|  |  |
| --- | --- |
| **How to make responsiveness super simple with CSS Variables**  A quick tutorial on how to create responsive websites in 2018.  If you haven’t heard of CSS Variables before, it’s a new feature of CSS which gives you the power of variables in your stylesheet, without having to do any setup.  In essence, CSS Variables allow you to skip the old way of setting styles:  h1 {  font-size: 30px; }  navbar > a {  font-size: 30px; }  …in favour of this:  :root {  --base-font-size: 30px; }  h1 {  font-size: var(--base-font-size); }  navbar > a {  font-size: var(--base-font-size); }  While the syntax might seem a bit weird, this gives you the obvious benefit of being able to change the font sizes across your entire app through only changing the--base-font-size variable.  Now let’s see how this new technology can make your life easier when building responsive websites.  **The setup**  We’re going to add responsiveness to a portfolio website which looks like this:  https://cdn-images-1.medium.com/max/1000/1*tLQrkgJJhKV3YrzPxsVVFA.png  It looks nice when viewed on your desktop. However, as you can see on the left image below, this layout doesn’t work well on mobile.  https://cdn-images-1.medium.com/max/600/1*CZkMgq0rp9nTdChVxwq33g.png  https://cdn-images-1.medium.com/max/600/1*zpFS--eNMyAzkdZWS1lLRQ.png  To the left: how it looks on mobile initially. To the right: how we want it to look.  On the right image, we’ve changed a a few things on the styles to make it work better on mobile. Here’s what we have done:   1. **Rearranged** the grid so that it’s stacked vertically instead of across two columns. 2. **Moved** the entire layout a bit more up 3. **Scaled**the fonts down   In order to do this, we needed to change the following CSS:  h1 {  font-size: 30px; }  #navbar {  margin: 30px 0; }  #navbar a {  font-size: 30px; }  .grid {  margin: 30px 0;  grid-template-columns: 200px 200px; }  More specifically, we needed to make the following adjustments inside of a media query:   * Reduce font size of the h1 to 20px * Reduce the margin above and below the #navbar to 15px * Reduce the font size inside the #navbar to 20px * Reduce the margin above the .grid to 15px * Change the .grid from from two-columns to one-column   ***Note:****There is, of course, much more CSS in this application, even within these selectors. However, for the sake of this tutorial, I’ve stripped away everything which we aren’t changing in the media query. Check out*[*this Scrimba playground*](https://scrimba.com/c/cwJmLhn)*to get the entire code.*  **The old way**  Doing all of this would be possible without CSS Variables. But it would require an unnecessary amount of code, as most of the bullet points above would need their own selector inside the media query, like this:  @media all and (max-width: 450px) {    navbar {  margin: 15px 0;  }    navbar a {  font-size: 20px;  }    h1 {  font-size: 20px;  }  .grid {  margin: 15px 0;  grid-template-columns: 200px;  }  }  **The new way**  Now let’s see how this can be solved with CSS Variables. To begin with, we’ll rather store the values which we are reusing or changing inside variables:  :root {  --base-font-size: 30px;  --columns: 200px 200px;  --base-margin: 30px; }  And then we’ll simply use these variables across the app:  #navbar {  margin: var(--base-margin) 0; }  #navbar a {  font-size: var(--base-font-size); }  h1 {  font-size: var(--base-font-size); }  .grid {  margin: var(--base-margin) 0;  grid-template-columns: var(--columns); }  Once we have this setup, we can simply change the values of the variables inside the media query:  @media all and (max-width: 450px) {  :root {  --columns: 200px;  --base-margin: 15px;  --base-font-size: 20px; }  This is much cleaner than what we had before. We’re only targeting the :root, as opposed to specifying all the selectors.  We’ve reduced our media query from **four selectors down to one** and from **thirteen lines down to four**.  And this is just a simple example. Imagine a full-blown website where, for example, the --base-margin control most of the free spacing around the app. It’s a lot easier to just flip the value of it, as opposed to filling your media query up with complex selectors.  To sum up, CSS Variables are definitely the future of responsiveness. If you want to learn this technology once and for all, I’d recommend that you check out my [free course on the subject on Scrimba.](https://scrimba.com/g/gcssvariables)  You’ll become a CSS Variables master in no time :) | **Отзывчивость – это очень просто с CSS-переменными**  Быстрый урок по созданию отзывчивых сайтов в 2018.  