**Создание адаптивного дизайна**

**Цель работы:** изучить концепцию адаптивного веб-дизайна и опробовать несколько способов создания адаптивного дизайна

В результате выполнения работы студент должен

Знать:

* основные идеи адаптивного веб-дизайна;
* CSS media queries и их роль в адаптивных шаблонах;

Уметь:

* конвертировать стандартный резиноый шаблон в адаптивный шаблон, который адаптируется к различным размерам браузера;
* строить и редактировать основные объекты;
* выполнять основные операции над объектами;
* изменять цвет и толщину линии, цвет заливки.

**Общие положения**

Основная идея адаптивного веб-дизайна состоит в том, что сайт должен «реагировать» на устройство, на котором его просматривают. В более широком плане это может означать следующее:

* Адаптация шаблона в соответствии с различными размерами экранов, начиная от широкоэкранных настольных компьютеров до крошечных телефонов
* Изменение размеров изображения в соответствии с разрешением экрана
* Обслуживание низкоскоростных изображений на мобильных устройствах
* Упрощение элементов страницы для использования в мобильных устройствах
* Скрытие несущественных элементов на небольших экранах
* Предоставление больших по размеру и удобных для нажатия пальцами ссылок и кнопок для мобильных пользователей
* Выявление и реагирование на мобильные функции, такие как геолокация и ориентация устройства.

Однако, в большинстве случаев, когда люди говорят о «адаптивном веб-дизайне», они имеют в виду главным образом первую из этих позиций: создание реагирующего шаблона, который сможет легко приспосабливаться к различным размерам экрана.

В настоящее время большинство веб-сайтов предназначены для просмотра в браузерах настольных компьютеров с окном около 1000px в ширину. Если окно вашего браузера значительно меньше этого размера, то содержание страницы или выглядит сдавленным, или у вас появляется горизонтальная полоса прокрутки. Аналогичным образом, если вы выводите ваш браузер на широкоформатный экран, то содержание часто выглядит слишком растянуто по горизонтали, или появляется излишнее количество белого пространства на странице.

Так как сегодня используется много разных размеров экрана и ширины браузера, то веб-дизайнерам очень трудно удовлетворить всех пользователей, использующих традиционную технику.

Мобильные браузеры с их крошечными экранами между 320 и 768 пикселями в ширину делают ситуацию только хуже. Большинство современных мобильных устройств до некоторой степени позволяют обходить ограничения с помощью масштабирования и панорамирования веб-страниц при помощи пальцев. Однако нельзя уйти от факта, что просмотр традиционной веб-страницы, предназначенной для рабочего стола, на мобильном браузере является малоприятным делом.

Вы можете решить эту проблему путем создания двух веб-сайтов — один для настольных компьютеров и один для мобильных устройств — но тогда вам потребуется дополнительное время на разработку, а также вам придется поддерживать два набора шаблонов для вашего сайта.

Это объясняет, почему разработка адаптивного веб-дизайна является очень важным вопросом. При создании реагирующего дизайна вам нужно создать только одну версию вашего веб-сайта, которая будет прекрасно работать на всех устройствах, начиная от крошечных мобильных и заканчивая широкоэкранными компьютерами. Сайт реагирует по мере того, как изменяется экран или браузер, становится шире или уже, адаптируя свой шаблон соответствующим образом.

**Понимание media queries**

Ключом к созданию адаптивных шаблонов является стандарт CSS media queries. С помощью media queries вы можете создать множество наборов правил CSS, которые применяются только когда видимая область браузера находится в рамках определенной ширины или высоты, или в пределах диапазона заданной ширины / высоты. Например, при создании media query для окон, ширина которых менее 700 пикселей, можно применить набор пользовательских правил CSS, которые автоматически подстраивают элементы страницы, когда страница выводится на узкие окна браузера.

Media queries работают почти во всех современных браузерах, за исключением Internet Explorer 8 и его более ранних версий.

