GeekBrains

HTML/CSS

Основы создания адаптивного сайта

[HTML5/CSS3]

На этом уроке

1. Изучим основы создания адаптивного сайта.
2. Узнаем, что такое медиазапросы.
3. Узнаем, как сделать гибкие размеры блоков.

Оглавление

Адаптивный сайт

Адаптивная вёрстка предполагает отсутствие горизонтальной полосы прокрутки и масштабируемых областей при просмотре на любом устройстве, читабельный текст и большие области для кликабельных элементов. С помощью медиазапросов можно управлять компоновкой и расположением блоков на странице, перестраивая шаблон таким образом, чтобы он адаптировался под разные размеры экранов устройств.

Адаптивные свойства блоков

Устанавливают максимальную ширину элемента. Значение ширины элемента будет вычисляться в зависимости от значений установленных свойств `width`, `max-width` и `min-width`.

max-width

Посмотрим, чем же отличается свойство `width`, которое мы использовали до этого урока, и в каких ситуациях его необходимо будет поменять на `max-width`. Если вы задали родительскому блоку значение `width`, при адаптиве у вас появится полоса прокрутки в нижней части браузера. Чтобы такого не произошло, вам нужно поменять это свойство ширины на `max-width`.

Одна из самых частых ошибок начинающих веб-разработчиков — вывод, что абсолютно все значения `width`: надо поменять на `max-width`:. Это неверный вывод, так как новое значение хорошо подходит для родительского блока, но если вы его зададите дочерним элементам, когда у родителя задано свойство `display`: `flex`;, то получится, что значение ширины дочерних элементов равно 0. Рассмотрите пример подобной ошибки.

Это свойство в большинстве ситуаций относится к родительским блокам. Рассмотрите похожий пример: можно попробовать поменять разрешение экрана и увидеть, как блоки перестраиваются, но не разъезжаются в разные стороны экрана.

min-width

Если сравнивать два, казалось бы, похожих свойства `max` и `min width`, то можно заметить большую разницу в использовании: `max-width` встречается в большинстве проектов, а вот `min-width` встречается реже, может и вовсе отсутствовать. Свойство `min-width` используется, чтобы размеры контентной части не становились меньше, чем 320 px.

Пример: вам необходимо уменьшить ширину блока `result` до значения меньше, чем 320px, и увидеть, что появилась полоса прокрутки.

Мы можем использовать это свойство как минимальный ограничитель, и даже если пользователь откроет сайт с устройства с меньшим разрешением экрана, контент будет располагаться верно, правда, с полосой прокрутки.

min-height

Это свойство очень популярно при создании адаптивного сайта, и его можно встретить практически в каждом проекте. Свойство `min-height` используется, если контента может стать больше или при адаптиве контент начнёт перестраиваться.

Пример. При уменьшении экрана на отметке в 375px блок становится больше, и мы можем заметить, что в этом блоке может поместиться любое количество текста, — это особое преимущество свойства `min-height`. Попробуйте изменить это значение на `height` и посмотреть, что будет с заголовком внутри блока при уменьшении экрана.

max-height

Так же, как и в примере с `min-width`:, свойство `max-height` используется в разы реже, чем `min-height`:, так что его можно встретить далеко не в каждом адаптивном сайте.

Рассмотрим, где же мы можем его применить и для чего оно используется, на примере, в котором есть ограниченное значение высоты блока текста. Мы можем заметить, что высота блока зависит от контента внутри, но если менять разрешение экрана, то блок становится очень большим. Тогда появляется полоса прокрутки на мобильном устройстве, чтобы можно было пролистать контент внутри блока и он не стал больше, чем 200px.

Эмуляция мобильных устройств

Чтобы посмотреть отображение сайта на планшете или мобильном телефоне, в отладчике браузера есть специальный функционал — эмуляция мобильных устройств.

Как посмотреть отображение сайта на планшете и мобильном телефоне

1. Открыть сайт в браузере.
2. Открыть отладчик браузера: F12, Ctrl + Shift + L, правой клавишей мыши — «Просмотр кода элемента».
3. Левый верхний угол отладчика (toggle device toolbar).
4. Вы сможете увидеть отображение вашего сайта на мобильном устройстве.

