ВВЕДЕНИЕ В АДАПТИВНОСТЬ.

Принципы веб-разработки. Понятие адаптивности и ее актуальность. Способы реализации адаптивности. Принципы адаптивного дизайна.

Принципы веб-разработки

В числе основополагающих принципов веб-разработки говорят о доступности страниц. И смысл здесь не только в правах доступа или правильно размещенном коде.

Наши страницы должны быть доступны людям, говорящим на разных языках, использующим разные устройства и обладающим разными возможностями.

Именно поэтому, современные веб-страницы должны отвечать ряду требований, таких как:

- кроссбраузерность,
- семантичность,
- валидность,
- адаптивность.

Понятие адаптивности, ее актуальность

Адаптивный дизайн призван сделать веб-страницы и отображение их содержимого соответствующими тому устройству, с которого они просматриваются.

Это значит, что один и тот же сайт можно просматривать на самых разных устройствах, независимо от разрешения и формата экрана, – смартфонах, планшетах, ноутбуках и т.д. При этом просмотр будет одинаково удобен – пользователям мобильных устройств, например, не нужно будет масштабировать отдельные области сайта, чтобы не

промахнуться мимо нужной ссылки. Важно, чтобы ваш сайт хорошо смотрелся и правильно отображался у любого из пользователей, независимо от того, какое устройство он использует.

Про актуальность этой темы сказано уже немало, поэтому сегодня, хочется поговорить о подходах, приемах и технологиях, которые этой самой адаптивности от нашего сайта позволят добиться.

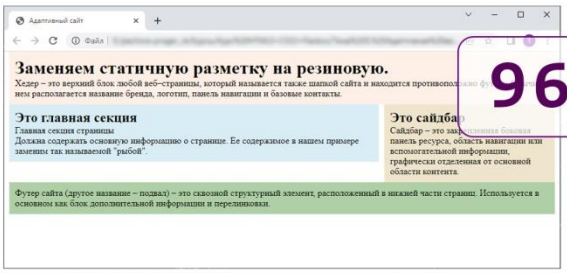
Способы реализации адаптивности

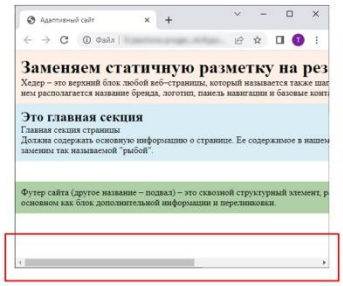
Использование того же Bootstrap для построения адаптивного сайта конечно здорово, но мы много раз говорили и продолжаем говорить, что, используя разные дополнительные инструменты, важно понимать, как они функционируют «под капотом».

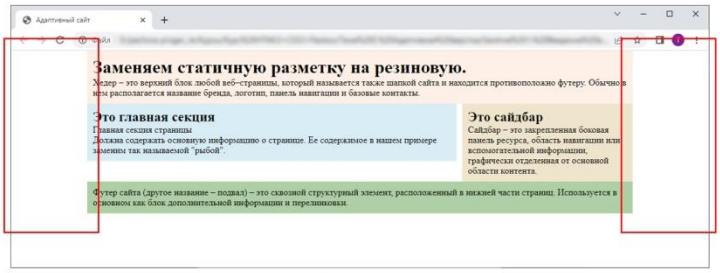
Понимание принципов построения адаптивности сайта позволит не просто бездумно их использовать, а действительно обратить себе на пользу, избежав при этом излишней загруженности вашего проекта.

Поэтому приступим. Имея абсолютно фиксированный сайт с заданной в пикселях шириной блоков и пунктах для размера шрифта, мы также далеки от адаптивности как студент первокурсник от защиты диплома.

При **фиксированной** разметке элементы имеют статичные размеры, неизменные при любых условиях. При попытке просмотра таких страниц на устройствах меньшего размера мы увидим полосы прокрутки, большего – пустые области вокруг элементов.



