Начнем с HTML

Когда делаете любой макет, сперва начните с крупных базовый частей. Если Вы откроете vk.com, то можно сразу выделить такую верстку:

```
<div class='top'></div>
<div class='cont'>
<div class='left'><div>
<div class='center'><div>
<div>
<div class='footer'><div>
```

Примечание. Если говорить о HTML5, то теги будут немного другими (семантическими) - но смысл остается тот же.

При этом не задумывайтесь пока, как это будет выглядеть. Ваша задача сейчас - это правильно и логично создать структуру HTML. Старайтесь называть классы CSS максимально понятно. Если Вы используйте где-то в компоненте классы, то лучше ставьте префиксы, например, для компонента FeedBack это можен быть fbCont или fb-cont. Желательно, придерживаться только одной выбранной нотации именования классов.

После того, как Вы определили структуру верхнего уровня, можно переходить к детализации блоков. Пример:

```
<div class='top'>
<div class='logo></div>
<div class='search'></div>
<div class='menu'></div>
</div>
```

И т.д.:

```
<div class='menu'>
  <ul>
        <li><a href='#'>Люди</a></li>
        <li><a href='#'>Сообщества</a></li>
        <li><a href='#'>Игры</a></li>
        <li><a href='#'>Музыка</a></li>
  </ul>
</div>
```

Таким образом в итоге Вы получаете довольно детальную схему HTML. После этого можно переходить к CSS.

Как видите, здесь мы обошлись одним тегом DIV.

Конечно, есть еще и другие теги: a, br, span, img, body, head, title, p, em, strong.

По сути это весь набор основных тегов. Изучите их применение на htmlbook.ru. Какие-то очень специфичные теги нет большого смысла изучать, т.к. довольно велика вероятность

того, что они не поддерживаются всеми браузерами, которые используются в настоящее время. Лучше Вы будете знать очень хорошо базис, чем знать все понемногу. Для 95% задач этого базиса будет вполне достаточно. Изучите все атрибуты этих тегов и примеры использования.

Задание 1. Сделайте блоковую структуру для своей страницы с детализацией блоков.

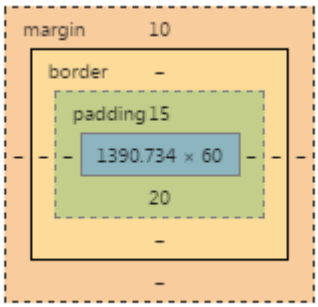
Переходим к CSS

Для большинства программистов это довольно больная тема. Я провожу довольно часто собеседования, и знания по CSS хромают практически у всех. На рынке труда очень много back end программистов. Но проблема в том, что для более-менее сложных приложений нужны front end программисты. Отдельный верстальщик - это хорошо, но при разработке JS-приложений программист очень хорошо должен представлять, что и как работает в CSS.

Для начала - как формируется в браузере страница.

Загружается HTML. Далее браузер разбирает, какие есть URL на странице (стили, скрипты, картинки) и загружает их. При обработке стилей браузер исходит из такого понятия как “поток”. Т.е. браузер последовательно отображает теги на странице в потоке. По умолчанию все теги находятся в потоке, именно по этому Ваша страница без CSS загружает последовательно все ваши DIV’ы.

Также есть такое понятие, как блочная модель. У блока могут быть ширина и высота (width, height), отступы (margin), поля (padding) и граница (border). Таким образом мы можем управлять расположением и внешними габаритами блока.



Основные селекторы CSS (как выбрать элементы):

- tag - выбор всех тегов tag
- .class - все элементы с классом class

- #id - элемент с идентификатором id
- input[type=checkbox] - все теги input с заданным атрибутом type
- :checked, :selected - выбрать отмеченные элементы
- a b - выбрать все теги b которые находятся в тегах a
- a>b - выбрать все теги b которые находятся непосредственно в тегах a
- a+b - выбрать все теги b которые идут сразу за тегами a

Это самый минимум селектором, которые надо наизусть знать. Помните, что jQuery библиотека имеет более широкий набор селекторов, т.е. не применяйте расширенные селекторы jquery в css.

