Зачем нужен JavaScript?

Мы рассмотрели базовые механики синтаксиса языка JavaScript. Но, как мы выяснили ранее - JavaScript был создан специально для того, чтобы добавить интерактивности сайтам. Такие сайты реагируют на ваши действия: добавляют лайк, когда вы нажимаете на «сердечко»; загружают новые посты в ленту, когда вы доходите до конца страницы; показывают оповещения о новом сообщении или письме. Для этого и нужен JavaScript. Сегодня это один из самых популярных и востребованных языков программирования, поэтому он пригодится каждому веб-разработчику.

Для того чтобы увидеть простоту и пользу на практике изменим написанный вами ранее index.html. У сайта, который вы создали, сделаем две темы: светлую и тёмную. Наша задача — сделать так, чтобы посетители могли переключаться между ними, щёлкая по кнопке «Изменить тему». Всё содержимое страницы находится внутри элемента с классом page. За светлую тему отвечает класс light-theme, а за тёмную — dark-theme. Чтобы темы менялись, нам нужно убирать с page один класс и добавлять другой.

Задание

1. Откройте папку “theme switcher”
2. Измените заголовки под себя
3. В папке создайте новый текстовый файл, назовите его script, расширение смените на .js
4. Перед закрывающимся тегом </body> добавьте строку с подключением этого файла:

<script src="scripts.js"></script>

5.

Находим элемент с помощью querySelector

Перед тем, как писать скрипт, давайте вручную переключим тему на странице — это поможет лучше понять задачу. Для этого поменяем классы у элемента page прямо в разметке.

```
<body class="page light-theme">
```

Чтобы увидеть разницу в темах (сформулированную в файле style css) замените light-theme на dark-theme

Мы переключили тему на странице, вручную поменяв классы в разметке. Пришло время сделать это с помощью JavaScript.

Программа на JavaScript — это последовательность инструкций, которые указывают браузеру выполнить какие-то действия. Мы собираемся менять классы у элемента page, и в первую очередь должны получить доступ к нему из скрипта.

```
document.querySelector('селектор');
```

Обратите внимание, эта инструкция состоит из двух частей. Первая часть — это элемент, внутри которого будет искать JavaScript. Словом document обозначается веб-страница, к которой подключили скрипт. Неважно, как называется файл на самом деле, в JavaScript это всегда «документ». Он является элементом-родителем для любого другого элемента на странице.

Вторая часть инструкции — это то, что нужно сделать. Её называют методом. Метод querySelector ищет по селектору, который указан в скобках.

Итак, чтобы найти на странице элемент с классом page, мы должны написать:
`document.querySelector('.page');`

Задание

1. В файле script.js в первой строке напишите `document.querySelector('.page');`

Выводим элемент в консоль

Мы сказали JavaScript найти на странице элемент с классом page, но как узнать, что он его действительно нашёл? И что элемент — тот самый? Для этого нам понадобится консоль, с который мы уже умеем общаться.

Чтобы вывести найденный с помощью querySelector элемент в консоль, нам необходимо усовершенствовать нашу предыдущую строку, добавлением соответствующей команды:

```
console.log(document.querySelector('.page'));
```

Инструкция получилась длинной и сложной, но вскоре мы разберём, как её упростить. А пока займёмся тестированием кода. Так называется этап создания программы, когда мы проверяем, что всё работает, как надо. И это не менее важно, чем написание кода.

Давайте убедимся, что JavaScript нашёл нужный элемент, а заодно потренируемся работать с консолью.

Задание

1. В первой строке у нас уже есть поиск элемента .page `console.log(document.querySelector('.page'));`
2. Точно таким же способом поступим с элементом страницы '.theme-button' на следующей строке

Убираем класс с помощью classList.remove

Мы убедились, что JavaScript действительно нашёл элемент с классом page. Пора заняться переключением тем. У найденного элемента есть ещё второй класс — light-theme, благодаря ему на странице включена светлая тема. Чтобы не было смешения стилей, когда мы переключим тему, этот класс у элемента нужно убрать. Для этого нам понадобится метод classList.remove. Используют его так:

```
элемент.classList.remove('класс');
```

Метод убирает с элемента тот класс, который указан в скобках. Обратите внимание, что мы не ставим точку перед именем класса в classList.remove. Это не селектор, JavaScript и так знает, что мы имеем дело с классом.

Обратите внимание, что буква «L» в названии метода заглавная. JavaScript чувствителен к регистру и “любит” в синтаксисе использовать известный вам CamelCase. Если написать classlist.remove, метод не сработает.

