



Лекция №9

адаптивная верстка



ОТЗЫВЧИВЫЙ ДИЗАЙН

На заре веб-дизайна страницы создавались для экрана определённого размера. Если у пользователя был экран большего или меньшего размера чем ожидал дизайнер, то результат мог быть от нежелательных полос прокрутки, до слишком длинной строки и плохого использования пространства. Поскольку становились доступны много различных размеров экранов, появилась концепция *отзывчивого (адаптивного) веб-дизайна (responsive web design (RWD))* — набор методов, которые позволяют веб-страницам менять свой макет и внешний вид в соответствии с разной шириной экрана, разрешением и т.д. Это та самая, идея которая изменила подход к дизайну веба для множества устройств, и в этой статье мы поможем вам понять основные методы, которые вам необходимо знать, чтобы освоить его.



Исторические макеты сайтов

В какой-то момент истории при разработке веб-сайта у вас было два варианта:

- Вы могли создать *жидкий* сайт, который будет растягиваться чтобы заполнить окно браузера
- или сайт с *фиксированной шириной*, который будет иметь фиксированный размер в пикселях.

Эти два подхода, как правило, приводили к тому, что веб-сайт лучше всего выглядел на экране человека, создавшего сайт! Жидкий сайт приводил к раздавленному дизайну на маленьких экранах и не читаемо длинным строкам на больших.

Сайт с фиксированной шириной рисковал иметь горизонтальную полосу прокрутки на экранах меньших чем ширина сайта (как видно ниже) и много белого пространства на краях дизайна на больших экранах.




Когда мобильный веб стал становиться реальностью с первыми функциональными телефонами, компании желающие охватить мобильники начали создавать в основном специальные мобильные версии своих сайтов, с различными URL (часто что-то наподобие *m.example.com* или *example.mobi*). Это означало, что необходимо было разрабатывать и поддерживать в актуальном состоянии две отдельные версии сайта.

Кроме того, эти мобильные сайты часто предлагали очень урезанный вариант. Поскольку мобильные гаджеты стали мощнее и способными отображать целые веб-сайты, пользователей мобильных устройств раздражало, что они обнаруживали себя запертыми в мобильной версии сайта, неспособные получить доступ к информации, которая, как они знали, есть в полнофункциональной версии сайта.




Гибкий макет до отзывчивого дизайна

Было разработано несколько подходов чтобы попытаться разрешить недостатки построения веб-сайтов жидким методом или методом с фиксированной шириной. В 2004 году Камерон Адамс написал пост [Resolution dependent layout](#) , описывающий метод создания дизайна который мог бы адаптироваться к разным разрешениям экрана. Этот подход требовал, чтобы JavaScript узнавал разрешение экрана и загружал корректный CSS.

Зои Миккели Гилленвотер сыграла важную роль в своей работе описав и формализовав различные способы посредством которых могут быть созданы гибкие сайты, пытаюсь найти золотую середину между заполнением экрана или полностью фиксированным размером.



Отзывчивый дизайн

Термин адаптивный дизайн был [Придуман Итаном Маркоттом в 2010 году](#)  и описывал использование трёх методов в сочетании.



1. Первой была идея жидких сеток, нечто что уже исследовала Гилленвотер, что можно прочесть в статье Маркотта - [Fluid Grids](#)  (опубликовано в 2009 в A List Apart).
2. Вторым методом была идея [жидких изображений](#) . Используя очень простой метод настройки свойства `max-width` на `100%`, изображения будут становиться меньше если содержащий столбец становится уже чем изначальный размер изображения, но никогда не становится больше. Это позволяет изображению уменьшаться чтобы соответствовать столбцу гибких размеров, а не перекрываться с ним, но не расти и становиться пиксельным если столбец становится шире изображения.
3. Третьим ключевым компонентом были [медиавыражения](#). Медиавыражения позволяют переключать тип макета применяя только CSS то, что Камерон Адамс исследовал, используя JavaScript. Вместо того чтобы иметь один макет для всех размеров экранов, макет мог изменяться. Боковые панели можно перемещать для маленьких экранов, либо отображать альтернативную навигацию.



Очень важно понять, что **адаптивный веб-дизайн** — это не отдельная технология, это термин используемый, чтобы описать подход к веб-дизайну или набор лучших практик, используемых для создания макета, который может реагировать на используемое устройство для просмотра контента. В первоначальном исследовании Маркотта это означало гибкие сетки (с использованием floats) и медиавыражения, однако почти за 10 лет, прошедших с момента написания этой статьи, адаптивная работа стала стандартом по умолчанию. Современные методы макета CSS отзывчивы по своей сути, и у нас есть новые штучки, встроенные в веб-платформу для того, чтобы делать дизайн отзывчивых сайтов проще.