Задание 2. Изучите псевдоселекторы CSS

Есть ряд важных свойств:

1. **position** - управляет позиционированием на странице. Возможные варианты:
a. *static* - значение по умолчанию
b. *absolute* - элемент извлекается из потока (т.е по сути не занимает места на странице) и позиционируется с помощью свойств left, top, right, bottom относительно родительского элемента с position: relative (если его нет, то окна браузера)
c. *relative* - элемент все еще в потоке (т.е. занимает в нем место), но его можно двигать с помощью свойств top, left, right, bottom. Также задает элемент как контейнер для внутренних элементов с установленным position:absolute
2. **float, clear** - обтекание элемента
a. Все элементы потока обтекают данный элемент либо справа, либо слева. Clear отменяет действие float
3. **display** - как себя будет вести элемент на странице (блок или нет). Основные значения:
a. *block* - элемент ведет себя как блок (занимает всю строку) и на него действует блочная модель (margin, padding, width, height)
b. *inline* - ведет себя как потоковый элемент, т.е. рендерится слева направо и сверху вниз (ведет себя как текст - слева направо и сверху вниз заполняет все возможное место)
c. *inline-block* - потоковый элемент, имеющий блочную модель (т.е. для него действуют margin,padding, width, height)

Задание 3. Крайне важно полностью понять, как работают эти свойства. Уделите этому достаточно времени и изучите нюансы использования всех этих свойств.

Есть также ряд других важных свойств, которые также следует изучить: width, height, background, font, z-index, border и т.д. Изучите примеры их использования на htmlbook.ru.

Пару слов о единицах:
px - пиксели
pt - размер шрифта (мы его не используем)
em - относительная единица. 1 em - это размер текущего шрифта.
% - задание метрики в процентах.

Если несколько стилей действуют на один и тот же элемент, то тут включается принцип каскадности: сначала применяются стили из внешних файлов, затем из тега <style></style> и затем из внутреннего атрибута style.

Таким образом можно переопределять стили.

Есть возможность жестко переопределить стиль с высшим приоритетом (!important):

```
a{ color:#ccc !important; }
```

Приоритет выбора стилей зависит от указания точности стилей.

Например, в следующем примере цвет ссылок в div'ах и span'ах будет красный:

```
div span>a {color:red;}
a{color: green;}
```

Правила определения спецификация довольно сложны. Их Вы можете найти в книге Эрика Мейера "CSS. Подробное руководство". Это одна из лучших книг по CSS.

Задание 4. Сделайте простейший макет (HTML+CSS) для любого выбранного сайта, например, http://www.microsoft.com/ru-ru/default.aspx

Также следует сказать несколько слов о CSS3. В основном это улучшения в виде скругленных углов, прозрачности, множественного фона и т.д. Базис весь заложен в CSS2.1 и его вполне достаточно для начала.

Задание 5. Найдите, что нового дает CSS3 по сравнению CSS2.1.

CSS фреймворк

Такие фреймворки нужны для облегчения жизни верстальщикам. Есть предопределенные классы, которые закладывают определенное поведение в блок. Поэтому

Вам даже нет необходимости писать CSS код, т.к. за Вас это уже сделали разработчики фреймворка - достаточно просто подставить определенный класс в разметку.

Я рекомендую Вам выбрать один любой фреймворк и работать только с ним, изучая его в глубину.

http://getbootstrap.com/css/ - css фреймворк, как часть Bootstrap. Позволяет делать довольно просто макеты страниц и имеет все необходимое типовое оформление элементов страниц (кнопки, формы, списки, меню и т.д.).

Задание 6. Изучить Bootstrap CSS и сделать тот же макет, что Вы делали ранее, но теперь только с использованием классов Bootstrap. Сравнить эти 2 подхода и сделать выводы.

Задание 7. Изучите что такое LESS. Есть содержательная статья на Хабре. http://habrahabr.ru/post/136525/

Еще одной замечательной особенностью Bootstrap является то, что он позволяет делать адаптивный дизайн - т.е. страница будет учитывать ширину экрана (это может быть планшет, настольный монитор или телефон). Соответственно, внешний вид страницы будет меняться в зависимости от типа оборудования, на котором выводится страница.

С CSS нужна практика. Сделайте 10 макетов для разных сайтов и Вы почувствуете что более-менее освоили верстку. Самое главное из теории, что Вам нужно знать - это ключевые CSS свойства, CSS-селекторы, общие принципы работы CSS (каскадность, рендеринг).

Очень важный момент: очень-очень часто вполне адекватным решением будет купить готовый дизайн и готовую верстку. Это гораздо дешевле и быстрее, чем делать дизайн через дизайнера, а потом еще его верстать (причем кроссбраузерно и адаптивно). Шаблоны есть бесплатные и платные. Всегда предлагайте своим клиентам вариант сделать дизайн на шаблоне. Он ничем не уступает уникальному дизайну по качеству. Вернее обычно даже наоборот, качество дизайна шаблона выше, единственный момент - он не является уникальным (что в общем-то не имеет большого смысла, если ваши прямые конкуренты не используют этот шаблон - что маловероятно). Много шаблонов можно найти здесь - templatemonster.com. За 15-150 долларов вы можете получить очень хороший дизайн + готовую верстку.

Задание 8. Изучите чит лист
http://www.smashingmagazine.com/2009/07/css-3-cheat-sheet-pdf/

В следующей главе Вы погрузитесь в основную тему веб-разработки - front end программирование.