Вот некоторые media queries, которые, скорее всего, придется использовать при построении адаптивных шаблонов:

**min-width: width**

Применяется, если ширина окна больше или равна ширине шаблона

**max-width: width**

Применяется, если ширина окна меньше или равна ширине шаблона

**min-device-width: width**

Применяется при ширине экрана устройства больше или равной ширине шаблона

**max-device-width: width**

Применяется при ширине экрана устройства меньше или равной ширине шаблона

*Вы также можете указать точные значения ширины, используя свойства «width» и «device-width», хотя вы, скорее всего, будете использовать варианты минимальной и максимальной ширины, которые позволяют задать диапазон значений.*

Разница между шириной шаблона и шириной устройства очень трудноуловима, но важна:

**width** — это ширина окна браузера. На браузерах обычных компьютеров, width, как правило, меньше ширины экрана. Тем не менее, на мобильных браузерах, width обычно больше ширины экрана, так как большинство мобильных браузеров используют «виртуальное окно», которое больше размеров экрана, что позволяет пользователю увеличивать и уменьшать масштаб изображения, а также перемещаться по области окна, перетаскивая изображения. Например, Mobile Safari использует виртуальное окно, ширина которого составляет 980px, хотя ширина экрана устройства IOS, как правило, составляет от 320 до 768px (в вертикальном режиме). Более подробно про окна вы узнаете [далее](http://naikom.ru/blog/archives/5566#viewport-meta-tag).  
**device-width** — это ширина экрана устройства. На настольном компьютере эта информация обычно не так важна для нас, как веб-разработчиков. Знать ширину экрана на мобильном устройстве может быть полезно, так как в таком случае вы сможете адаптировать шаблон, чтобы он соответствовал экрану устройства. Однако, если вы используете мета-тег viewport для блокировки ширины окна в соответствии с шириной экрана (подробнее об этом позже), то вы можете просто использовать width вместо device-width для достижения того же результата.

*Другое трудноуловимое различие в том, что width (min-width/max-width) измеряется в пикселях CSS, в то время как device-width (min-device-width/max-device-width) измеряется в пикселях устройства. Когда страница увеличена до 100% на мобильном устройстве, то 1 CSS пиксель = 1 пикселю устройства. Однако если масштаб страницы уменьшен, тогда 1 CSS пиксель меньше 1 пикселя устройства. Дальнейшее развитие событий осложняется появлением дисплеев высокой чёткости, таких как Retina Display от Apple, в котором пиксель устройства может фактически вмещать более одного пикселя экрана! ( Это делает жизнь веб-разработчика легче, так как ему не нужно больше беспокоиться о том, будет дисплей устройства нормальной или же высокой плотности).*

**Ход работы**

**Задание 1 Создайте неадаптивный шаблон**

1. Создайте HTML-файл (скопируйте код):

<!doctype html>

<html>

<head>

<title>3 ЭТАЖ</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="main0.css" />

</head>

<body>

<div id="header">

<div class="inner">

<h2>3 ЭТАЖ</h2>

</div>

</div>

<div id="pageBody">

<div id="nav">

<ul>

<li><a href="#">Главная</a></li>

<li><a href="#">О нас</a></li>

<li><a href="#">Контакты </a></li>

<li><a href="#">Друзья-приятели</a></li>

</ul>

</div>

<div id="content">

<div class="inner">

<h1>Адаптивный дизайн</h1>

<p>Адаптивный дизайн состоит из трёх главных принципов:

Резиновый макет

Резиновые изображения

Медиазапросы

1. Гибкий макет

Для создания гибкого макета мы должны использовать только относительные единицы измерения: em для размера шрифтов и % для размеров элементов. В редком использовании px тоже нет ничего плохого, но лучше свести их использование к минимуму.

