Максимум в ширине сетка имеет 960 пикселей. Данное решение основано на том, что большинство современных мониторов, на момент создания сетки, имели разрешение не меньше 1024 на 768 пикселей. Создание макета на основе данной сетки, в дальнейшем, поможет ускорить процесс разработки (вёрстки).







Также стоит отметить, что при разработке «резинового» макета страницы существует понятие максимальной ширины. Данное утверждение основывается на удобстве восприятия информации. Если предположить, что наш сайт не имеет максимального значения по ширине, то на больших мониторах информация будет сильно растягиваться и её неудобно будет читать. Чаще всего ограничиваются шириной в 1280 пикселей.

Модульная сетка 960GS отвечает концепции «фиксированного» дизайна, для «резинового» дизайна можно обратить внимание на адаптацию этой же сетки на сайте http://www.designinfluences.com/fluid960gs/ или воспользоваться сеткой, которую предлагает фрэймворк Bootstrap (http://getbootstrap.com/css/#grid).

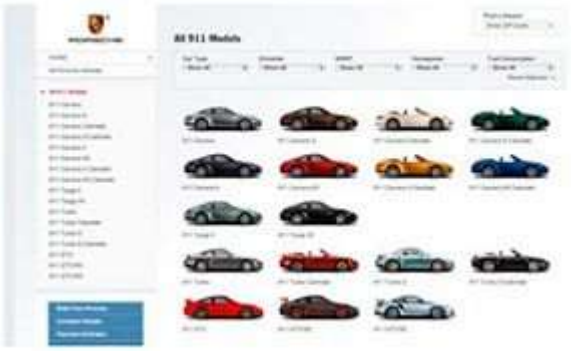
Благодаря модульной сетке блоки контента и элементы будут располагаться на определённом расстоянии друг от друга, будут иметь удобоваримую ширину, что в дальнейшем визуально будет приятно пользователю и не будет вызывать у него какие-либо неудобства в восприятии сайта.

Модульная сетка, по сути, – это некая визуальная абстракция, визуальное деление страницы на равные по ширине столбцы с равными отступами между ними. Визуализировать данную модель можно посредствам направляющих или отдельного слоя, на котором будут изображены эти столбцы. Именно такое решение вы найдёте в шаблонах сетки 960gs.

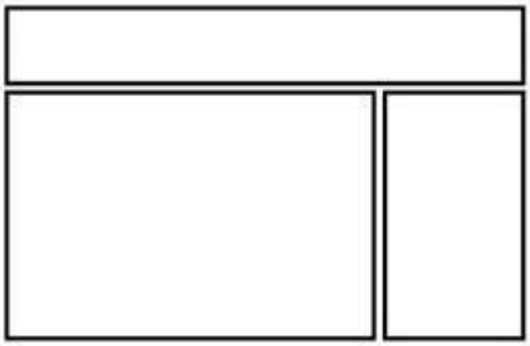
Макеты веб-страниц

Среди всего многообразия составления макета веб-страницы можно выделить четыре наиболее распространённых:

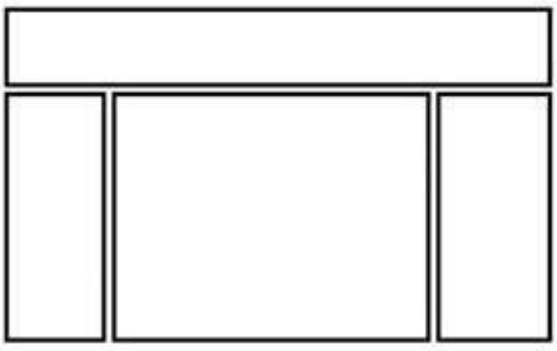
- Навигация в левом столбце



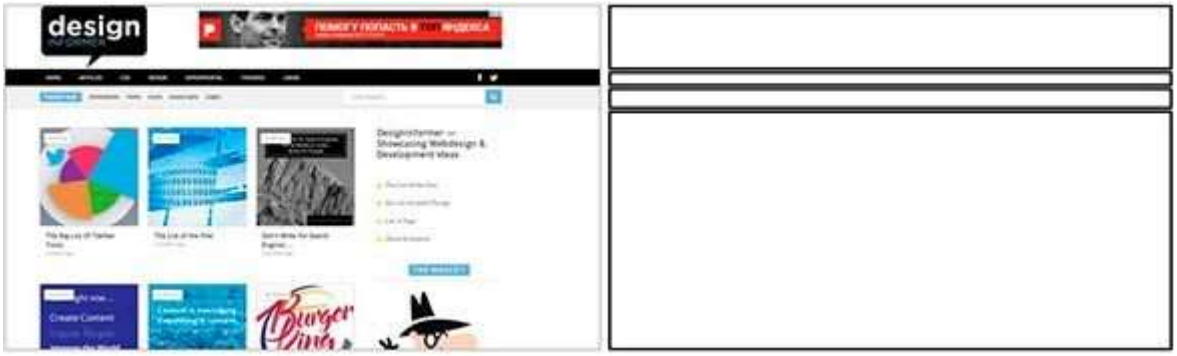

- Навигация в правом столбце

- Навигация в трёх столбцах

- Горизонтальная навигация. На данном этапе сайты с таким типом навигации составляют большинство. Удобство такого подхода легко объяснить тем, что в данном случае у нас остаётся больше пространства для контента, составляющего наш сайт.



