Видео недели:

[Адаптивная верстка](https://www.youtube.com/watch?v=1NRLw9hw03I)

**Адаптивная верстка**

**Адаптивная вёрстка** сайта позволяет веб-страницам автоматически подстраиваться под экраны планшетов и смартфонов. Мобильный интернет-трафик растёт с каждым годом и чтобы эффективно обрабатывать этот трафик, нужно предлагать пользователям адаптивные сайты с удобным интерфейсом.

Поисковые системы используют ряд критериев для оценки адаптивности сайта при просмотре на мобильных устройствах. Google старается упростить пользование Интернетом для владельцев смартфонов и планшетов, отмечая в мобильной выдаче адаптированные под мобильные устройства сайты специальной пометкой **mobile-friendly**. В Яндексе также работает алгоритм, который отдает предпочтение сайтам с мобильной/адаптивной версией для пользователей в мобильном поиске.

Существует два основных подхода для создания сайтов, легко адаптирующихся для разных типов устройств:

* **Responsive Design (RWD)** — отзывчивый дизайн — проектирование сайта с определенными значениями свойств, например, гибкая сетка макета, которые позволяют одному макету работать на разных устройствах. Главная особенность отзывчивого веб-дизайна — за счёт подвижной сетки макет автоматически реагирует на изменение размеров экрана, раздуваясь или сужаясь, как воздушный шар.

Гибкость макета базируется на использовании относительных единиц измерения вместо фиксированных пиксельных значений, что позволяет регулировать ширину в соответствии с доступным пространством.

Гибкость **текстового содержимого** достигается путем вычисления размеров шрифта относительно размера шрифта в браузерах по умолчанию 16px, например для фиксированного размера font-size: 42px относительный размер равен 42px / 16px = 2.625em. Также размер текста может быть установлен с помощью vw единицы. Таким образом, размер текста будет соответствовать размеру окна браузера.

Проблема **гибких изображений** решается с помощью правила img {width: 100%; max-width: 100%;} для всех картинок на сайте. Это правило гарантирует, что изображения никогда не будут шире, чем их контейнеры и никогда не превысят своих истинных размеров на больших экранах.

Масштабируемая **векторная графика** — используйте svg-изображения, которые масштабируются до любого размера без потери качества и хорошо выглядят на дисплеях Retina.

Пример гибкой верстки:

* **Adaptive Design (AWD)** — адаптивный дизайн, или динамический показ — проектирование сайта с условиями, которые изменяются в зависимости от устройства, базируясь на нескольких макетах фиксированной ширины.

Адаптивная вёрстка предполагает отсутствие горизонтальной полосы прокрутки и масштабируемых областей при просмотре на любом устройстве, читабельный текст и большие области для кликабельных элементов.

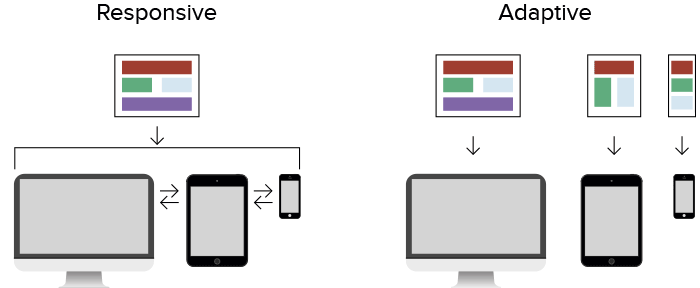
В отличие от отзывчивого дизайна, **адаптивный дизайн** (Adaptive Web Design) ориентируется на размеры устройств. Он использует несколько статичных макетов для разных типов устройств (мобильные устройства, планшеты, настольные компьютеры), базируясь на контрольных (переломных) точках. То есть макеты загружаются при определенных размерах окна браузера устройства, а переходы между макетами происходят скачкообразно, а не плавно.

Обычно адаптивные макеты имеют шесть вариантов макетов в зависимости от ширины экрана:

* + 320
  + 480
  + 760
  + 960
  + 1200
  + 1600

В адаптивных макетах главную роль играет функциональность, то есть при создании дизайна учитываются особенности устройств, например, сенсорное управление для мобильных устройств или большие пространства для настольных мониторов.

**В чём разница между отзывчивым и адаптивным веб-дизайном**



Поэтому основное отличие между этими приёмами — отзывчивый дизайн — один макет для всех устройств, адаптивный дизайн — один макет для каждого вида устройства.