Чтобы выключить светлую тему, убрав класс light-theme у элемента page, используем инструкцию:

```
document.querySelector('.page').classList.remove('light-theme');
```

Получился такой «паровозик»: найди элемент, а потом убери у него класс. Давайте посмотрим, что произойдёт, когда эта инструкция выполнится

Задание

1. Откройте инструменты разработчика, перейдите во вкладку Элементы
2. Ваш body будет иметь класс class="page light-theme"
3. Переключитесь на консоль и выполните `document.querySelector('.page').classList.remove('light-theme');`
4. Вернитесь во вкладку Элементы - в body осталось только class="page"

Добавляем класс элементу с помощью classList.add

Теперь нужно включить тёмную тему, добавив элементу page класс dark-theme. Для этого воспользуемся методом classList.add:

```
элемент.classList.add('класс');
```

Этот метод добавляет элементу класс, указанный в скобках. При этом, если вы заглянете в index.html, то увидите, что ничего не изменилось: класс light-theme по-прежнему на месте, а dark-theme отсутствует. Почему так? Дело в том, что скрипт не меняет исходный файл с разметкой, но, выполняя инструкции, меняет страницу прямо в браузере пользователя.

На второй строке напишите инструкцию, которая добавит класс dark-theme элементу page: document.querySelector('.page').classList.add('dark-theme');

Перейдите во вкладку index.html в редакторе кода и убедитесь, что в исходной разметке классы у элемента page не изменились.

Задание

1. Приведите вид вашего файла script.js к:

```
document.querySelector('.page').classList.remove('light-theme');
document.querySelector('.page').classList.add('dark-theme');
```

Объявляем переменную

Мы переключили тему на странице, но строки с кодом получились длинными и трудночитаемыми. А ещё пришлось дважды искать с помощью querySelector один и тот же элемент. Мы его уже один раз нашли, зачем искать снова?

Создадим переменную! Сохранять в переменные можно что угодно, в том числе элементы, например:

```
let header = document.querySelector('header');
```

Когда в коде встречается переменная, браузер вместо её имени подставляет присвоенное ей значение. Благодаря этому мы можем написать, например, так:

```
console.log(header);
header.classList.add('new-class');
```

Давайте создадим в нашей программе переменную и запишем в неё элемент с классом page. С помощью этой переменной мы сделаем наши инструкции короче и понятней. И заодно убедимся, что ничего не сломалось.

Задание

1. На первой строке объявите с помощью let переменную с именем page и присвойте ей значение document.querySelector('.page')
2. На следующей строке замените document.querySelector('.page') на переменную page

```
page.classList.remove('light-theme');
```

3. По аналогии замените document.querySelector('.page') на последней строке.

Знакомимся с обработчиком событий

Мы переключили тему, используя JavaScript, и улучшили код, добавив переменную. Но тема на странице переключилась лишь однажды, а нам нужно, чтобы темы сменяли друг друга каждый раз при нажатии на кнопку «Изменить тему».

JavaScript следит за всем, что происходит на странице, и мы можем сказать ему, что сделать, если пользователь, например, кликнет по кнопке. Сначала мы находим саму кнопку и сохраняем её в переменную:

```
let themeButton = document.querySelector('.theme-button');
```

После того, как кнопка была найдена, мы указываем JavaScript, что делать, когда по этой кнопке кликнут. Это может быть, например, вывод сообщения в консоль:

```
themeButton.onclick = function() {
  console.log('Клик!');
};
```

Инструкции, которые выполняются после клика по кнопке, располагаются внутри фигурных скобок.

Клик по кнопке, с точки зрения браузера, это событие. Свойство onclick означает «по клику» и говорит JavaScript, какое событие мы хотим отслеживать. А та часть инструкции, которая идёт после onclick, называется обработчиком событий. Кнопка «Изменить тему», по клику на которую должны переключаться темы, имеет класс theme-button. Давайте найдём её и посмотрим, как работает обработчик событий. Для тестирования снова используем консоль: скажем JavaScript выводить сообщение при каждом клике

Задание

1. На 5 строке объявите переменную themeButton и запишите в неё элемент с классом theme-button
2. На следующей строке добавьте обработчик событий: `themeButton.onclick = function() {};`
3. Внутри фигурных скобок обработчика добавьте вывод в консоль: `console.log('Кнопка нажата!');`
4. В мини-браузере дважды нажмите на кнопку «Изменить тему».
5. Проверьте, что в консоли появились два новых сообщения.

Переключаем тему по клику

Отлично, мы убедились, что обработчик событий работает: каждый раз, когда мы нажимаем на кнопку, в консоль выводится новое сообщение. Осталось сделать так, чтобы по клику изменялась тема на странице.

Переключать её с помощью classList.remove и classList.add мы уже научились и даже написали весь необходимый код. Чтобы классы менялись по клику, нам нужно перенести эти инструкции внутрь фигурных скобок обработчика событий.