Mobile First

С учётом тенденций последних лет данный подход плотно занимает свою нишу в разработке и дизайне сайтов. Тенденция такова, что практически около 60% пользователей интернета используют для доступа в сеть мобильные устройства, поэтому правилом хорошего тона становится разработка не только десктопной версии сайта, но и мобильной версии. При использовании данного подхода разработка макета сайта, дизайна и вёрстки начинается с мобильной версии, а затем уже прорабатываются макеты для других разрешений: добавляются блоки, баннеры, дополнительные элементы дизайна и пр.

Дизайн

После создания макета проекта можно переходить непосредственно к созданию дизайн-макета. На данном этапе начать стоит с определения цветовой гаммы проекта.

Один из способов определения основного цвета в проекте – это составление mood board. Для этого необходимо выписать себе все синонимы, связанные с темой проекта, а затем каждый синоним набрать в поиске по картинкам Google или Yandex. На основе найденных изображений выписать себе цвета, которые чаще всего встречаются на них (каких цветов больше). Найденные цвета будут составлять визуальное восприятие нашего проекта и вызывать у пользователя соответствующие чувства.

Для работы с выбранным цветом и составлением палитры цветов нашего сайта можно использовать следующие инструменты:

- Color Scheme Designer 3 (http://colorschemedesigner.com/csd-3.5/). Помимо выбора цветовой схемы данный сервис позволяет посмотреть пример того, как выбранные цвета будут смотреться на сайте

- Adobe Color CC (https://color.adobe.com/ru/). Данный ресурс, в отличие от Color Scheme Designer 3, позволяет создавать палитры ещё и на основе загруженных изображений (которые, например, могли появиться у нас на этапе составления mood board). Так же данный сервис обладает большим архивом палитр других пользователей.

- COLORlovers (http://www.colourlovers.com/). Обширное сообщество, где можно подобрать различные палитры.

Важно отметить, что при подборе цветов для палитры всегда стоит выбирать как минимум 2 контрастирующих цвета. Достижение нужного контраста между цветами – необходимое условие для того, чтобы у вас получился хороший интерактивный дизайн.

При работе над дизайном главной и внутренних страниц стоит помнить о некоторых основных принципах.

Элементы Call to Action

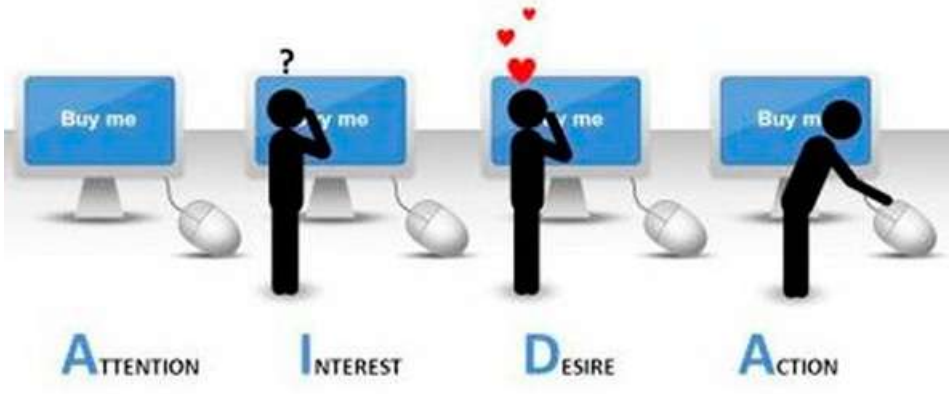
Понятие призыва к действию относится к интерактивным элементам сайта: кнопки, баннеры и пр. Данные элементы оформляются таким образом, что пользователю должно хотеться непременно на них нажать. Например, это может быть кнопка с призывом к действию (Нажми), яркий баннер с заманчивым предложением, яркой картинкой и пр.

Данное понятие хорошо вписывается в принцип AIDA (Attraction Interest Desire Action).

AIDA

Данное понятие применяется чаще при дизайне главных страниц, страниц акций и пр., где необходимо подтолкнуть пользователя к тому или иному действию: подписка, покупка и пр. Если перевести данный акроним на русский, то мы получим следующие понятия:

- Привлечение внимания
- Интерес
- Желание
- Действие

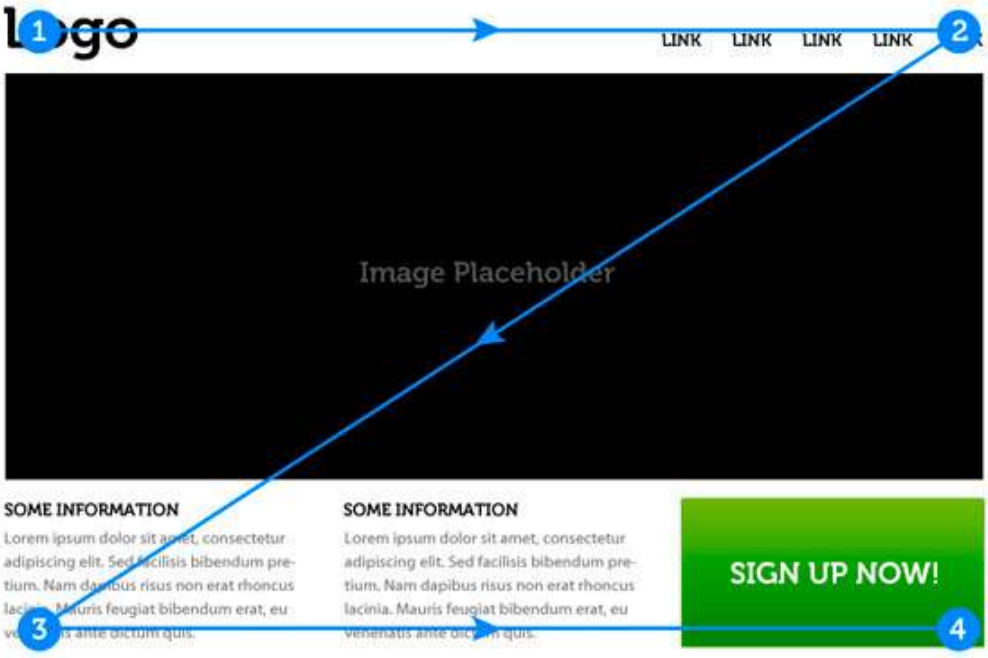


Таким образом становится понятен принцип построения дизайна, опирающегося на данное понятие: например, яркая картинка, баннер должны привлечь внимание пользователя, сопутствующий посыл в тексте должен вызвать в нём интерес и желание, а завершающим аккордом должна стать, например, кнопка с призывом к действию.

Но и данный принцип не работает сам по себе без некоторых других: **схема просмотра страницы** (наиболее, естественный путь движения глаз по странице), **визуальные направляющие**.