**Гибкие изображения**

Работа с картинками — одна из самых главных проблем при работе с адаптивным дизайном. Существует много способов для изменения размера изображений, и большинство из них довольно просто реализовать. Одно из решений — использование max-width в CSS:

img {max-width: 100%;}

Если свойство max-width установлено на 100%, изображение будет уменьшаться, если это необходимо, но никогда не будет увеличиваться до размера, превышающего его исходный размер.

Максимальная ширина изображения равняется 100% от ширины экрана или окна браузера, поэтому чем меньше ширина, тем меньше картинка. Обратите внимание, что max-width не поддерживается в IE, поэтому там используйте width: 100%.

Представленный способ является неплохим вариантом создания адаптивных изображений, но, изменив только размер, мы оставим вес картинки прежним, из-за чего время загрузки на мобильных устройствах увеличится.

**<picture>**

HTML элемент <picture> позволяет определять разные изображения для разных размеров окна браузера.

<picture>

<source srcset="img\_smallflower.jpg" media="(max-width: 600px)">

<source srcset="img\_flowers.jpg" media="(max-width: 1500px)">

<source srcset="flowers.jpg"> <img src="img\_smallflower.jpg" alt="Flowers">

</picture>

**Медиазапросы**

Медиа запросы (media queries) - это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, это конструкции, которые позволяют определять на основании некоторых условий какие стили необходимо использовать на веб-странице, а какие нет.

В общем случае медиа-запрос состоит из ключевого слова, описывающего тип устройства (необязательный параметр) и выражения, проверяющего характеристики данного устройства. Из всех характеристик чаще всего проверяется ширина устройства width. Медиа-запрос является логическим выражением, которое возвращает истину или ложь.

Медиа-запросы могут быть добавлены следующими способами:

1. С помощью HTML:

<link rel="stylesheet" media="screen and (color)" href="example.css">

1. С помощью правила @import внутри элемента <style> или внешней таблицы стилей:

@import url(color.css) screen and (color);

1. Непосредственно в коде страницы:

<style>

@media (max-width: 600px) {

#sidebar {display: none;}

}

</style>

1. Внутри таблицы стилей style.css:

@media (max-width: 600px) {

#sidebar {display: none;}

}

**Опасненько!** Таблица стилей, прикрепленная через тег <link>, будет загружаться вместе с документом, даже если её медиа-запрос вернет ложь.

Пример, как может меняться дизайн страницы в зависимости от размера:

Медиа запросы появились в спецификации CSS3 и на сегодняшний день поддерживаются во всех современных браузерах (Chrome 4+, Firefox 3.5+, IE 9+, Opera 9+, Safari 4+).

Для поддержки медиа-запросов в старых браузерах можно воспользоваться JavaScript-библиотекой css3-mediaqueries.js, доступной по адресу <https://code.google.com/archive/p/css3-mediaqueries-js/>

**Тип носителя**

Тип носителя представляет собой тип устройства, например, принтеры, экраны.

* all Подходит для всех типов устройств.
* print Предназначен для страничных материалов и документов, просматриваемых на экране в режиме предварительного просмотра печати.
* screen Предназначен в первую очередь для экранов цветных компьютерных мониторов.
* speech Предназначен для синтезаторов речи.

Раньше определяли еще несколько дополнительных типов, таких как aural, braille, embossed, projection, tty, tv и handheld, но сегодня они считаются устаревшими.

Пример использования screen и print

**Медиа функции**

Для составления условия в @media можно использовать следующие фукнции:

* width - указывает требования к ширине области просмотра устройства (браузера).
* /\* применить стили CSS только для устройств с шириной области просмотра, равной 320px \*/
* @media (width: 320px) { /\* Стили CSS ... \*/ }
* min-width - задаёт минимальную ширину области viewport в px, em или других единицах.
* /\* для устройств (браузеров), которые предоставляют для страницы минимальную ширину области просмотра, равную 544 пикселя \*/
* @media (min-width: 544px) { /\* Стили CSS ... \*/ }
* max-width - указывает на то, какой должна быть максимальная рабочая область устройства (браузера).
* /\* стили, которые будут применены к элементам страницы с рабочей областью не больше 1199 пикселей \*/
* @media (max-width: 1199px) { /\* Стили CSS ... \*/ }
* height, min-height и max-height - задают требования аналогично вышеприведённым функциям, но в отношении высоты viewport.
* /\* стили, которые будут применены к элементам страницы в том случае, если viewport браузера будет больше 720px в высоту \*/
* @media (min-height: 720px) { /\* Стили CSS ... \*/ }
* orientation - функция, которая проверяет то, в каком режиме (portrait или landscape) отображается страница.

