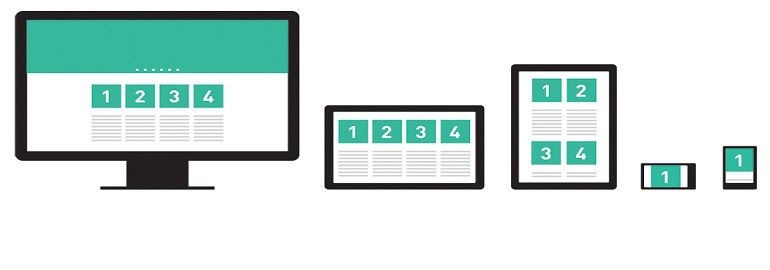
**Адаптивная вёрстка: что это и как использовать**



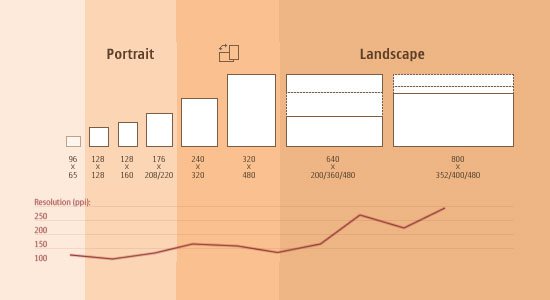
В наши дни практически каждый заказчик хочет получить мобильную версию для своего сайта. И это естественно: нужен один дизайн для iPhone, другой для iPad — и он также должен подходить под все другие размеры экранов.

В теме веб-дизайна и разработки мы быстро приближаемся к точке, когда мы будем не способны справляться с бесконечными новыми разрешениями и экранами. Для многих вебсайтов создание отдельной версии для каждого нового разрешения будет невозможным или, по крайней мере, не практичным. Должны ли мы смириться с потерей посетителей, для которых у нас нет верстки? Или можно что-то сделать?

**Адаптивная верстка** — подход, предполагающий изменение дизайна в зависимости от поведения пользователя, размера экрана, платформы и ориентации девайса. Другими словами, страница должна автоматически подстраиваться под разрешение, изменять размер картинок и т.д.  Это позволит устранить нужду в разработке дизайна для каждого нового устройства, появляющегося в продаже.

**Регулировка разрешения экрана**

Можно разбить устройства на разные категории и верстать для каждой из них отдельно, но это займет слишком много времени, и кто знает, какие стандарты будут через пять лет? Тем более, согласно статистике, основанной на 400 проданных с 2005 по 2008 год устройствах (а сейчас 2016), мы имеем целых спектр разнообразных устройств:



**Частичное решение: делаем всё гибким**

Конечно, это не идеальное решение, но оно решает большую часть проблем.

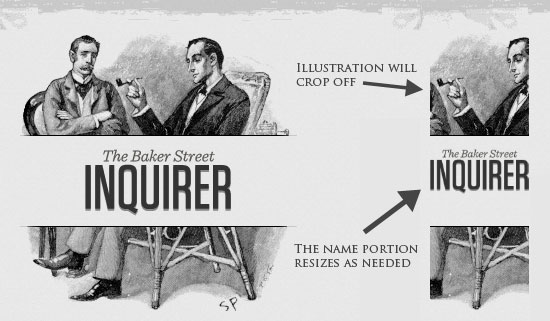
В своей статье ИтанМаркотт (EthanMarcotte) создал [простой шаблон](http://alistapart.com/d/responsive-web-design/ex/ex-site-flexible.html), демонстрирующий использование гибкой вёрстки:



Весь дизайн — микс[адаптивных слоев](http://alistapart.com/article/fluidgrids), [картинок](http://unstoppablerobotninja.com/entry/fluid-images) и в некоторых местах умной разметки. Создание адаптивных слоев — частая практика, что нельзя сказать об адаптивных картинках. Поэтому ниже представлены техники для реализации гибких картинок:

* [Hiding and Revealing Portions of Images](http://zomigi.com/blog/hiding-and-revealing-portions-of-images/);
* [CreatingSlidingCompositeImages](http://zomigi.com/blog/creating-sliding-composite-images/);
* [Foreground Images That Scale With the Layout](http://zomigi.com/blog/foreground-images-that-scale-with-the-layout/).