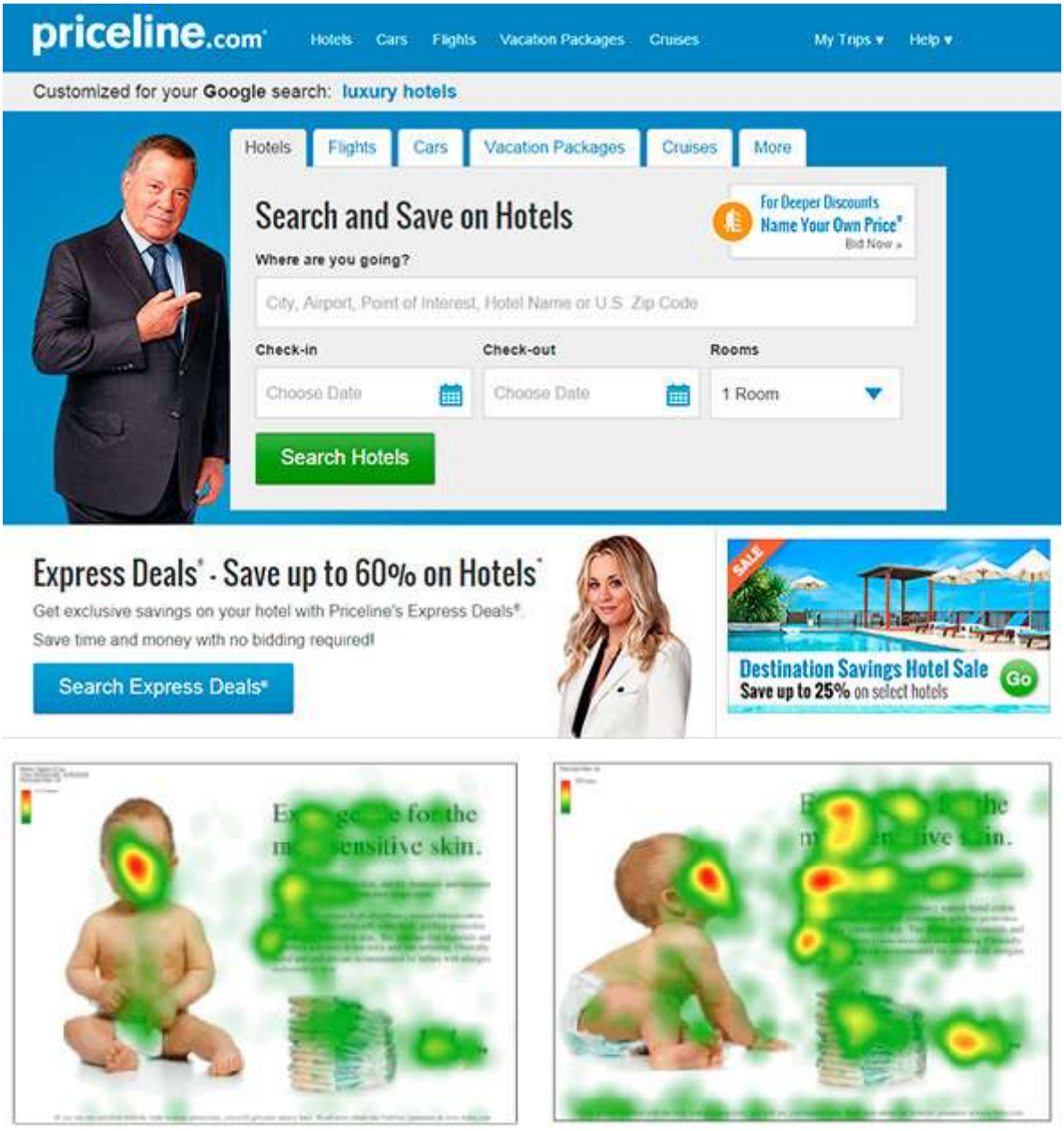
Схема просмотра страницы

Довольно часто встречается Z-схема просмотра страницы. В соответствии с этим элементы страницы обычно располагают следующим образом: логотип слева вверху, меню справа вверху, информационные блоки, картинки слева внизу, кнопка с призывом к действию справа внизу.



Визуальные направляющие

Визуальными направляющими называют декоративные элементы страницы, которые перенаправляют взгляд пользователя на те или иные элементы дизайна, формы, кнопки и пр. В качестве визуальных направляющих могут выступать стрелка, направление взгляда человека на изображении, направление указательного пальца, в общем всё, что может как-то указывать в ту или иную сторону.



На первом изображении взгляд невольно следует за указательным пальцем мужчины, а его прямой взгляд невольно привлекает внимание к себе при первом взгляде на страницу.

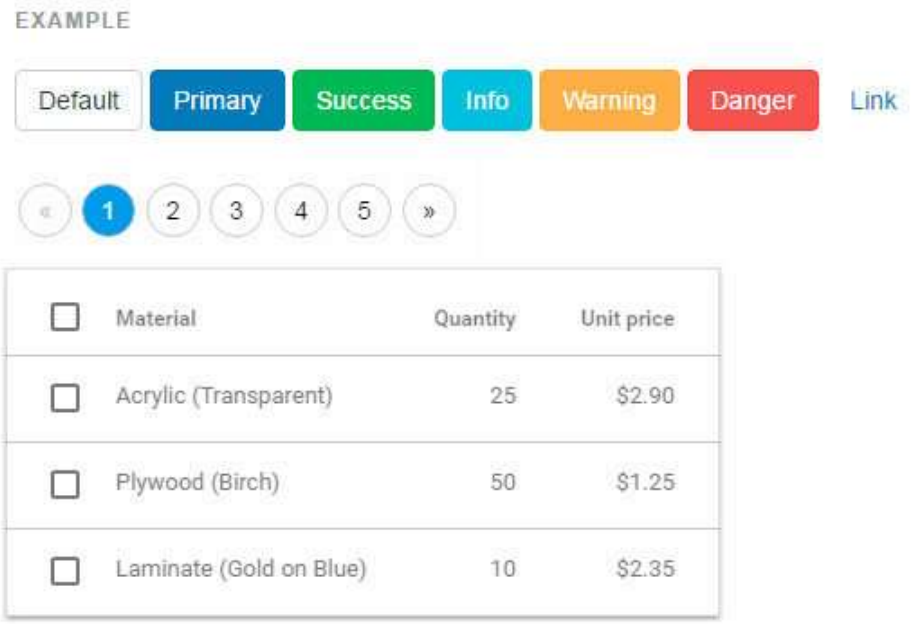
Тот же эффект продемонстрирован на втором изображении с наложенной тепловой картой: в первом случае нас привлекает в основном прямой взгляд ребенка (прямо в глаза), во втором случае направление взгляда ребенка и его поза невольно обращают взгляд пользователя на блок справа.

Фрэймворки

Стоит так же отметить, что иногда, при разработке дизайна страницы, используют фрэймворки Bootstrap, Foundation, Material Design Lite, которые, помимо готовых

элементов дизайна (кнопки, формы ввода и пр.), предлагают свою модульную сетку, CSS сниппеты (часть кода, разметки, которая может неоднократно использоваться) для вставки элементов в страницу (тех же кнопок, элементов форм и пр.) и классы разметки, а так же JS скрипты для соответствующих интерактивных элементов.

Использование данных библиотек может в значительной мере сэкономить время при разработке проекта (дизайне, вёрстке), правда в то же время может сделать ваш сайт похожим на другие, если использовать элементы дизайна фрэймворков как есть. На основе каждого фрэймворка можно найти огромное количество платных и бесплатных тем и страниц, а так же разработать свои.



Готовые элементы дизайна из Bootstrap, Foundation и Material Design Lite (MDL)

Плоский дизайн

Плоский дизайн использует минимум текстур и оформление, создание дизайна базируется на понятиях контраста, соотношения цветов и размеров.

Немаловажное влияние на тренды в веб-дизайне оказало развитие мобильных операционных систем. Чаще всего двигателем прогресса становятся решения от

компании Apple, которая в своих дизайн-решениях iOS сначала использовала имитацию реальных объектов, а затем всё упростила до плоского дизайна (Flat UI). Сейчас в вебе главенствует Flat дизайн и Material дизайн, который активно развивает Google.