Регулировка разрешения экрана

В принципе, можно разбить устройства на разные категории и верстать для каждой из них отдельно, но это займет слишком много времени, да и кто знает, какие стандарты будут через пять лет? Тем более, согласно статистике мы имеем целый спектр разнообразных разрешений:



Portrait



Landscape



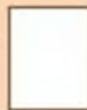
96
x
65



128
x
128



128
x
160



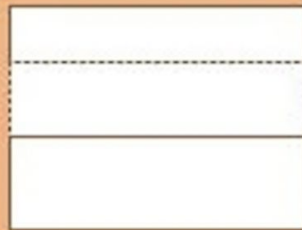
176
x
208/220



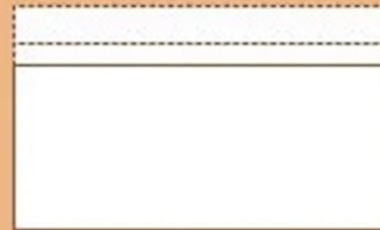
240
x
320



320
x
480



640
x
200/360/480



800
x
352/400/480

Resolution (ppi):

250

200

150

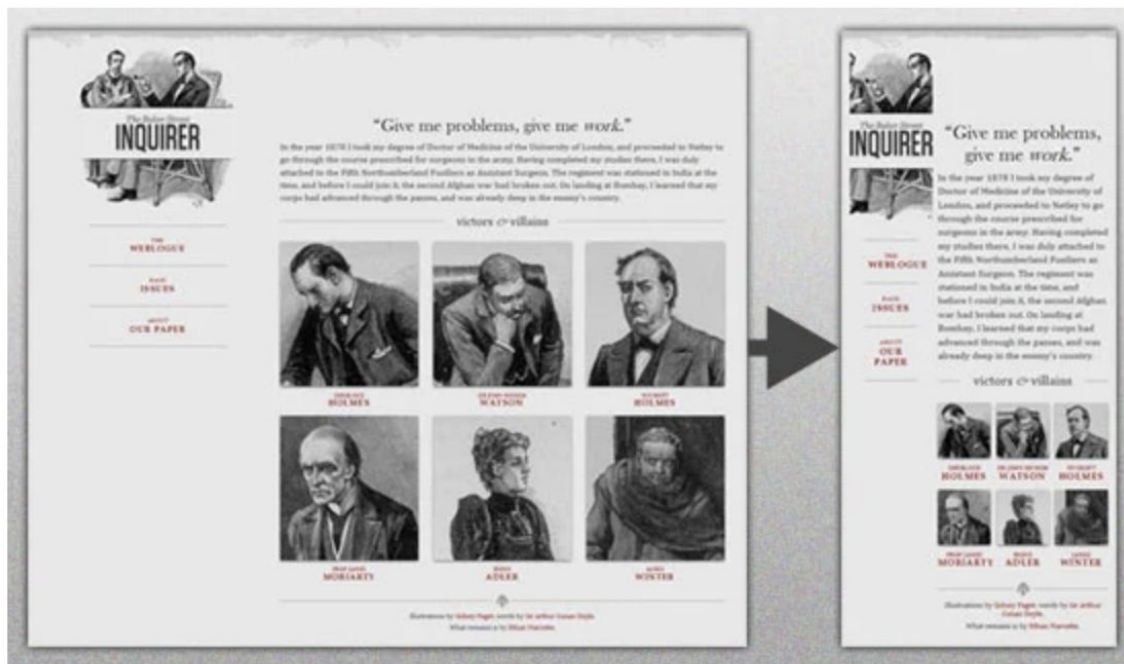
100



Частичное решение: делаем всё гибким

Конечно, это не идеальный способ, но он устраняет большую часть проблем с адаптивной вёрсткой.

Итан Маркотт (Ethan Marcotte) создал простой шаблон, демонстрирующий использование гибкой вёрстки:



Элемент `<picture>`

HTML-элемент `<picture>` служит контейнером для одного или более элементов `<source>` и одного элемента `` для обеспечения оптимальной версии изображения для различных размеров экрана. Браузер рассмотрит каждый из дочерних элементов `<source>` и выберет один, соответствующий лучшему совпадению; если совпадений среди элементов `<source>` найдено не будет, то будет выбран файл, указанный атрибутом `src` элемента ``. Затем выбранное изображение отображается в пространстве, занятом элементом ``.