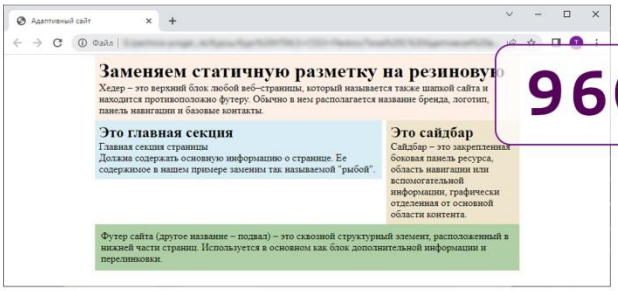


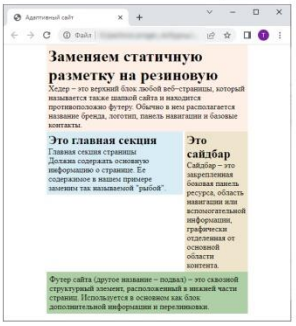


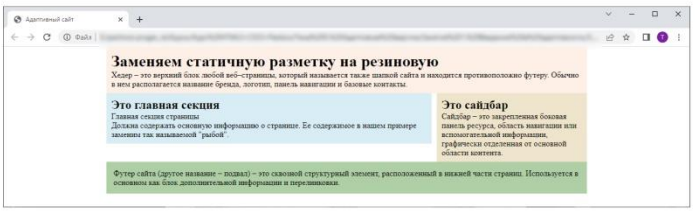
600px

1200px

Первой попыткой заставить наш сайт менять свою структуру при изменении устройства отображения страницы или ширины окна браузера в целом может стать замена пикселей на проценты, а пунктов на em-ы. Такая верстка скорее резиновая, чем адаптивная, но все же.







600px

1200px

Резиновые страницы будут растягивать и сжимать себя и свое содержимое при изменении размеров экрана или устройства.

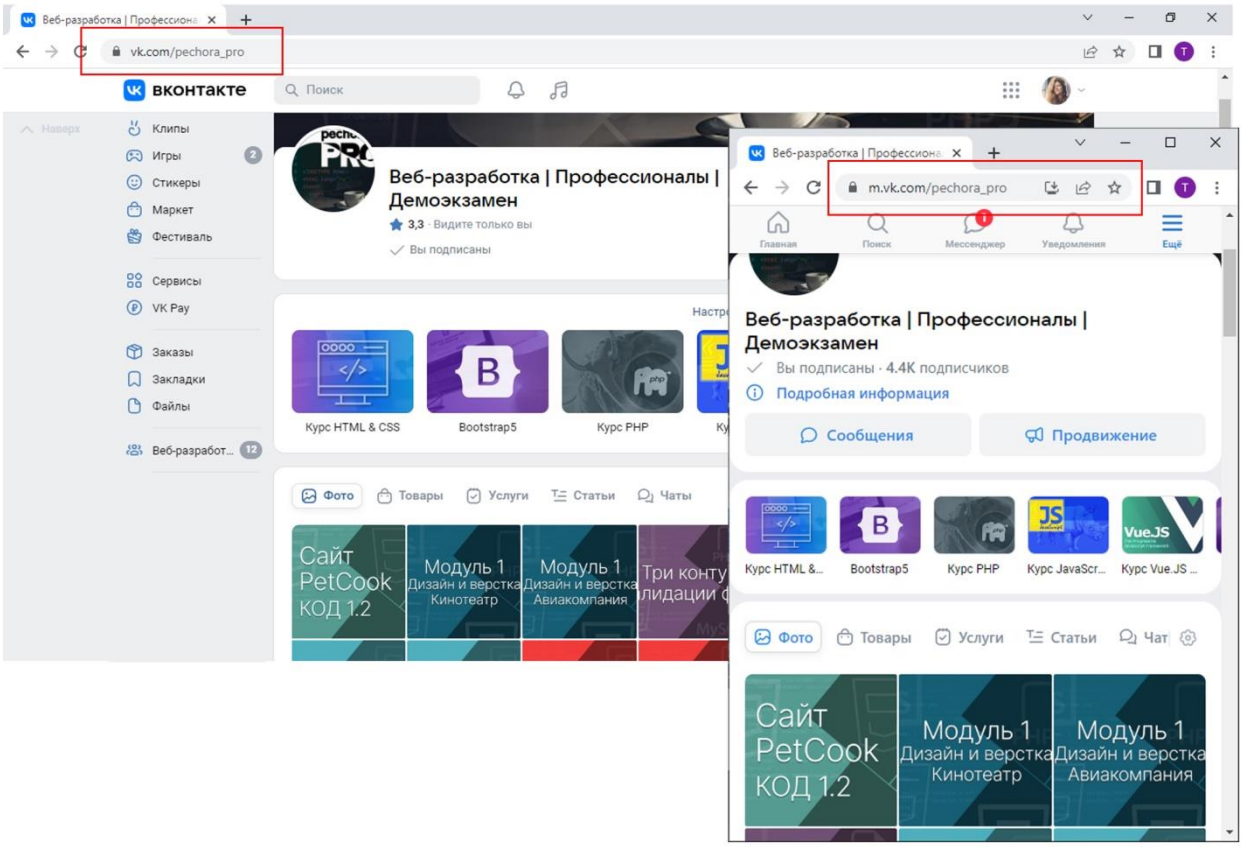
Более совершенная версия такого подхода - CSS flexbox и **гибкая** разметка. Помимо всего прочего flexbox-ы позволят нам автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство, что пойдет на пользу отображению нашего сайта на разных устройствах.