С первого взгляда может показаться, что все легко, но это не так. Взгляните на логотип:



Если сделать разрешение слишком маленьким, может показаться, что четкость картинки упала, но это необходимо для того, чтобы сохранить пропорции названия. Поэтому картинка поделена на две части: первая часть (иллюстрация) используется как фон, вторая (логотип) изменяет свои размеры пропорционально.

<h1 id="logo">

  <a href="#"><imgsrc="site/logo.png" alt="The Baker Street Inquirer"/></a>

</h1>

Элемент h1 содержит изображение в качестве фона, и картинка выровнена согласно фону контейнера (заголовка).

**Гибкие изображения**

Одна из самых главных проблем, которая нуждается в решении при работе с адаптивным дизайном — это работа с картинками. Существует много способов изменять размер изображений, и большинство из них реализуется очень просто. Одним из таких примеров является использование max-width в CSS:

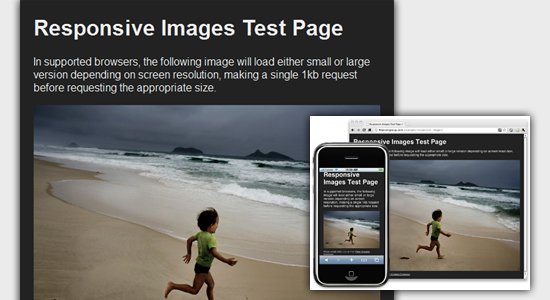
img {max-width: 100%;}

**Максимальная ширина**изображения равняется 100% от ширины экрана или окна браузера, и, когда ширина уменьшается, то же делает и картинка. Заметьте, что max-width**не поддерживается в IE**, поэтому используйте width: 100%.

Представленный способ является отличным началом в создании адаптивных изображений, но разрешение и время загрузки должны оставаться приоритетными.

**Ещё один способ: отзывчивые изображения**

Техника, представленная FilamentGroup, не только изменяет размер изображений в соответствии с пропорциями, но и сжимает разрешение картинок на маленьких экранах, чтобы на больших они не занимали дополнительного места.



Для использования данной техники требуется несколько файлов, все они доступны на [Github](https://github.com/filamentgroup/Responsive-Images). Сначала берём JavaScript-файл (*rwd-images.js*), файл *.htaccess* и *rwd.gif* (файл изображения). Потом используем немного HTML, чтобы связать большие и маленькие разрешения: сначала маленькое изображение с префиксом *.r* (для обозначения того, что картинка должна быть адаптивной), потом отсылка к большему изображению, используя data-fullsrc:

<imgsrc="smallRes.jpg" data-fullsrc="largeRes.jpg">

Для любого экрана шире 480 пикселей загрузится изображение с большим разрешением (*largeRes.jpg*); маленьким экранам не понадобится загружать большое изображение, загрузится маленькое (*smallRes.jpg*).

**Интересная возможность для iPhone**

Есть одна интересная вещь, касающаяся iPhone и iPod. Дизайн, созданный для больших экранов, просто сожмется для маленького браузера, без необходимости добавления скролла или дополнительной мобильной верстки. Однако изображений и текста не будет видно:



Для решения данной проблемы воспользуемся тегом meta:

<metaname="viewport" content="width=device-width; initial-scale=1.0">

Если initial-scaleравно единице, ширина картинок становится равной ширине экрана.

**Настраиваемая структура макета страницы**

Для чрезвычайных изменений размеров мы можем захотеть изменить расположение элементов в целом через отдельный файл со стилями или, что более эффективно, через CSS-медиазапрос. Это не должно вызвать особо много проблем, т.к. большинство стилей останутся прежними, но некоторые изменятся.

Например, у нас есть главный файл со стилями, который задает #wrapper, #content, #sidebar, #nav вместе с цветами, фоном и шрифтами. Если наши главные стили делают макет слишком узким, коротким, широким или высоким, мы можем это определить и подключить новые стили.

*style.css (основной):*

/\* Default styles that will carry to the child style sheet \*/

html,body{

   background...

   font...

   color...