Задание

1. Переместите инструкции

```
page.classList.remove('light-theme');
page.classList.add('dark-theme');
```

внутрь обработчика событий, ниже вывода в консоль.

2. Дважды нажмите на кнопку «Изменить тему». Обратите внимание, что тема переключилась только один раз.
3. Откройте консоль, убедитесь, что сообщения по-прежнему появляются при каждом клике на кнопку.

Знакомимся с classList.toggle

Мы заставили тему переключиться при клике по кнопке! Теперь нашу страницу можно назвать интерактивной.

Есть только одна маленькая проблема: тема переключается лишь один раз. Сообщения в консоли по-прежнему появляются при каждом клике, а значит, обработчик работает. Так в чём же дело?

Это происходит потому, что JavaScript выполняет всё в точности так, как мы ему сказали: берёт элемент page и сначала убирает класс light-theme, а после добавляет класс dark-theme. Так что при первом клике классы меняются и светлая тема переключается на тёмную, а при следующих кликах нашим инструкциям уже нечего делать: dark-theme уже есть — его не надо добавлять, а light-theme уже удалили — удалять нечего.

Так как заставить темы переключаться при каждом клике? Веб-разработчикам часто приходится решать подобные задачи, и создатели JavaScript позаботились, чтобы чередовать классы можно было легко и удобно с помощью метода classList.toggle:

```
элемент.classList.toggle('класс');
```

Если класс у элемента есть, метод classList.toggle ведёт себя как classList.remove и класс у элемента убирает. А если указанного класса у элемента нет, то classList.toggle, как и classList.add, добавляет элементу этот класс.

Используем метод-переключатель classList.toggle и убедимся, что при нажатии на кнопку «Изменить тему» у элемента будет то убираться, то добавляться класс light-theme

Задание

1. Удалите инструкции classList.remove и classList.add
2. Внутри обработчика событий напишите инструкцию, которая будет переключать класс light-theme у элемента page:

```
page.classList.toggle('light-theme');
```

3. В браузере дважды нажмите на кнопку «Изменить тему».

Завершаем переключатель тем

Мы написали инструкцию, которая убирает у элемента класс light-theme, если он есть, и добавляет, если его нет. Благодаря этому при клике по кнопке «Изменить тему» светлая тема на странице то выключается, то включается. Но что насчёт тёмной темы? Нам ведь нужно, чтобы темы переключались. Для этого надо, чтобы у элемента page чередовались классы light-theme и dark-theme: сначала добавлялся один и удалялся другой, а потом наоборот. Как это сделать?

Нужно написать вторую инструкцию с classList.toggle. Она будет то добавлять, то удалять класс dark-theme при клике. Чтобы темы переключались, инструкции должны работать в противофазе. Для этого в исходной разметке у элемента page должен быть указан один класс темы, а второй нет. Тогда один класс удалится, а другой, наоборот, добавится к элементу.

В index.html у элемента page есть класс light-theme. Вот что произойдёт, когда мы в первый раз кликнем на кнопку «Изменить тему»:

```
<!-- Исходная разметка -->
<body class="page light-theme">
  ...
</body>

<!-- Разметка в браузере после classList.toggle('light-theme') -->
<body class="page">
  ...
</body>

<!-- Разметка в браузере после classList.toggle('dark-theme') -->
<body class="page dark-theme">
  ...
</body>
```

Когда пользователь кликнет по кнопке второй раз, первая инструкция добавит класс light-theme, а вторая — удалит класс dark-theme. Нечётные клики будут делать то же, что и первый клик, а чётные — то же, что и второй. Пользователь сможет переключить тему столько раз, сколько пожелает.

Добавим внутрь обработчика событий вторую инструкцию и убедимся, что при клике по кнопке «Изменить тему» темы на странице чередуются, как и планировалось

Задание

1. На следующей строке напишите инструкцию, которая будет переключать класс dark-theme у элемента page
2. В браузере дважды нажмите на кнопку «Изменить тему»

Пользуемся!

По итогу мы получил очень простой скрипт буквально в несколько строк кода. Разберем его еще раз:

```
let page = document.querySelector('.page');
```

Здесь мы создали переменную, в которую положили найденный по классу элемент страницы, стили которого хотим контролировать

```
let themeButton = document.querySelector('.theme-button');
```

Еще одна и переменная, и снова элемент, на этот раз - кнопка. Теперь мы сможем отслеживать взаимодействие пользователя с ней

```
themeButton.onclick = function() {
  page.classList.toggle('light-theme');
  page.classList.toggle('dark-theme');
};
```

Создаем обработчик события по клику, внутри которого занимаемся переключением двух классовых значений - светлой и темной темы. Сами темы стилистически описаны в файле style.css