Разработка

Итак, процесс дизайна макета страницы плавно перетекает в процесс «оживления» сделанного на предыдущих этапах. Прежде чем сразу начинать писать HTML, CSS и JS стоит немного поговорить о редакторах кода и структуре проекта.

Редакторы кода

Из наиболее популярных редакторов кода на сегодня можно выделить три:

- Sublime Text (http://www.sublimetext.com/3)
- Atom (https://atom.io/)
- Brackets (http://brackets.io/)

Отчасти, все эти редакторы похожи по принципу работы, когда при установке мы получаем редактор, в который затем можем «доставить» необходимые модули и плагины, так сказать, «редакторы на стероидах». Разница лишь в технологиях, которые были использованы при написании редакторов, если Sublime Text написан при помощи C++ и Python, то 2 других используют JavaScript, HTML, CSS (Less). За счёт этой разницы Sublime Text может работать чуть быстрее своих коллег.

Существуют также более продвинутые IDE (студии) такие, как Web Storm, PHP Storm и пр., но для вёрстки проекта вполне подойдёт редактор кода, а не целая студия (IDE).

Чтобы определиться, что подходит лично вам для разработки, стоит самим покопаться в многообразии средств и выбрать то, что больше всего удовлетворяет вашим потребностям.

Структура проекта

Под структурой проекта понимается хранение файлов проекта в его директории. Часто приходится видеть, когда все файлы «свалены» вместе, названия файлам даны «капсом», цифрами или русскими буквами и пр. Во-первых, это банальное неуважение к тому, кто будет работать с вашим проектом далее, во-вторых, чем больше будет ваш проект, тем больше будет становиться файлов и, в конце концов, вы просто запутаетесь, что к чему относится и что нужно, а что нет.

Лучше всего отдельные категории файлов помещать в свои папки: картинки в папку *images* или *img*, css в папку *css*, javascript в папку *js*. В корне будет лежать только *index.html* и страницы сайта, либо только index.html, а страницы в отдельной папке pages. Соблюдая эти правила вы никогда не запутаетесь в проекте.

Также стоит сказать и об именовании файлов проекта. Чаще всего применяются следующие имена: главная страница – это *index.html*, стили проекта *styles.css*, скрипты *scripts.js* или *app.js*, минимизированные версии файлов имеют префикс .min, картинки носят не пространные названия на русском языке или набора цифр, а

отражают то, что на них изображено, например, *button.png*, *download-icon.png*, *logo.png* и т.д.

Работа над проектом

Итак, определившись с редактором кода, структурой мы можем приступать к разработке. Прежде всего стоит отметить, что вёрстка страницы делается поэтапно: сначала пишется HTML-структура (HTML-код), затем добавляются стили, а после, если необходимо, пишутся скрипты (JS), добавляются необходимые плагины и библиотеки.

Учитывая вышесказанное, мы можем условно разделить работу над проектом на следующие этапы:

- Написание HTML
- Написание CSS
- Написание JS

Написание HTML

Сейчас при написании HTML кода уже смело можно использовать тэги и элементы разметки, которые появились вместе с стандартом HTML5, если вам необходимо поддерживать старые браузеры, то можно использовать, например, плагин html5shiv (https://github.com/afarkas/html5shiv), который обеспечивает поддержку новых стандартов в старых браузерах или библиотеку Modernizr (https://modernizr.com/) (html5shiv входит в сборку Modernizr), которая определяет возможности браузера, с помощью которого просматривается сайт.

При вёрстке сайтов в настоящее время, априори, используется блочный подход, никаких таблиц, iframe'ов и пр. Таблицы выполняют только свою прямую роль – представление информации в виде таблицы. В вёрстке таблицы используются лишь при работе с электронными письмами.

На этапе написания HTML мы, как бы, создаём скелет страницы, её абстрактную модель при помощи тэгов (языка разметки HTML). Стоит отметить, что структуру может быть проще написать, если у нас есть прототип, составленный на первом этапе или же, если мы сами, глядя на дизайн-макет, на бумаге схематично разрисовали себе все блоки страницы.

При написании разметки мы также сразу можем прописывать элементам классы и идентификаторы.

Правила именования классов

В проекте во всём должен быть порядок: от структуры проекта до имен классов, разметки и написания кода. Если при разметке важно следить за типом информации и размещением её в соответствующих блоках (заголовок, список, ссылка, строчный элемент, параграф и пр.), то при именовании классов и идентификаторов важно соблюдать здравый смысл. Классы должны давать абстрактное понятие о блоке, к которому они относятся, чтобы код было легче читать, а затем и писать стили. В принципе здесь не должно быть ничего сложного, если мы размечаем меню, то логично *содержащему блоку дать класс .nav или .navigation, если это блок с текстом, то можно дать ему класс .block-text и т.д.*

БЭМ

На сегодняшний день есть один популярный подход, который касается принципов построения проекта в целом, но на данном этапе нас интересует именно именование классов. Подход называется БЭМ и расшифровывается, как Блок Элемент Модификатор.