Раньше ширина этого блога была фиксированной и равнялась 1080px, вместо этого нам нужно взять какое-то более гибкое значение, например 90% окна браузера. При этом я решил, что сайт все равно не должен быть слишком широким, и поэтому сразу же ограничил его ширину до моих любимых 1080px, в итоге было:

h1,#logo,#width,#slogan,#h ul{width:1080px;margin-left:auto;margin-right:auto}

Стало:

h1,#logo,#width,#slogan,#h ul{width:90%;max-width:1080px;margin-left:auto;margin-right:auto}

Почему именно 90%? Просто мне так захотелось, я решил, что отступы по 5% справа и слева будут самое то. Можно было поставить 99% или 80%, всё зависит от вашей задумки.

Итак, есть 2 колонки: основная колонка (та, где вы сейчас находитесь глазами) и правая колонка. Их ширина равнялась 780px и 280 px, но теперь мы должны пересчитать её в процентах, и вот вам формула на все случаи жизни:

target / context = result

или скажем проще:

Ширину (размер шрифта) нужного элемента в px разделить на ширину (размер шрифта) родительского элемента в px = искомый результат в % (или em).

На этом можно было бы закончить рассказ о резиновом макете, так как всю суть я уже рассказал, но для лучшего усвоения я продолжу.

Итак, основная колонка в фиксированном варианте равна 780 px, её контекст (родитель) в фиксированном варианте имеет ширину 1080px, в итоге:

780 / 1080 = 0.72222222222

Так как нам нужны проценты, а не em, то результат нужно умножить на 100, в итоге получаем:

72.222222222%

Вы, наверное, надеялись, что я округлю это число? Но нет, в книге этого категорически не рекомендуют. Поэтому смотрите по ситуации, насколько важна в вашем случае точность. В случае двух колонок такая точность совершенно не важна, поэтому я со спокойной душой решил округлять всё до 4 знаков после точки. В итоге вот мой CSS для основной колонки:

#c{float:left;width:72.2222%;} /\* 780 / 1080 \*/

Обратите внимание, рекомендуется оставлять комментарии с расчётами, иначе вы забудете и запутаетесь что на что делить в дальнейшем. Лично я люблю оставлять в комментариях только первое число - делитель.

Проделываем тоже самое с правой колонкой:

#r{float:right;width:25.9259%;color:#000} /\* 280/1080 \*/

Надеюсь, суть ясна, теперь перейдём к размерам шрифта, который рассчитывается точно также.

По умолчанию десктопные браузеры выставляют размер шрифта 16px, но нам пиксели не желательны, поэтому если вы хотите оставить это же значение нужно:

16 (нужное значение) / 16(значение по умолчанию) = 1 em

Но я считаю шрифт в 16 пикселей слишком мелким, я предпочитаю 18, поэтому:

18 / 16 = 1.125 em

Пишем в CSS:

body{font-family:"Times New Roman";font-size:1.125em;}

Размер шрифта моего логотипа «Site on!» в фиксированном макете составляет 40px, переведём его в em:

40/18=2.2222em

Вы обратили внимание? Теперь мы делим не на 16px, а на 18px, так как у родителя (у тега body) моего логотипа размер 18px, в итоге CSS:

body{font-family:"Times New Roman";font-size:1.125em;} /\* 18px \*/

#logo a{text-decoration:none;font-size:2.2222em;color:#000} /\* 40px \*/

И так с каждым элементом, действительно нудно, не правда ли? Но наши труды с лихвой окупятся в виде стабильного трафика с мобильных устройств.</p>

</div>

</div>

<div id="sidebar">

<div class="inner">

<h2>О нас</h2>

<p>Наши специальности:

<p>ИС - Информационные системы

<p>ПИ - Прикладная информатика

<p>ПС - Програмирование в компьютерных системах</p>

<h2>Контакты</h2>

<p>Приходите к нам на Гагарина,7.</p>

</div>

</div>

</div>

<div id="footer">

<div class="inner">

<h3>ПИ ИС ПС</h3>

</div>

</div>

</body>

</html>

1. Создайте файл CSS, main.css.