Если вы открыли мобильную версию сайта и замечаете, что сайт выглядит точно так же, как десктопная версия, значит, вы забыли добавить свойство `viewport`.

Viewport

Типичный сайт, оптимизированный для мобильных устройств, содержит следующий метатег:

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

Свойство `width` определяет размер окна просмотра. Он может быть установлен на определённое количество пикселей, скажем, `width=600` или на специальное значение `device-width`, которое означает ширину экрана в пикселях CSS в масштабе 100%. Есть также соответствующие значения `height` и `device-height`, которые могут быть полезны для страниц с элементами, изменяющими размер или положение на основе высоты окна просмотра.

Сайты могут устанавливать свой `viewport` на определённый размер. Например, определение «`width=320, initial-scale=1`» может использоваться для точного размещения на маленьком дисплее телефона в портретном режиме. Это может вызвать проблемы, когда браузер не отображает страницу большего размера. Чтобы исправить это, браузеры, если необходимо, увеличат ширину окна просмотра, чтобы заполнить экран по заданной шкале.

Для страниц, задающих начальный или максимальный масштаб, это значит, что свойство `width` фактически переводит в минимальную ширину `viewport`. Например, если ваш макет должен иметь ширину не менее 500 пикселей, вы можете использовать следующую разметку. При ширине экрана

более 500 пикселей браузер будет расширять область просмотра (а не увеличивать), чтобы она соответствовала экрану.

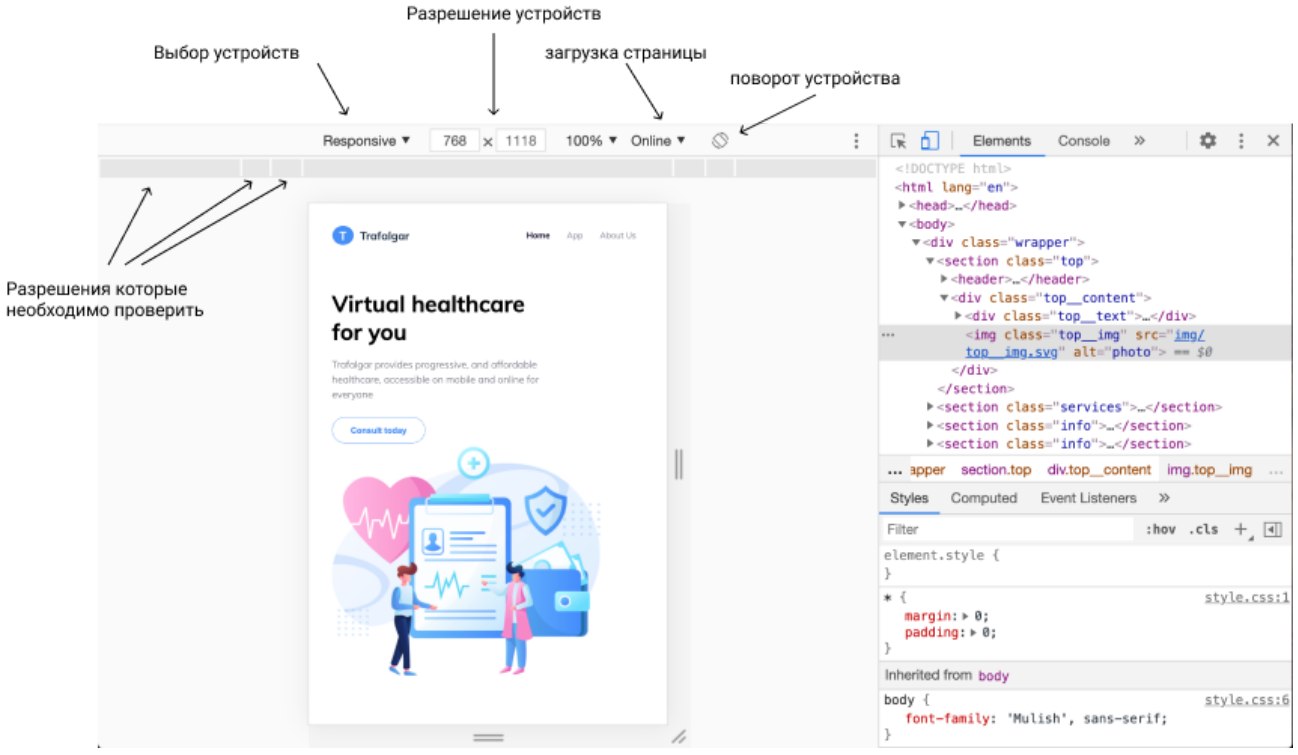
Другие доступные атрибуты — `minimum-scale`, `maximum-scale`, и `user-scalable`. Эти свойства влияют на начальный масштаб и ширину, а также ограничивают изменения уровня масштабирования.