}

h1,h2,h3{}

p, blockquote, pre, code, ol, ul{}

/\* Structural elements \*/

#wrapper{

  width: 80%;

  margin: 0 auto;

  background: #fff;

  padding: 20px;

}

#content{

  width: 54%;

  float: left;

  margin-right: 3%;

}

#sidebar-left{

  width: 20%;

  float: left;

  margin-right: 3%;

}

#sidebar-right{

  width: 20%;

  float: left;

}

*mobile.css (дочерний):*

#wrapper{

  width: 90%;

}

#content{

  width: 100%;

}

#sidebar-left{

  width: 100%;

  clear: both;

  /\* Additional styling for our new layout \*/

  border-top: 1px solid #ccc;

  margin-top: 20px;

}

#sidebar-right{

  width: 100%;

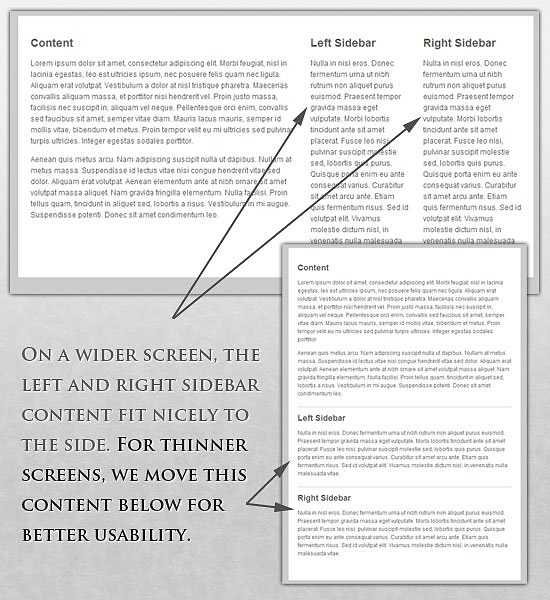
  clear: both;

  /\* Additional styling for our new layout \*/

  border-top: 1px solid #ccc;

  margin-top: 20px;

}



**Медиазапросы CSS3**

Давайте посмотрим, как можно использовать CSS3-медиазапросы для создания адаптивного дизайна. min-widthзадает 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-widthделает противоположное.

Пример:

@media screen and (min-width: 600px) {

  .hereIsMyClass{

    width: 30%;

    float:right;

  }

}

Медиазапрос заработает только в том случае, когда min-widthбудет больше или равна 600 px.

@media screen and (max-width: 600px) {

  .aClassforSmallScreens{

    clear: both;

    font-size: 1.3em;

  }

}

В этом случае класс (*aClassforSmallscreens*) сработает, когда ширина экрана будет меньше или равна 600 px.

В то время как min-widthи max-widthмогут быть применимы к экранам и окнам браузеров, нам может понадобиться работать только с шириной устройства. Для этого можно использовать min-device-widthи 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;

  }

}

Специально для iPad у медиазапросов есть свойство *orientation*, значениями которого могут быть либо *landscape*(горизонтальный), либо *potrait*(вертикальный):

@media screen and (orientation: landscape) {

  .iPadLandscape{

    width: 30%;

    float: right;

  }

}

@media screen and (orientation: portrait) {

  .iPadPortrait{

    clear: both;

  }

}

Также значения медиазапросов можно комбинировать, например:

@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 между 800 и 1200 px.

Кто-то может захотеть загрузить определенный лист со стилями для разных значений медиазапросов, это можно сделать вот 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"/>

**JavaScript**

Если ваш браузер не поддерживает CSS3-медиазапросы, то замену стилей можно организовать с помощью jQuery следующим образом:

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>

<script type="text/javascript">

  $(document).ready(function(){

    $(window).bind("resize", resizeWindow);

    functionresizeWindow(e){

      varnewWindowWidth= $(window).width();

      // If width width is below 600px, switch to the mobile stylesheet

      if(newWindowWidth< 600){

        $("link[rel=stylesheet]").attr({href: "mobile.css"});

      } else if(newWindowWidth> 600){

        // Else if width is above 600px, switch to the large stylesheet

        $("link[rel=stylesheet]").attr({href: "style.css"});