Шаблон должен включать в себя:

* **Хедер**, #header. Полоса простирается всю ширину окна.
* **Контент**, # content. Эта область находится слева, и занимает 65% от ширины окна.
* **Сайдбар** (боковая колонка), # sidebar. Это область находится справа от области основного содержания, и занимает 20% от ширины окна.
* **Меню навигации**, #nav. Оно так же находится в правой части сайта, а именно справа от боковой панели. Оно занимает 15% от ширины окна.
* **Футер**, #footer. Как и в заголовке, эта полоса простирается через всё окно.

Контент, сайдбар и навигацию обернули в #pageBody div, которому даем максимальную ширину в 1200px, чтобы предотвратить слишком большое растяжение содержимого сайта на очень широких окнах браузера.



1. Измените размера окна браузера, чтобы увидеть, как изменяется шаблон в зависимости от изменений ширины видимой области.

По мере сужения видимой области все элементы страницы тоже сужаются пропорционально. Все это прекрасно работает, пока ширина браузера не становится меньше 1000px, после чего три колонки начинают теснить друг друга. При дальнейшем сужении окна браузера, примерно до 600px, колонки становятся настолько сжатыми, что текст внутри становится нечитаемым.

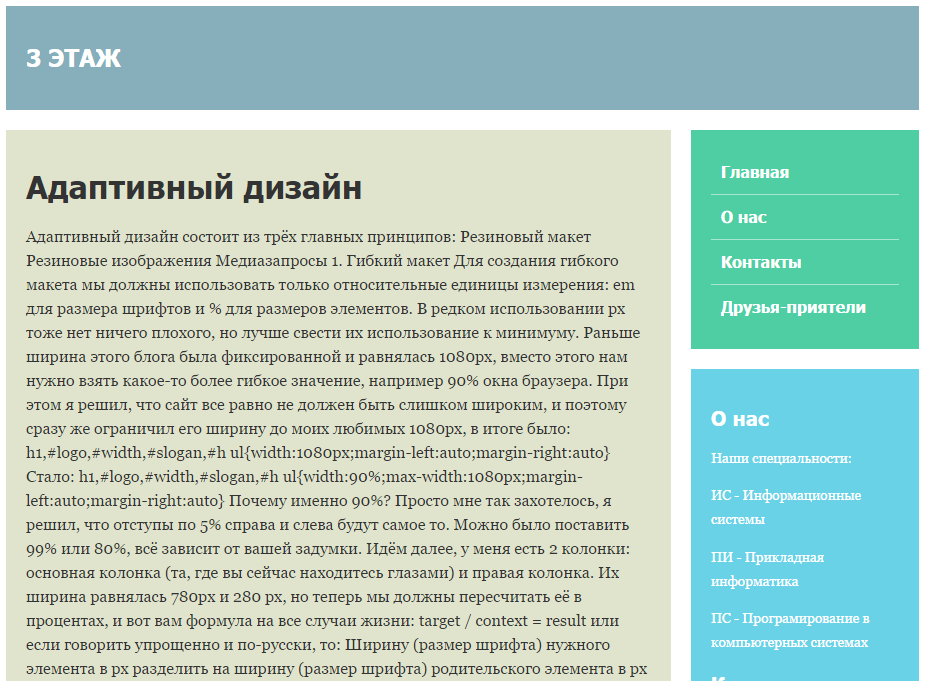
**Задание 2 Создайте media-запрос для адаптации к окнам шириной в 1000 или менее пикселей**

1. Напишите media query, чтобы понять, является ли ширина окна меньше или равной 1000px. Если ширина именно такая - запускаются следующие наборы правил:

* Первый набор правил растягивает ширину контента с 65% до 75%.
* Второй набор правил ставит навигацию в верхнюю часть боковой панели, устанавливая ширину обоих элементов равной 25%.
* Третий набор правил удаляет правое поле из содержания боковой панели и добавляет вместо этого верхнее поле, так, чтобы получилось некоторое пространство по вертикали между навигацией и сайдбаром.