Не все мобильные браузеры обрабатывают изменения ориентации таким же образом. Например, Mobile Safari часто просто увеличивает масштаб страницы при смене с вертикальной ориентации на горизонтальную, вместо того чтобы выкладывать страницу так, как если бы она была первоначально загружена в «ландшафт». Если веб-разработчики хотят, чтобы их настройки масштаба оставались неизменными при переключении ориентации на iPhone, они должны добавить значение `maximum-scale`, чтобы предотвратить масштабирование, которое иногда имеет нежелательный эффект, мешающий пользователям изменять масштаб:

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

Toggle device toolbar

Рассмотрим функционал мобильной разработки в Google Chrome.



На иллюстрации мы можем увидеть, что у программы есть весь необходимый функционал без перегруженных возможностей. С ним должен справиться любой начинающий веб-разработчик.

Медиазапросы

Наряду с типами носителей в CSS3 включена поддержка различных технических параметров устройств, на основе которых требуется загружать те или иные стили. К примеру, можно определить смартфон с максимальным разрешением 640 пикселей и для него установить одни стилевые свойства, а для остальных устройств — другие. Также можно выявить различные характеристики вроде наличия монохромного экрана и ориентации (портретная или альбомная). Все характеристики легко комбинируются, поэтому допустимо задать стиль только для устройств в альбомной ориентации с заданным разрешением экрана.

Возможности медиазапросов не ограничиваются выявлением мобильных устройств, с их помощью можно создавать адаптивный макет. Такой макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину, число колонок, размеры изображений и текста. Медиазапросы ограничивают ширину макета, и при достижении этого значения, к примеру, за счёт уменьшения окна или при просмотре на устройстве с указанным размером, применяется другой стиль.

Синтаксис

Все запросы начинаются с правила `@media`, после чего следует условие, в котором используются типы носителей, логические операторы и медиафункции.

Плюсы

1. Чёткое отображение страниц на экране с любым разрешением.
2. Возможность просмотра группы контента на любом устройстве.
3. Отсутствие горизонтальной полосы прокрутки независимо от размера окна.

Условия для Media Queries

```
@media screen and (max-width: XXXpx) { }
@media screen and (min-width: XXXpx) { }
@media screen and (min-width: XXXpx) and (max-width: YYYpx) { }
@media screen and (max-device-width: XXXpx) { }
```

С их помощью можно отслеживать разрешение экрана пользователя и отображать необходимые стили для каждого разрешения или устройства.

Пример:

```
@media screen and (min-width: 1024px) {
        .content {
            background-color: #ccc;
        }
}
```

В результате, если у пользователя экран больше или равен 1024px, задний фон для `content` будет серым.

Логические операторы

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

Оператор and

Оператор `and` связывает друг с другом разные условия:

```
@media screen and (min-width: 1024px) {
        /* CSS-стили */
}
```

Стили этого запроса будут применяться только для экранных устройств с шириной области просмотра не более 1024px.

```
@media (min-width: 320px) and (max-width: 768px) {
        /* CSS-стили */
}
```

Стили этого запроса будут применяться для всех устройств при ширине области просмотра от 320px до 768px включительно.

Оператор запятая

Оператор запятая работает по аналогии с логическим оператором `or`.

```
@media screen, projection {
        /* CSS-стили */
}
```

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

Оператор not

Оператор `not` позволяет сработать медиазапросу в противоположном случае. Ключевое слово `not` добавляется в начало медиазапроса и применяется ко всему запросу целиком, то есть запрос:

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

будет эквивалентен запросу:

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

Стратегии использования медиазапросов

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

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.