Пример, в котором в зависимости от ориентации экрана, отображается одна или другая картинка:

/\* landscape (альбомный) - это режим, в котором наоборот ширина viewport больше её высоты \*/

@media (orientation: landscape) {

#background-image { background: url(image1.png) no-repeat; }

}

/\* portrait (портретный) - это режим, в котором высота viewport больше ширины \*/

@media (orientation: portrait) {

#background-image { background: url(image2.png) no-repeat; }

}

* aspect-ratio (min-aspect-ratio, max-aspect-ratio) - позволяют указать то, как ширина устройства должна относиться к высоте. В качестве значений допускается использовать только целые значения.
* /\* для дисплеев с соотношением сторон 16/9 \*/
* @media screen and (device-aspect-ratio: 16/9) { /\* Стили CSS ... \*/ }
* /\* для дисплеев с соотношением сторон 1336/768 \*/
* @media screen and (device-aspect-ratio: 1336/768) { /\* Стили CSS ... \*/ }
* resolution (min-resolution, max-resolution) - указывает разрешение (плотность пикселей) устройства вывода. В качестве единиц измерения разрешения используются следующие величины: dpi (количество точек на дюйм), dpcm (количество точек на сантиметр), dppx (количество точек на пиксель).
* /\* для экранов, имеющих высокую плотность пикселей
* (т.е. для таких, у которых отношение аппаратных пикселей к CSS не менее 2) \*/
* @media screen and (min-resolution: 2dppx) { /\* Стили CSS ... \*/ }
* /\* при печати с разрешением свыше 300 точек на дюйм \*/
* @media print and (min-resolution: 300dpi) { /\* Стили CSS ... \*/ }

**Логические операторы**

С помощью логических операторов можно создавать комбинированные медиазапросы, в которых будет проверяться соответствие нескольким условиям.

**Оператор and**

Оператор and (И) связывает друг с другом разные условия. В случае его использования ОБА условия должны обязательно выполняться.

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

/\* CSS-стили \*/;

}

Стили этого запроса будут применяться только для экранных устройств с шириной области просмотра не более 600px.

@media (min-width: 600px) and (max-width: 800px) {

/\* CSS-стили \*/;

}

Стили этого запроса будут применяться для всех устройств при ширине области просмотра от 600px до 800px включительно.

Хороший пример:

**Оператор запятая**

Оператор запятая работает по аналогии с логическим оператором or (ИЛИ). В случае его использования достаточно совпадения любого из условий.

@media screen, projection {

/\* CSS-стили \*/;

}

В данном случае CSS-стили, заключенные в фигурные скобки, сработают только для экранных или проекционных устройств.

**Оператор not**

Оператор not (НЕ) позволяет сработать медиазапросу в противоположном случае. Ключевое слово not добавляется в начало медиазапроса и применяется ко всему запросу целиком, т.е. запрос

@media not all and (monochrome) {...}

будет эквивалентен запросу

@media not (all and (monochrome)) {...}

Если медиазапрос составлен с использованием оператора запятая, то отрицание будет распространяться только на ту часть, которая идет до запятой, т.е. запрос

@media not screen and (color), print and (color)

будет эквивалентен запросу

@media (not (screen and (color))), print and (color)

**Метатег viewport**

Чтобы мобильные браузеры для операционных систем Android и iOS не изменяли автоматически размер страниц сайта, используется специальный тег <meta> с атрибутом name="viewport". В данном теге допускается устанавливать определенное значение для параметров width и initial-scale:

<meta name="viewport" content="width=device-width, initial-scale=1">

Здесь initial-scale=1 говорит о том, что размер страницы в браузере будет равен 100% величины области просмотра. Это означает, что 1 пиксель окна просмотра будет = 1 пиксель CSS.

width=device-width говорит о том, что ширина страницы будет равна 100% ширины окна любого браузера. То есть, ширина страницы сайта соответствует ширине устройства, поэтому её не надо масштабировать.

Как видно из картинки, очень важно использовать эти настройки viewport. Хорошая новость в том, что при использовании генерации страницы через *!+Tab* в Emmet этот метатег добавляется автоматически.

Пример:

<meta name="viewport" content="initial-scale=2.0, width=device-width">

Здесь содержимое в окне браузера будет в 2 раза больше по сравнению с физическим размером.

Также тег <meta> можно использовать для контроля над тем, насколько пользователи могут увеличивать и уменьшать ширину страницы:

<meta name="viewport" content="width=device-width, maximum-scale=3, minimum-scale=0.5">

Данный код позволит увеличивать ширину страницы до величины, равной 3-х кратной ширине экрана устройства и уменьшать её до половины ширины экрана устройства.