Добавьте созданный media query в конце main.css.

1. Проверьте адаптивный шаблон по media query в действии.

****

**Задание 3 Создайте media-запрос для адаптации к окнам шириной в 700 или менее пикселей**

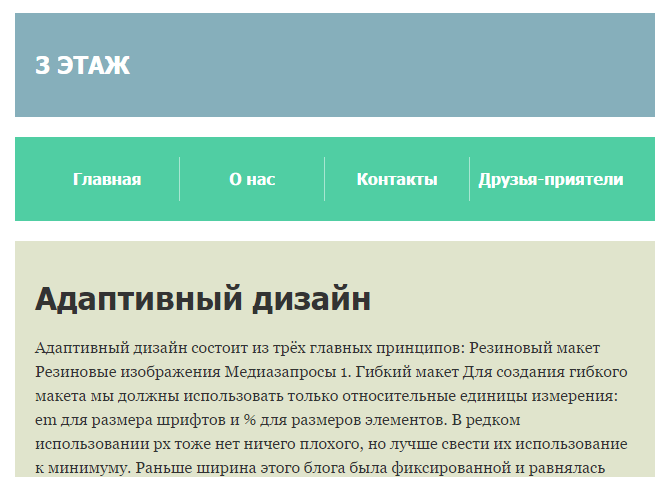
Когда страница просматривается в окне шириной меньше или равной 700 пикселям, применяются следующие наборы правил:

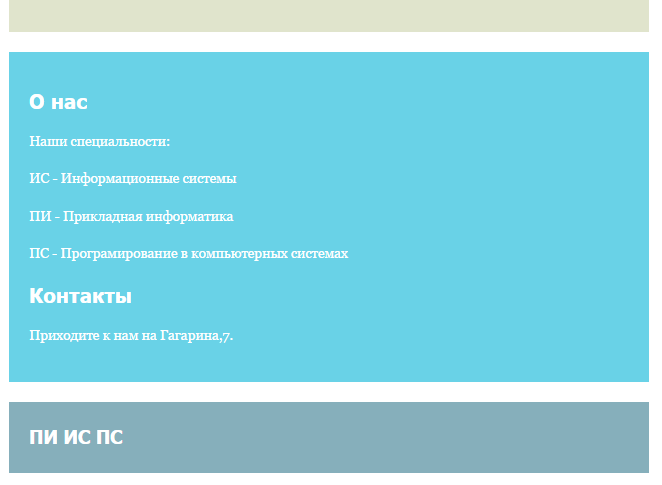
* Растягиваем контент на всю ширину окна и удаляем правый отступ
* Перемещаем навигационное меню над контентом и растягивает его на всю ширину окна
* Располагаем навигационные горизонтально, а не вертикально
* Перемещаем сайдбар под контент и растягиваем его на всю ширину окна

1. Напишите media query, реализующий рассмотренные правила и поместите в конце main.css.

:

На рисунке ниже показан новый шаблон с одной колонкой при просмотре в узком окне:



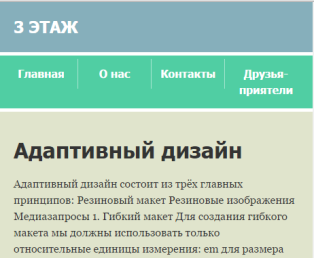


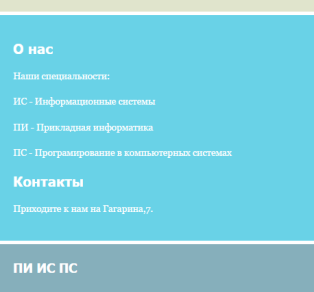
**Задание 4 Создайте media-запрос для адаптации к viewports шириной в 480 или менее пикселей**

При ширине окна в 480 пикселей, наш шаблон с одной колонкой теряет много ценного места на экране из-за ширины белого поля в 20 пикселей.