В этом материале мы рассмотрели основы, которые помогут вам создать адаптивный сайт и решить проблемы, которые могут возникать по мере создания проекта. Также важная особенность — использование всех возможностей flexbox, которые мы изучали ранее.

Как прижать подвал к нижней части экрана

Когда пользователь открывает короткие по высоте страницы, сразу бросается в глаза, что информация, которая должна быть в нижней части окна просмотра, «прилипает» к содержимому страницы и находится где-то посередине, а то и вверху. Все полноценно оформленные страницы сайтов имеют «прижатый» (`sticky`) к низу страницы футер. Это означает, что когда контента между хедером (шапкой, `header`) и футером (подвалом, `footer`) мало, футер всё равно остается внизу окна

браузера, как бы вы ни изменяли размер окна. Если окно не будет умещать в себе контент и футер, появится прокрутка.

Прижать `footer` к низу экрана — это требование по умолчанию, норма, которую нужно запомнить и соблюдать, тем более сделать это довольно просто. Иногда дизайнер может сделать макет, в котором не нужно прижимать footer к низу экрана, но в 99% случаев при создании сайтов это потребуется. Кроме этого, прижать футер нужно, чтобы сайт корректно отображался по высоте на разных мониторах, например на 13-дюймовом нетбуке или на 24-дюймовом настольном мониторе, и при разных разрешениях экрана.

Первый способ

В нашем курсе мы рассматривали технологию flexbox, которая отлично подходит для расставления элементов на странице. Можно прижать подвал к нижней части сайта с использованием этой технологии.

vh

Относительная единица измерения — 1% от высоты области просмотра.

HTML:

```
<div class="wrapper">
    <div class="content">Lorem ipsum dolor sit amet, consectetur adipisicing
elit.   Adipisci iure tempore similique inventore, aliquam dicta placeat.</div>
    <div class="footer"></div>
</div>
```

CSS:

```
* {
  margin: 0;
  padding: 0;
}

.wrapper {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.content {
  flex-grow: 1;
}

.footer {
```

```
  background-color: green;
  height: 100px;
}
```

Пример работы кода.

Плюсы этого подхода:

1. Может быть любая высота блока `footer`.
2. Вместо блока `wrapper` можно использовать `body`.

Второй способ

Особенность этого способа заключается в использовании `calc()` и единицы измерения `vh`, которые поддерживаются только современными браузерами.

calc()

Используется для указания вычисляемого значения свойств, которые в качестве значений используют размер. Это позволяет задавать значения, основанные на сложении или вычитании разных единиц измерений. Например, можно задать 100% — 20px. Если значение не может быть вычислено, оно игнорируется.

1. Firefox до версии 16.0 поддерживает значение `-moz-calc`.
2. Chrome до версии 26.0 поддерживает значение `-webkit-calc`.
3. Safari с версии 6.0 поддерживает значение `-webkit-calc`.
4. Internet Explorer 9.0.

HTML:

```
<div class="content">Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Adipisci iure tempore similique inventore, aliquam dicta placeat.</div>
<div class="footer"></div>
```

CSS:

```
* {
  margin: 0;
  padding: 0;
}
.content {
  background-color: blue;
  min-height: calc(100vh - 100px);
}
```

```
.footer {
  background-color: green;
  height: 100px;
}
```

100vh — высота окна браузера.

100px — высота футера.

Есть множество способов прижатия футера к низу экрана, мы рассмотрели некоторые из них. На практике вам покажут несколько методов, как разобранных в методичке, так и альтернативных. Прижатый футер — это правило хорошего тона. Сделать качественный сайт — важная часть веб-разработки.

Дополнительные материалы

Адаптивный сайт

Используемые источники

1. Meta viewport.
2. Media screen.

Практическое задание

1. Доделать главную страницу, если ещё не доделали.
2. Создать адаптивную версию главной страницы.
3. * Познакомиться с сеткой, присутствующей в макете.
4. * Приступить к вёрстке страницы «Контакты».

Задачи со звёздочкой предназначены для продвинутых учеников, которым мало сделать обычное ДЗ. Это дополнительные задания, и можно их не выполнять.