http://www.nicothin.ru/page/10-priznakov-professionalnoj-verstki.html

10 признаков профессиональной html-верстки

Материал ориентирован на верстальщиков начального и среднего уровня, и на заказчиков, желающих выяснить технический уровень сдаваемых им сверстанных страниц.

Нередко встречаю в сети новые сайты, сверстанные по всем «правилам» конца прошлого века: табличная верстка, фоновые картинки, вставленные с помощью тега `<img>`, обильное использование тега `<font>`, размеров контентного шрифта в пикселях и пр. Уверен, что (за редким исключением) подобные архаизмы нужно забывать.

Итак, каковы же признаки профессиональной верстки?

DIV-ная структура документа

Когда-то было действительно удобно использовать таблицы для разметки HTML-документа – блочная разметка просто не позволяла реализовать некоторые макеты. Но с появлением каскадных таблиц стилей (CSS) времена табличной верстки прошли и сейчас верстать таблицами – признак ретроградности.

Главные минусы табличной верстки:

1. избыточность кода,
2. ужасная читаемость кода,
3. проблемы просмотра с экранов мобильных устройств,
4. жесткая очередность загрузки страницы

Таблицы включены в спецификацию HTML для представления табличных данных, структура документа должна создаваться с помощью блоков `<div>`. Исключения редки, в моей практике они не встречались.

Пример простейшей табличной верстки двухколоночного макета:

```
<table width="100%" border="0" cellspacing="0" cellpadding="0">
 <tr  colspan="2">
   <td class="header" colspan="2">Шапочка</td>
 </tr>
 <tr>
   <td class="content">Контент</td>
   <td class="sidebar">Сайдбар</td>
 </tr>
 <tr  colspan="2">
   <td class="footer" colspan="2">Подвал с крысами</td>
 </tr>
```

```
</table>
```

Пример того же макета, сверстанного блоками:

```
<div id="header">Шапочка</div>
<div id="wrapper">
 <div id="content">Контент</div>
 <div id="sidebar">Сайдбар</div>
</div>
<div id="footer">Подвал без крыс</div>
```

Хорошо видна разница в объеме кода и его читаемости.

В последнем случае, совершенно не важно – располагается сайдбар (навигация, облако тегов, реклама и пр.) справа или слева от контента, в любом случае, сначала загрузится содержимое контентного блока.

Семантически верная структура страниц

Это касается HTML-документа в общем (см. первичную блочную структуру ниже), но более всего актуально для контентной части. Аккуратное использование тегов заголовков, параграфов, списков важно как для адекватной индексации поисковыми системами, так и для удобства конечного пользователя – при отключенном CSS и на экранах мобильных устройств сайт будет выглядеть ясно и логично.

Минимум кода

Чем проще и короче написан HTML-код, тем он профессиональнее. (Сделаем поправку на функционал и настраиваемость этого кода.)

О сокращении объема HTML-кода можно написать отдельную статью, я ограничусь одним, наиболее показательным, примером.

Рассмотрим участок HTML:

```
<div id="category">
 <h1 class="category-header">Рубрики</h1>
  <p class="left">Это облако рубрик</p>
  <ul class="category-list">
    <li><a class="category-link" href="#">кукловодство</a></li>
    <li><a class="category-link" href="#">танкостроение</a></li>
    <li><a class="category-link" href="#">добыча нефти</a></li>
    </ul>
</div>
```

Полужирным шрифтом в примере выделены лишние элементы.

Классы для тегов `<h1>`, `<p>`, `<ul>` и `<a>` в этом блоке введены для придания соответствующим элементам какого-то своего оформления средствами CSS, однако это – лишний код, написанный верстальщиком, плохо знакомым с CSS.

Для приведенного примера, стиль оформления ссылок в списке может выглядеть так:

```
.category-link {свойства}
```

А если написать короткую версию HTML:

```
<div id="category">
 <h1>Рубрики</h1>
  <p>Это облако рубрик</p>
  <ul>
    <li><a href="#">кукловодство</a></li>
    <li><a href="#">танкостроение</a></li>
    <li><a href="#">добыча нефти</a></li>
  </ul>
</div>
```

то стиль будет выглядеть так:

```
#category li a {свойства}
```

Это может показаться мелочью, но подобный прием может на треть сократить объем HTML при крайне незначительном увеличении объема CSS, что отразится на скорости загрузки страницы самым позитивным образом.

Четкие названия структурных блоков

Некоторые считают это хорошим тоном, но я уверен, что логичные и аккуратные названия классов (идентификаторов) блоков, составляющих структуру документа – неотъемлемый признак профессиональной верстки.

Почти все сайты имеют строго выделенные части:

1. «Шапка» (логотип и пр.)
2. Контент (собственно, содержимое страницы)
3. Сайдбар (навигация, реклама и прочие второстепенные блоки)
4. «Подвал» (копирайты, счетчики, условия использования информации и пр.)