Если вдруг до этого вы ни разу не слышали о CSS-переменных, то поясним, что это новая функция CSS, благодаря которой в таблице стилей вы получаете все возможности переменных без какой-либо дополнительной настройки.  Иначе говоря, CSS-переменные позволяют отойти от старых способов создания стилей:  h1 {  font-size: 30px; }  navbar > a {  font-size: 30px; }  …в пользу следующего:  :root {  --base-font-size: 30px; }  h1 {  font-size: var(--base-font-size); }  navbar > a {  font-size: var(--base-font-size); }  На первый взгляд такой синтаксис выглядит не совсем привычно, но зато дает вам очевидное преимущество – с ним вы сможете изменять размеры шрифтов по всему приложению, работая только с одной переменной --base-font-size .  Давайте посмотрим, как эта новая техника облегчит вам жизнь при создании отзывчивых сайтов.  **Установка**  Мы будем добавлять отзывчивость сайту-портфолио, который выглядит примерно так:  https://cdn-images-1.medium.com/max/1000/1*tLQrkgJJhKV3YrzPxsVVFA.png  На десктопе все смотрится довольно неплохо. Но, как вы видите на левой картинке ниже, с мобильной версией дела обстоят иначе. https://cdn-images-1.medium.com/max/600/1*CZkMgq0rp9nTdChVxwq33g.png  https://cdn-images-1.medium.com/max/600/1*zpFS--eNMyAzkdZWS1lLRQ.png  Слева: то, как страница открывается на мобильных устройствах сейчас. Справа: как мы хотим, чтобы она выглядела.  На картинке справа мы немного изменили стили для лучшего отображения на мобильных. Вот, что мы сделали:   1. **перегруппировали** гриды таким образом, чтобы содержимое отображалось вертикально, а не растягивалось на две колонки. 2. **сдвинули** весь шаблон немного вверх. 3. **уменьшили** размер шрифтов.   Для этого потребовались следующие корректировки CSS:  h1 {  font-size: 30px; }  #navbar {  margin: 30px 0; }  #navbar a {  font-size: 30px; }  .grid {  margin: 30px 0;  grid-template-columns: 200px 200px; }  Если говорить точнее, то в медиа-запросах мы подправили следующее:   * уменьшили размер шрифта h1 до 20px; * уменьшили отступы сверху и снизу от #navbar до 15px; * уменьшили размер шрифта внутри #navbar до 20px; * уменьшили отступ над .grid до 15px; * заменили два столбца в .grid на один.   ***Примечание:****Конечно же, в нашем приложении, да даже внутри этих селекторов, было намного больше CSS. Но, чтобы не уходить от темы, я убрал все лишнее и незадействованное в нашем в медиа-запросе.*  [*Вот здесь в Scrimba*](https://scrimba.com/c/cwJmLhn)*вы найдете полный код приложения.*  **Старый способ**  Сделать все это можно и без CSS-переменных. Однако тогда вас ждет лишний пласт ненужного кода, т.к. почти каждый из этих пунктов требует создания собственного селектора внутри медиа-запроса. Например, вот такого:  @media all and (max-width: 450px) {    navbar {  margin: 15px 0;  }    navbar a {  font-size: 20px;  }    h1 {  font-size: 20px;  }  .grid {  margin: 15px 0;  grid-template-columns: 200px;  }  }  **Новый способ**  Давайте посмотрим, как решить эту проблему с помощью CSS-переменных. Для начала, сохраним внутри переменных значения, которыми будем неоднократно пользоваться или изменять:  :root {  --base-font-size: 30px;  --columns: 200px 200px;  --base-margin: 30px; }  А затем будем брать эти переменные и вставлять их в нужное место приложения:  #navbar {  margin: var(--base-margin) 0; }  #navbar a {  font-size: var(--base-font-size); }  h1 {  font-size: var(--base-font-size); }  .grid {  margin: var(--base-margin) 0;  grid-template-columns: var(--columns); }  Как только мы прописали все это, то можем смело менять значения переменных внутри медиа-запроса:  @media all and (max-width: 450px) {  :root {  --columns: 200px;  --base-margin: 15px;  --base-font-size: 20px; }  Такой код – более чистый. И вместо того, чтобы создавать множество селекторов, мы просто прописываем :root.  Таким образом, мы уменьшили наш медиа-запрос с **четырех селекторов к одному,** и **13 строк кода превратились в 4**.  Это был простой пример. А теперь представьте себе полноценный сайт, в котором, к примеру, --base-margin отвечает за все пустое пространство вокруг приложения. Куда проще взять и поменять его значение один раз, чем перегружать медиа-запрос комплексными селекторами.  Подводя итог, можно сказать, что CSS­-переменные – это, однозначно, будущее отзывчивости. |