1. Уберите внешние отступы и уменьшите внутренние отступы до 5 пикселей, а также уменьшите отступы на заголовок и навигацию, чтобы получить больше места для размещения содержания.
2. Проверьте работу шаблона при любой ширине браузера, начиная с 1600 пикселей и заканчивая 320 пикселями.

Изображения ниже показывают шаблон, просматриваемый на браузере шириной в 480 пикселей, до и после применения последнего media query.





**Задание 5 Работа с мобильными устройствами: viewport meta tag**

1. Добавьте в head мета-тег:

<meta name="viewport" content="initial-scale=1, maximum-scale=1" />

**На iPhone наш адаптивный шаблон ведет себя так, будто ширина окна чуть менее 1000 пикселей, а не 320 пикселей. Почему?**

Браузер iPhone уменьшит масштаб страницы, текст будет маленьким и трудно читаемым. В данном случае сработал наш media query на «1000 или менее пикселей» — он уменьшил страницу до двух колонок, вместо трех, но при этом media query на «480 или менее пикселей» не запускается, хотя ширина экрана iPhone составляет всего 320px при вертикальном режиме и 480px — при горизонтальном. Что происходит?

iPhone, как и большинство мобильных браузеров, использует виртуальное окно, которое больше по размерам, чем физический экран. Это позволяет мобильным браузерам отображать веб-страницы, предназначенные для больших окон браузеров рабочих столов. Первоначально окно масштабируется таким образом, чтобы вся страница помещалась на экран. Затем пользователь может потянуть или дважды нажать на конкретную область окна, чтобы увеличить ее.

Окно iPhone шириной 980px прекрасно работает для большинства веб-сайтов, которые предназначены для просмотра на рабочем столе с разрешением монитора 1024×768. Вот почему наш media query «1000 или менее пикселей» запускается на iPhone. Тем не менее, мы создали реагирующий шаблон, который разработан специально, чтобы адаптироваться к маленьким экранам, со страницей, увеличенной до полного ее размера.

**С помощью мета-тега viewport** можно управлять размером окна мобильного браузера.

В нашем случае мы хотим, чтобы ширина окна соответствовала ширине экрана устройства, и чтобы соблюдалось соотношение 1:1 между CSS пикселями страницы и пикселями экрана устройства. Мы можем сделать это с помощью двух значений в тегах viewport: первоначальный масштаб initial-scale и максимальный масштаб maximum-scale:

<meta name="viewport" content="initial-scale=1, maximum-scale=1" />

* initial-scale = 1 — этот показатель гарантирует, что страница, при самом первом ее просмотре будет полностью увеличена так, чтобы ширина окна совпадала с шириной экрана устройства при вертикальном режиме — например, 320 пикселей на iPhone и 768 пикселей на IPad.
* maximum-scale = 1 — этот показатель предотвращает увеличение масштаба страницы больше, чем при соотношении 1:1 в дальнейшем, даже при повороте в горизонтальный режим. Другими словами, этот показатель заставляет iPhone увеличить ширину окна до 480px в горизонтальном режиме. Если мы не включим максимальную величину масштаба, то ширина окна будет 320px, как в вертикальном режиме, так и в горизонтальном режиме. Иногда это может быть именно то, что вы хотите получить, но в случае нашего адаптивного шаблона такой вариант может стать причиной слишком сильного растягивания содержания в горизонтальном режиме.

Теперь, при просмотре нашей страницы на iPhone в вертикальном режиме, ширина окна составляет 320px. При просмотре в горизонтальном режиме ширина составляет 480px. Это позволяет успешно задействовать наш media query «480 или менее пикселей», как вы можете видеть на изображении ниже:

**Задание 6 Добавление графических изображений**

1. Добавьте изображение макета sh.jpg. в файл .html и параметры форматирования его в файл стилей, а также параметры для масштабирования при изменении ширины экрана просмотра – в соответствующие медиа-запросы.