Вкратце можно описать данный подход, как некое соглашение по именованию классов и представлению разметки страницы. Каждый элемент страницы являет собой сущность, которая может существовать независимо от контекста, тогда мы говорим о Блоке (*.block*) или же только в контексте другой сущности, тогда мы говорим об Элементе (*.block__element*). Каждый Блок или Элемент могут иметь различные модели представления: цвет, форму, прозрачность и пр. За подобные свойства будет отвечать Модификатор (*.block__element_mod*).

Таким образом мы представляем наш код, как композицию блоков, элементов и их модификаций.

SMACSS

Также существует подход SMACSS (расшифровывается данный акроним как Scalable and Modular Architecture for CSS – Масштабируемая и Модульная Архитерктура для CSS), который разделяет понятие о классах и разметке на несколько уровней: базовый, макет, модуль, состояние, тема.

- К базовому уровню будет относиться всё, что касается непосредственно тэгов html.
- К уровню макета мы будем относить всё, что касается основных составляющих страницы: секции.
- К уровню модуль мы будем относить всё, что касается переиспользуемых элементов страницы: баннеры, навигация, списки, блоки информации и пр.
- Уровень Состояние описывает как будут выглядеть модули и секции в том или ином состоянии: отображаются или не отображаются, сжатые или раскрытые, активны или неактивны и пр.
- Уровень темы чем-то схож с уровнем Состояния и отражает как модули или секции могут выглядеть.

В данном подходе используются следующие соглашения по именованию. Уровни определяются при помощи префиксов и соответствующего буквенного обозначения:

- Макет: *.l-* или *.layout-*
- Так как модули составляют основную часть проекта именовать их, используя префикс module- избыточно. Поэтому для них используются имена как есть, например: *.example {}*, *.afisha {}* и пр.
- Состояния имеют префикс *.is-*, например *.is-hidden {}*
- Уровень Темы именуется подобно модулям.

При данном подходе часто бывает удобно каждый уровень абстракции и его классы держать в отдельном файле.

Важно помнить, что любое соглашение по именованию призвано создать определённый уровень абстракции, благодаря которому работать с проектом будет удобно и он не будет перегружен излишним кодом и прочими артефактами.

Написание CSS

Правила именования классов подводят нас к следующему этапу. Когда написана html структура проекта, определены классы можно переходить к написанию CSS стилей и нарезке макета.

Стоит упомянуть о 2-х CSS файлах-дополнениях: normalize.css и reset.css.

Reset.css

Изначально в проектах повсеместно использовался reset.css, написанный Эриком Мейером. Цель данного свода правил – сбросить стили браузера, которые он по умолчанию использует для отображения элементов разметки. Таким образом, при использовании reset.css нам не нужно переписывать стили браузера, по сути мы работаем с «чистым листом» и можем сосредоточиться на написании собственных стилей с нуля.

Normalize.css

Normalize.css, наоборот, не сбрасывает все стили «в ноль», а нормализует их, приводит отображение стилей проекта к более-менее однообразию во всех современных браузерах.

У обоих сводов правил есть свои плюсы и минусы, на данный момент популярен normalize.css. Популярность данного свода правил обусловлена ещё и тем, что вам не приходится при разработке заново прописывать основные свойства для базовых элементов страниц, а лишь модифицировать их по необходимости.

Не стоит перегружать ваши стили чрезмерной специфичностью, например, если мы хотим, чтобы ссылка внутри списка имела красный цвет, то не обязательно записывать всю линейку классов и тэгов: *.main –nav ul li a*, достаточно просто указать *.main-nav a*. Слишком «специфичные» правила делают стили контекстно зависимым и заставляют вас писать лишний код, а так же влияют на скорость отрисовки страницы, так как при разборе правил CSS парсер читает их справа налево и ему сначала, если брать описанный выше пример, нужно взять все ссылки (a), затем отбросить все ссылки, которые не лежат в li и т.д. пока он не доберётся до содержащего класса.

Специфичность кода, так же означает чрезмерную привязку к базовым тэгам. Например, если мы хотим, чтобы элемент внутри .block был синим и при разметке использовали элемент span (в правилах мы написали *.block span {background-color: blue}*), то при замене его на div, например, нам нужно будет написать новое правило уже для div внутри .block. Таким образом наш код будет обрастать ненужными правилами, которые по сути дублируют друг друга. Гораздо проще определить всего лишь один класс (класс элемента), который при присвоении внутренним элементам будет приводить нас к желаемому результату.

Ещё один пример контекстной зависимости кода – это использование имен тэгов вместе с именами классов, например, div.block {display: block}. Если мы захотим тот же класс применить к span, то нам придётся написать опять span.block {display: block} или span.block, div.block {display: block}, что в любом случае приведёт к увеличению кода на лишнюю строчку.

Почему это важно? Во-первых, вы не делаете лишнюю работу, во-вторых, вам не нужно потом среди 10000 строк искать нужное правило и, наконец, чем больше строк кода, тем больше размер файла, а большие по объему файлы медленнее передаются по сети, что в свою очередь может быть причиной долгой загрузки, а излишняя специфичность причиной долгой отрисовки вашей страницы.