Кстати, стоит отметить так называемую **гибридную** верстку (пример в задании к лекции), сочетающую в себе элементы статичной и гибкой разметки. При таком подходе часть элементов страницы имеет фиксированный размер, а часть подстраивается под устройство отображения и размер экрана. В качестве подспорья здесь следует обратить внимание на свойство **display**, которое определяет, как элемент должен быть показан в документе.

Здесь до чистой адаптивности еще далеко, но для простых страниц этого может быть вполне достаточно.

Интересно, что до появления адаптивности как таковой активно использовался другой подход.

Разработчики создавали **две** полноценные **версии сайтов**: для ПК и для мобильного устройства и размещали их на разных поддоменах. При таком подходе хоть и обеспечивалось удобство просмотра страниц с разных устройств, но сложность создания и обслуживания проекта возрастала в несколько раз.



Под каждый тип операционной системы нужна была своя версия сайта. Весь трафик ресурса делился на 2 версии сайта, а материал нужно было добавлять на страницы дважды.

В отличие от мобильных версий, адаптивный дизайн – это один адрес сайта, один дизайн, одна система управления и содержание сайта.

Принципы адаптивного дизайна

Итак, а**даптивная** верстка (мобильная или responsive) – с помощью стилей и правил описывает разные варианты поведения элементов при разных размерах экрана или устройств.

Проектирование начинается с адаптивной версии веб-сайта для мобильных устройств. На этом этапе дизайнеры и разработчики стремятся правильно передать смысл и основные идеи с использованием небольшого экрана и всего одной

колонки. Содержимое страниц минимально. Такая концепция разработки получила название **Mobile first –** сначала мобильные.

Второй составляющей адаптивности для нас станет прогрессивное улучшение **– Progressive enhancement**.

Подход предполагает, что веб-интерфейсы должны создаваться поэтапно, от простого к сложному. На каждом из этапов должен получаться законченный веб-интерфейс, который будет лучше, красивее и удобнее предыдущего.

Вместе с этим подходом необходимо упомянуть еще один. **Graceful degradation** может выражаться в возможности работы при отключенном JavaScript, в достаточно аккуратном отображении интерфейса в браузере, не поддерживающем новые свойства CSS3, в адекватном отображении сайта при отключенных изображениях. В каждом из этих случаев работа пользователя с интерфейсом будет в принципе возможна, хотя и не так удобна.

И последняя часть – обеспечение **отзывчивости** дизайна. Здесь будем рассматривать три составляющие:

- Применение гибкого макета на основе сетки.
- Использование гибких изображений (**flexible media**).
- Работа с медиа-запросами (**media queries**).

ЗАДАНИЕ №25. ПЕРЕВОД РАЗМЕТКИ

Рассмотрим процесс перевода фиксированной разметки в резиновую.

1. Создадим страницу с блоками container, header, section, aside, footer.

```
<div class="container">
    <header>
        <h1>Заменяем статичную разметку на резиновую</h1>
        <p>Хедер – это ...</p>
    </header>
    <section>
        <h2>Это главная секция</h2>
        <p>Главная секция страницы</p>
        <p>...</p>
    </section>
    <aside>
        <h2>Это сайдбар</h2>
        <p>Сайдбар – это ...</p>
    </aside>
</div>
<footer>
    <p>Футер сайта ...</p>
</footer>
```

2. Зададим статичную разметку блокам в файле style.css.

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font: 12pt, Arial, Helvetica, sans-serif;
}

.container,
footer {
    width: 960px;
    margin: 0 auto;
}

.container {
    background-color: #e2e2e2;
}
```

```
header {
    background-color: #fff0e6;
    padding: 10px;
}

section {
    background-color: #d8edf5;
    width: 650px;
    float: left;
    padding: 10px;
}

aside {
    background-color: #efe5cc;
    width: 300px;
    float: right;
    padding: 10px;
}

footer {
    background-color: #afd0a6;
    padding: 10px;
    clear: both;
}
```

3. Проверим поведение страницы при изменении размера окна.
4. Перепишем стили, используя формулы.

Ширина каждого блока задается в процентном соотношении по формуле:

Цель\Контекст * 100% = Результат.

Размер текста вычисляется по формуле:

Размер в px\16= Размер в em

```
body {
    font: 0.75em, Arial, Helvetica, sans-serif;
}

.container,
footer {
    ...
    width: 70%;
}

header {
    padding: 1%;
```

```
}

section {
    ...
    width: 67.708333333%;
    padding: 1%;
}

aside {
    ...
    width: 31.25%;
    padding: 1%;
}

footer {
    ...
    padding: 1%;
}
```

5. Проверим поведение страницы при изменении размера окна.