Распространенность подобных структур послужила поводом для введения в следующую (5-ю) версию HTML таких тегов, как `<header>`, `<footer>` и некоторых других.
А пока до использования 5-й версии HTML еще далеко, признаком профессиональной верстки считается структура, подобная следующей:

```
<div id="header">Шапочка</div>
<div id="wrapper">
  <div id="content">Контент</div>
  <div id="sidebar">Сайдбар</div>
</div>
<div id="footer">Подвал</div>
```

где **header** – «шапка», **footer** – «подвал», **wrapper** – служебный блок, необходимый для управления выравниванием и объединения контентной и параконтентных частей. (Двухколоночная структура взята для примера, как наиболее распространенная на данный момент.)

Комментарии после закрывающих блочных тегов

В структуре документа нередко встречаются участки вида:

```
</div>
</div>
</div>
```

Даже если есть отбивка (пробелами, табами), бывает крайне трудно разобраться, какой `<div>` где закрывается. А любая модификация CMS или правка верстки требует таких знаний.

Решение простое: если один `<div>` включает (или может потенциально включать) большой участок кода, после закрывающего тега пишется комментарий, включающий названия класса или идентификатора блока. Например:

```
</div>
</div> <!-- /sidebar -->
</div> <!-- /wrapper -->
```

Правильный сброс float-блоков

У блочной верстки множество плюсов, но есть и минусы. Самый главный из них связан с обтеканием элементов (блоков, имеющих свойства `float: right` или `float: left`). Если такие блоки сильно отличаются по высоте, то следующие за ними в потоке HTML-документа элементы начинают обтекать один из блоков.

Для «сброса обтекания» часто используется следующий принцип:

```
<div id="content"> <!- родительский блок->
[… ]
<div style="float: left">некое содержимое</div>
<div style="float: left">другое содержимое</div>
<div style="clear: both"></div> <!-- «сбрасывающий» блок -->
[...]
</div> <!-- /content -->
```

То есть вставляется дополнительный «сбрасывающий» блочный элемент, предотвращающий смещение последующих элементов (в примере выделен полужирным).

Этот подход имеет право на существование только в одном случае – когда невозможно указать ширину родительского блока. Если же ширину родительского блока указать можно (единицы измерения значения не имеют), то короче и правильнее опустить «сбрасывающий» блок, а для родительского блока задать следующие CSS-свойства:

```
#content {width: [ширина]; overflow: hidden}
```

Есть похожие техники, связанные с использованием псевдокласса `:after` или свойства `height: 0` – несколько менее удобные и красивые, на мой взгляд.

Фоновые картинки вместо тегов `<img>`

Это нужно для адекватного отображения страницы на экране мобильных устройств, при печати, в случаях, когда пользователь отключил загрузку изображений (скажем, для экономии трафика при использовании мобильного доступа в интернет).

Прописанные цвета фона

Бывают ситуации: дизайн предполагает общий белый фон и есть блок с фоновой картинкой темного цвета, а внутри этого блока есть ссылки, цвет которых – белый. В таком случае, пока картинка не загружена, ссылок не видно (белый на белом). А если пользователь отключил картинки, он вообще не увидит эти ссылки. Предположим, что сии ссылки – основная навигация по сайту, – пояснения нужны?

В описанном случае, для блочного элемента должна быть задана не только картинка фона, но и цвет, близкий к цвету картинки.

Использование спрайтов

Использование спрайтов при верстке описано в интернете не один десяток раз, однако имеет смысл дать ее краткое объяснение. Состоит данная техника в том, что используемые на сайте изображения размещаются в одном файле и, при использовании, позиционируются с помощью CSS.

К примеру, некая ссылка должна реагировать на наведение курсора сменой фоновой картинки. В этом случае, обе картинки (и та, которая видна сразу и та, которую видно только при наведении) помещаются в один файл с изображением – первая картинка вверху, вторая – внизу. В CSS ссылке прописывается «блочность» и фоновое изображение, выравненное по верхней стороне, а при наведении фоновое изображение выравнивается по нижней стороне. Соответственно, при наведении фоновая картинка меняется.

Эта техника позволяет сократить количество запросов к серверу, не использует JS (как традиционные rollover-кнопки), сокращает объем HTML-кода.

Есть и еще одна причина использовать эту технику: браузеры не загружают картинки, которые не показаны на странице сразу.

Отсутствие хаков

Это спорный пункт. Часто без хаков просто не обойтись, особенно для Internet Explorer 6 (божий дар хакера), но уверенность в том, что для IE просто нужно подключать отдельные CSS-файлы не оставляет вашего покорного слугу.

IE 5. x уже потерял актуальность (по данным Mail.ru, версии 5.5 и ниже – около 1 %/. по W3Counter – все еще лучше), а вот для 6-й и 7-й версии, частенько, приходится подключать отдельные CSS-файлы с исправлениями ошибок рендеринга в этих браузерах.

Автор считает наиболее правильным подключение стилей с указанием версии IE при помощи комментариев:

```
<!-[if lte IE 6]>
 <link rel="stylesheet" type="text/css" href="[адрес]/style_for_ie6.css"/>
<! [endif]->
<!-[if lte IE 7]>
 <link rel="stylesheet" type="text/css" href="[адрес]/style_for_ie7.css"/>
<! [endif]->
```

Полон ли приведенный список особенностей профессиональной верстки? Конечно, нет. Здесь я привел только самое, на мой взгляд, главное.