Если, вы используете правила, относящиеся к спецификации CSS3, то не поленитесь проверить необходимость использования браузерных префиксов на сайте http://caniuse.com.

Возвращаясь к Mobile First, стоит сказать, что при работе в данной концепции написание стилей стоит начинать с мобильной версии, а затем с помощью медиа запросов добавлять правила, которые будут работать на других разрешениях. Зачастую, при расширении правил от мобильной версии к десктопной, приходится писать гораздо меньше кода, чем при работе наоборот: от десктопной версии к мобильной.

Написание JS

Итак, заключительным этапом является написание JS скриптов. При создании веб страниц стало практически стандартом использование библиотеки jQuery, которая позволяет с лёгкостью манипулировать элементами web-страницы (DOM узлами), навешивать прослушку событий, отправлять запросы на сервер, обрабатывать результат выполнения и пр. Но не стоит слепо доверять трендам, на сегодняшний день нативный JavaScript достиг такого уровня, что вам возможно уже не нужен jQuery. Поэтому, прежде чем слепо прикреплять jQuery, стоит задуматься, так ли он нужен для ваших задач и не достаточно ли встроенных возможностей языка JavaScript.

Важно отметить, что не рекомендуется использовать JavaScript для стилизации, то есть не стоит для того или иного объекта DOM (элемента страницы) добавлять портянку CSS свойств с помощью JavaScript, чтобы выделить его состояние и пр. Рекомендуется использовать классы, то есть заранее в CSS определить классы-состояния (активный, не активный, скрытый, использованный и пр.) и при манипуляции элементами просто добавлять или убирать соответствующие классы.

Например, у нас есть элемент списка с классом *.list__item*, для того, чтобы показать активный элемент в данный момент (выбран пользователем) или нет, можно просто добавить класс (*.list__item--active*), а не прописывать все стили, присущие этому классу в JS:

document.querySelector(‘.list__item’).style.color = ‘red’;

и т.д.

Не рекомендуется решать задачи, которые явно относятся к уровню CSS с помощью JavaScript, пытаясь как-то выровнять объект или добавить ему стили при наведении и пр.

Опять-таки, возвращаясь к теме Mobile First, нельзя не упомянуть о 2-х концепциях, которые коррелируют с данной техникой. Progressive Enhacement и Graceful Degradation, что переводится как прогрессивное улучшение и последовательное ухудшение. Данные принципы описывают 2 разных подхода к разработке: в первом случае мы разрабатываем наш сайт, пишем скрипты с учётом старых браузеров и систем или же определяем поведение при невозможности запуска скриптов, а затем постепенно улучшаем скрипт, вводя современные функции и методы, таким образом мы получаем сайт, который будет одинаково хорошо работать в старых и новых браузерах и средах (то же касается и css).

При Graceful Degradation подход противоположный – мы разрабатываем для современных браузеров и, лишь, потом начинаем вносить доработки и изменения с учётом старых версий.

Подход Mobile First чем-то схож с Progressive Enhacement.

Проверка кода

После написания html, css и js для нашей страницы необходимо проверить всё ли сделано верно. Для этого можно использовать online средства:

- Для проверки html: https://validator.w3.org/
- Для проверки CSS: http://jigsaw.w3.org/css-validator/
- Для проверки JS: http://www.jslint.com/

Благодаря данным сервисам можно проверить, не забыли ли вы где-то закрыть тэг, верно ли используете параметры и атрибуты, всё ли в порядке с вашими стилями и правилами в них, а так же проверить ваш код на правильность написания функций, методов и пр.

Статьи с рекомендациями по написанию JS, HTML и CSS:

- 10 советов по написанию нативного JavaScript без jQuery
- Сайт-сборник рекомендаций по JS, HTML и CSS

Средства автоматизации

На сегодняшний день практически все рутинные процессы разработки (за исключением дизайна) можно в той или иной степени автоматизировать.

960GS

Мы уже говорили об этом средстве разработки в разделе про модульные сетки и дизайн. Нужно сказать, что 960GS предлагает не только макеты для создания дизайна, но и свою систему именования классов. Используя предлагаемые классы при вёрстке и при подключении css файла системы в проект, блоки сайта будут выстроены по этой сетке, что сэкономит время на написании практически того же самого самостоятельно.

```
<div class="container_12">
  <div class="grid_7 prefix_1">
      <div class="grid_2 alpha">
          ...
      </div>
      <div class="grid_3">
          ...
      </div>
      <div class="grid_2 omega">
          ...
      </div>
  </div>
  <div class="grid_3 suffix_1">
      ...
  </div>
</div>
```

Emmet

Emmet – это средство работы с html и css (http://emmet.io/). Плагин для работы с данным средством можно установить, например, в Sublime Text, тогда у вас появляется возможность не писать громоздкие куски кода на html и css, а записывать их в строчку, раскрывая затем в полноценную разметку. Например, строка *.block>ul.list>li.list__item*3*, может быть раскрыта в полноценный код:

```
<div class="block">
  <ul class="list">
    <li class="list__item"></li>
    <li class="list__item"></li>
    <li class="list__item"></li>
  </ul>
</div>
```

Затем вам просто необходимо вписать текстовые значения в элементы списка. То же касается и css: строка *w:100px+h:150px+bgc:#ff0* раскрывается в:

```
width: 100px;
height: 150px;
background-color: #ff0;
```

Благодаря данным средствам писать код становится быстрее и приятнее, так как вы меньше времени тратите на написания закрывающих блоков, переходам по строчкам, а так же уменьшается возможность, касательно html, не закрыть тот или иной элемент

разметки. В плане CSS, например, становится легче работать с браузерными префиксами, так как одна строчка с CSS3 свойством может быть раскрыта в несколько с добавлением необходимых префиксов.