Лишить пользователей возможности масштабирования можно с помощью атрибута user-scalable:

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">



Из этой картинки наглядно видно, что является шириной экрана, а что - шириной страницы.

**Стратегии использования медиа-запросов**

Для создания дизайна, позволяющего лучшим образом отображать сайт на различных устройствах, используют общие стратегии медиа-запросов:

1. Уменьшение количества колонок (столбцов) и постепенная отмена обтекания для мобильных устройств.
2. Использование свойства max-width вместо width при задании ширины блока-контейнера.
3. Уменьшение полей и отступов на мобильных устройствах (например, нижних отступов между заголовком и текстом, левого отступа для списков и т.п.).
4. Уменьшение размеров шрифтов для мобильных устройств.
5. Превращение линейных навигационных меню в раскрывающиеся.
6. Скрытие второстепенного содержимого на мобильных устройствах с помощью display: none.
7. Подключение фоновых изображений уменьшенных размеров.

**Универсальные шаблоны**

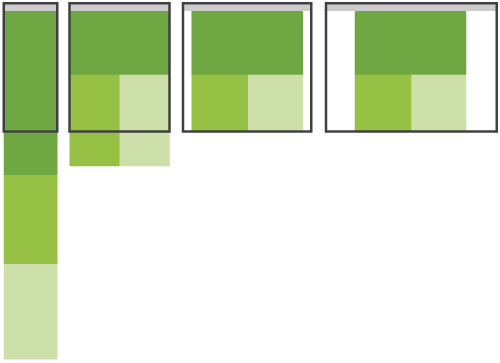
Большая часть макетов, используемых для создания отзывчивого веб-дизайна, относятся к одной из пяти категорий шаблонов, определенных Люком Вроблевски:

* **Mostly Fluid** (Наиболее резиновый)
* **Column Drop** (Столбцы друг под другом)
* **Layout Shifter** (Двигающийся макет)
* **Tiny Tweaks** (Крошечные изменения)
* **Off Canvas** (Вне экрана)

В некоторых случаях на странице может использоваться сочетание шаблонов, например Column Drop и Off Canvas.

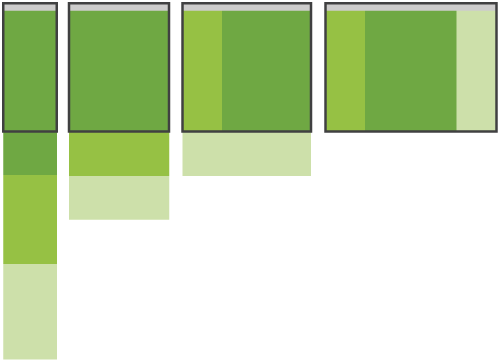
**Mostly Fluid**

Популярный макет, состоит главным образом из резиновой сетки. На экранах большой или средней ширины ее размер обычно остается неизменным, на больших экранах регулируются только поля. На небольших экранах резиновая сетка вызывает перерасчет макета для основного контента, а столбцы размещаются друг под другом. Достоинством шаблона является то, что в нем необходима только одна контрольная точка между небольшими экранами и экранами большого размера.



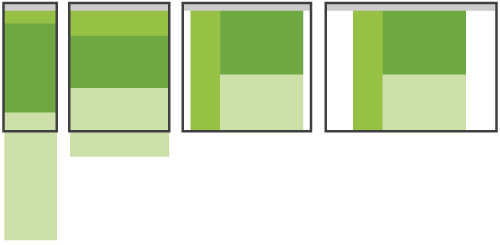
**Column Drop**

Столбцы размещаются один за одним по вертикали, если ширина окна не может отобразить весь контент. В результате столбцы будут располагаться вертикально друг под другом. Выбор контрольных точек для этого шаблона макета зависит от контента и определяется для каждого варианта дизайна отдельно.



**Layout Shifter**

Наиболее отзывчивый шаблон, так как в нем предусмотрено наличие нескольких контрольных точек для экранов различной ширины. Основным отличием этого макета является то, что вместо перерасчета дерева отрисовки и размещения столбцов друг под другом перемещается контент. Из-за значительных различий между основными контрольными точками, поддержка этого макета является более сложной задачей, кроме того, вероятно, придется менять не только общий макет контента, но и его элементы.