Атрибут `media`

Атрибут `media` позволяет определить медиавыражение, которое веб-браузер будет анализировать для выбора элемента `<source>`. Если медиавыражение определяется как ложное (`false`), то элемент `<source>` пропускается.

```
<picture>
  <source srcset="mdn-logo-wide.png" media="(min-width: 600px)">
  
</picture>
```



Адаптивная вёрстка с помощью медиазапросов CSS3




Рассмотрим, как можно использовать CSS3-медиазапросы для создания адаптивного дизайна. `min-width` задает минимальную ширину окна браузера или экрана, к которой будут применены определенные стили. Если какое-нибудь значение будет ниже `min-width`, то стили будут проигнорированы. `max-width` делает противоположное.

Пример:

```
@media screen and (min-width: 600px) {  
  .hereIsMyClass {  
    width: 30%;  
    float: right;  
  }  
}
```






В то время как `min-width` и `max-width` могут быть применимы и к ширине экрана, и к ширине окна браузера, нам может понадобиться работать только с шириной устройства. Например, чтобы игнорировать браузеры, открытые в маленьком окне. Для этого можно использовать `min-device-width` и `max-device-width`:

```
@media screen and (max-device-width: 480px) {  
  .classForiPhoneDisplay {  
    font-size: 1.2em;  
  }  
}
```

```
@media screen and (min-device-width: 768px) {  
  .minimumiPadWidth {  
    clear: both;  
    margin-bottom: 2px solid #ccc;  
  }  
}
```

У медиазапросов есть свойство *orientation*, значениями которого могут быть либо *landscape* (горизонтальный), либо *portrait* (вертикальный):



```
@media screen and (orientation: landscape) {  
  .iPadLandscape {  
    width: 30%;  
    float: right;  
  }  
}
```

```
@media screen and (orientation: portrait) {  
  .iPadPortrait {  
    clear: both;  
  }  
}
```



Также значения медиазапросов можно комбинировать:

```
— @media screen and (min-width: 800px) and (max-width: 1200px) {  
  .classForaMediumScreen {  
    background: #cc0000;  
    width: 30%;  
    float: right;  
  }  
}
```

Этот код будет выполнен только для экранов или окон браузеров шириной от 800 до 1200 px.



Загрузить определенный лист со стилями для разных значений медиазапросов можно так:

```
<link rel="stylesheet" media="screen and (max-width: 600px)"  
href="small.css"/>
```

```
<link rel="stylesheet" media="screen and (min-width: 600px)"  
href="large.css"/>
```

```
<link rel="stylesheet" media="print" href="print.css"/>
```






Опциональное отображение контента

Возможность сжимать и менять местами элементы, чтобы они уместились на маленьких экранах, — это замечательно. Но это не лучший вариант. Для мобильных устройств обычно используется более широкий набор изменений: упрощенная навигация, более сфокусированный контент, списки или строки вместо колонок.



Относительные значения

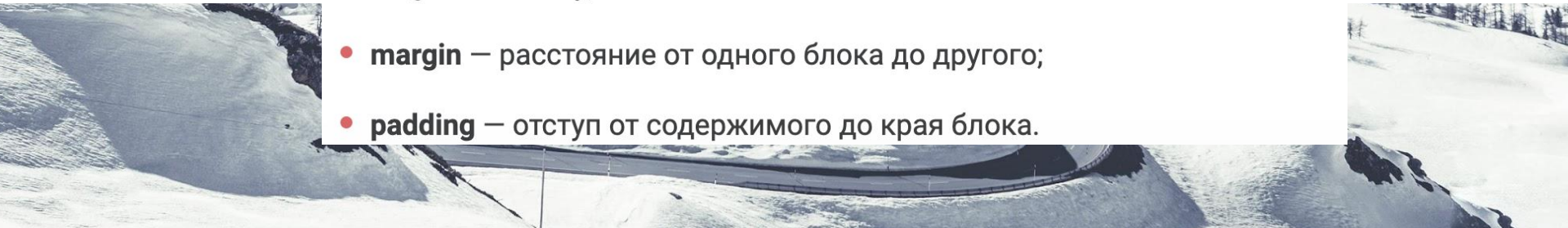


Относительные значения или единицы определяют **размер шрифта и отступов** не в сантиметрах и дюймах, а по отношению к условной величине, которую по умолчанию задает браузер.

Для размеров и отступов

С их помощью можно определить:

- **width** — ширину;
- **height** — высоту;
- **margin** — расстояние от одного блока до другого;
- **padding** — отступ от содержимого до края блока.



- **Размеры и отступы удобнее всего определять в процентах.** Например, можно определить параметр `width`, как 80% или 90%. Страница будет отображаться с этой шириной по отношению к исходному родительскому компоненту:

Можно воспользоваться единицами `vw` и `vh`. Обе они обозначают, по сути, то же самое, что и проценты: в первом случае — 1% ширины, а во втором — 1/100 долю высоты экрана.





Спасибо!