Jade

Далее стоит упомянуть и о таком языке работе с шаблонами, как jade (http://jade-lang.com/). Опять-таки, с помощью jade писать html становится быстрее и удобнее, а так же появляется возможность создания переиспользуемых блоков кода и миксинов, передавая которым на вход те или иные параметры мы будет получать нужную нам разметку. Благодаря jade в проекте, касательно разметки и написания страниц, может поддерживаться модульность, а часто используемые названия и блоки кода могут выноситься в качестве переменных (названия, имена страниц и пр.) или миксинов в отдельные файлы настроек и файлы, соответственно. Для работы с jade, а точнее для преобразования написанного кода из jade в html, необходимо использовать командную строку, а так же установить на ваш компьютер платформу nodejs (https://nodejs.org/en/, http://nodejs.ru/).

Командная строка

Работа в командной строке так же может ускорить некоторые этапы работы над проектом, например, создание папок и файлов. Запустив консоль в директории проекта и набрав в ней всего лишь одну строку:

```
mkdir project && cd project && mkdir css && touch css/styles.css && mkdir images && mkdir js && touch js/app.js && touch index.html
```

После нажатия enter мы получим папку project в которой будут находиться папки css, images, js, в папках css и js файлы styles.css и app.js, соответственно, а в корне project лежать файл index.html. Таким образом мы сэкономили время на создание папки, написание её названия, создание файлов и пр. Так же командная строка используется для работы с остальными средствами автоматизации.

Sass

Далее стоит поговорить о таком языке как Sass (http://sass-lang.com/). Данный язык упрощает работу с css. Благодаря Sass можно записывать вложенные классы, проводить математические вычисления прямо в коде, выносить переиспользуемые величины в качестве переменных в отдельные файлы или в начало файла, что в значительной степени упрощает дальнейшую работу со стилями и упрощает изменение тех или иных величин, так как они все будут храниться в одном месте и не нужно будет «ходить» по всему проекту и вносить соответствующие правки. Так же, благодаря Sass в проекте можно придерживаться принципа модульности, храня части разметки в соответствующих файлах и подключать их все в основной, затем с помощью консольной команды можно собрать весь код в один css файл, так же поддерживается сборка результирующего файла на лету по мере написания стилей. Опять-таки для работы с Sass потребуется командная строка. Процесс установки пакета можно изучить здесь: http://sass-lang.com/install.

HTML5Boilerplate

HTML5Boilerplate (https://html5boilerplate.com/) представляет собой заранее скомпонованный проект, в котором уже есть index.html, normalize.css и пр. В index.html, например, уже заранее прописаны основные куски кода, которые необходимы при написании html страницы. Проект позволяет скачать архив со всеми файлами как есть или же использовать кастомную сборку: http://www.initializr.com/. Опять-таки, принцип прост: зачем тратить время на написание одних и тех же строк или на проделывание одних и тех же манипуляций, если мы сразу можем сосредоточиться непосредственно на работе с проектом.

Gulp u Grunt

Gulp (http://gulpjs.com/) и Grunt (http://gruntjs.com/) представляют собой сервисы запуска задач, которые описываются в основных файлах данных сервисов, выполняются с помощью совместимых модулей и запускаются на выполнение с помощью консоли.

Данные сервисы помогают использовать в разработке, например, вышеописанные средства Jade и Sass, автоматически преобразовывать их в соответствующие файлы html и css, сжимать их, если необходимо, а так же проверять, например css свойства, на необходимость использования браузерных префиксов и подставлять их в автоматическом режиме.

Оба сервиса используют различный подход в описании выполнения задач, но суть их работы одинакова. Полезным будет данное сравнение: http://frontender.info/gulp-grunt-whatever/

Bootrstrap, Foundation, Material Design Lite

Данные фрэймворки уже упоминались в разделе про дизайн. Но стоит опять вспомнить о них, так как они, в определённой степени, помогают автоматизировать процесс разметки веб-страницы и работы с ней.

При подключении файлов фрэймворков в проекте мы можем использовать определённые в них классы и сниппеты разметки, которые будут работать по уже описанным правилам и иметь установленный внешний вид (например, кнопки, поля ввода, таблицы и пр.). Таким образом мы экономим время, например, на описание расположения блоков дизайна, их размеров в зависимости от разрешения, внешний вид полей формы, кнопок и их состояния.

Со всеми возможностями и примерами можно ознакомиться на соответствующих сайтах:

- Bootstrap: http://getbootstrap.com/
- Foundation: http://foundation.zurb.com/
- Material Design Lite: http://www.getmdl.io/