      }

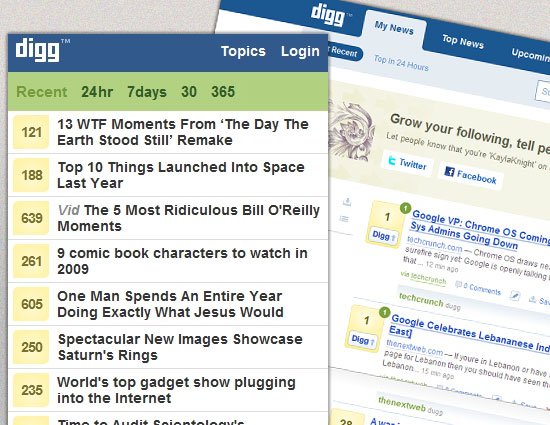
    }

  });

</script>

**Опциональное отображение контента**

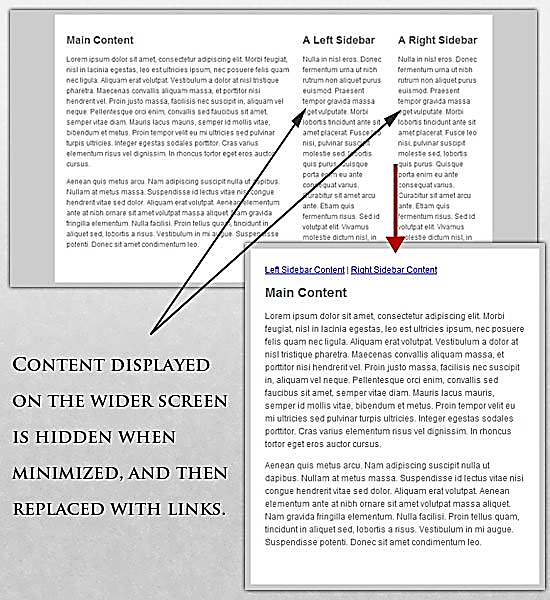
Возможность сжимать и менять местами элементы, так ,чтобы они все уместились на маленьких экранах, это замечательно. Но это не лучший вариант. Для мобильных устройств обычно используются: упрощенная навигация, более сфокусированный контент, списки или строки вместо колонок.



К счастью, CSS дает нам возможность показывать и прятать контент с неимоверной легкостью!

display: none;

display: none используется для объектов, которые нужно спрятать.



Вот наша разметка:

<p class="sidebar-nav"><a href="#">Left Sidebar Content</a> | <a href="#">Right Sidebar Content</a></p>

<div id="content">

  <h2>Main Content</h2>

</div>

<div id="sidebar-left">

  <h2>A Left Sidebar</h2>

</div>

<div id="sidebar-right">

  <h2>A Right Sidebar</h2>

</div>

В главном файле стилей мы меняем ссылки на колонки, т.к. у нас достаточно большой экран, чтобы отобразить весь контент.

*style.css (основной):*

#content{

  width: 54%;

  float: left;

  margin-right: 3%;

}

#sidebar-left{

  width: 20%;

  float: left;

  margin-right: 3%;

}

#sidebar-right{

  width: 20%;

  float: left;

}

.sidebar-nav{

  display: none;

}

Теперь прячем колонки и показываем ссылки:

*mobile.css (упрощенный):*

#content{

  width: 100%;

}

#sidebar-left{

  display: none;

}

#sidebar-right{

  display: none;

}

.sidebar-nav{

  display: inline;

}

С возможностями прятать и показывать элементы, изменять размеры картинок, элементов и многое другое, дизайн смог приспосабливаться к различным устройствам и экранам. Если размер экрана уменьшается, можно, например, использовать скрипт или альтернативный файл со стилями, чтобы увеличить белое пространство, или заменить навигацию для большего удобства.

https://tproger.ru/translations/responsive-web-design-tips/

Переводстатьи[«Responsive Web Design: What It Is And How To Use It»](https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/" \t "_blank)[НикитаМингалеев](https://tproger.ru/author/nmingaleev), юныйпадаван