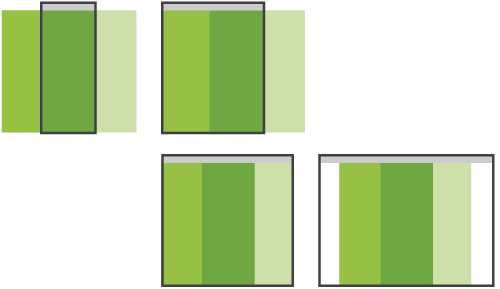
**Tiny Tweaks**

Шаблон вносит небольшие изменения в макет, например, регулирует размер шрифта, меняет размер изображений или перемещает контент. Он хорошо работает на макетах, состоящих из одного столбца, например, одностраничные сайты и статьи с большим количеством текста.



**Off Canvas**

Контент, который используется редко, например, элементы навигации или меню приложения, размещаются за пределами экрана, и показываются только тогда, когда это позволяет сделать размер экрана. На небольших экранах контент открывается одним щелчком.



**Полезные инструменты**

1. Chrome dev tools
2. Сайт для проверки разрешений на разных размерах экрана

<http://mattkersley.com/responsive/>

**Домашнее задание на эту неделю:**

**Вопросы 💎**

1. Как сделать так, чтобы при просмотре на телефоне текст стал читаемым, а картинка - большой?
2. В чём разница между отзывчивым и адаптивным веб-дизайном?
3. Какие величины лучше использовать для шрифтов в гибком дизайне?
4. Какой вид верстки использован на этой картинке? К какой категории шаблонов он относится?
5. Как задать стили для экранов шириной от 800 до 1200 пикселей?
6. Приведите минимум 2 примера как подключать медиазапросы?
7. Как можно задавать гибкие изображения?
8. Как задать стили только для landscape поворота экрана? И что вообще такое landscape и чем он отличается от portrait?
9. Назовите минимум 3 способа как можно тестировать, как выглядит сайт при разных размерах экранов?
10. Самостоятельно изучите, как можно подключить несколько картинок разных размеров через один тег <img>?

**Практическое задание 💎**

1. Доработать уже привычный нам макет с учетом знаний об адаптивности (жду 2 версии - респонсивную и адаптивную):
2. Доработать макет своей страницы из прошлой недели с учетом знаний об адаптивности любым из способов, предложенных выше.

*Страница макета - минимум 5 больших блоков, если берете старый макет, и 3 блока, если берете новый.*

Отчетность - ссылка на гитхаб.

1. Сделать пример из статьи <https://habr.com/ru/company/edison/blog/344878/>

Отчетность - код и видео работы страницы при изменении размера

**Ссылка на свою страничку из википедии 💎**

ОТВЕТЫ

1. <meta name="viewport" content="width=device-width, initial-scale=1">
2. Отзывчивый дизайн=резиновый, т.е. растягивается и сжимается в зависимости от размеров, а адаптивный меняет структуру в зависимости от устройства.
3. Относительные
4. Column drop
5. @media screen and (min-width:800px) and (max-width:1200px)
6. Через link в html-документе, или через @media внутри таблицы стилей
7. С помощью установки свойства max-width:100%
8. @media screen (orientation:landscape) Landscape – это ориентация, когда мы поворачиваем экран устройства, portrait-это стандартная ориентация
9. Chrome devtools, <http://mattkersley.com/responsive/> , тестирование непосредственно на устройствах разного размера
10. <img srcset="elva-fairy-320w.jpg 320w, //\*

elva-fairy-480w.jpg 480w,

elva-fairy-800w.jpg 800w"

sizes="(max-width: 320px) 280px,

(max-width: 480px) 440px,

800px"

src="elva-fairy-800w.jpg" alt="Elva dressed as a fairy">

**srcset** включает названия изображений, среди которых браузер выберет нужное и их размеры.Перед каждой запятой части значения в таком порядке:

1. Название изображения (elva-fairy-480w.jpg.)
2. Пробел.
3. **Актуальная ширина картинки** **в пикселах**(480w) —здесь используется w вместо px.  Эта настоящая ширина изображения, которая может быть просмотрена в свойствах картинки на компьютере

**sizes** определяет перечень медиавыражений (например, ширину экрана) и указывает предпочтительную ширину изображения, когда определённое медиавыражение истинно. мы пишем:

1. **Медиа-условие** ((max-width:480px)) —что медиа-условие описывает возможное состояние экрана. В этом случае, мы говорим "когда viewport width меньше или равен 480 пикселям".
2. Пробел.
3. **Ширину слота** (в оригинале "width of the slot"), занимаемую изображением, когда медиа-условие истинно. (440px)