Как сделать слайдер?

Чтобы сделать слайдер (или карусель) на сайте, мы будем использовать готовое решение — плагин swiper. Плагин позволяет просто и быстро реализовать данный функционал без особых знаний JS-кода.

HTML

Начнём с HTML-разметки. Она — не без нюансов. Очень важно сохранять правильные классы и вложенность элементов для работы слайдера.

```
<!-- контейнер слайдера -->
<div class="swiper-container">
  <!-- враппер -->
  <div class="swiper-wrapper">
    <!-- слайды -->
    <div class="swiper-slide">Slide 1</div>
    <div class="swiper-slide">Slide 2</div>
    <div class="swiper-slide">Slide 3</div>
    ...
  </div>
  <!-- кнопки пагинации -->
  <div class="swiper-pagination"></div>

  <!-- кнопки для навигации -->
  <div class="swiper-button-prev"></div>
  <div class="swiper-button-next"></div>
</div>
```

Для удобства в коде расставлены комментарии. Давайте пройдёмся по каждому элементу отдельно.

1. Swiper-container — див с классом для контейнера. Для этого дива в стилях плагина указаны два важных свойства — ширина и overflow. Ширина нужна слайдеру, чтобы корректно рассчитать размеры слайдов (именно так, слайды рассчитываются на основании ширины контейнера), а overflow: hidden — чтобы слайды не выходили за пределы контейнера.
2. Swiper-wrapper — крайне важный элемент слайдера, без которого его работа просто невозможна. Именно этот див «крутится», когда вы взаимодействуете со слайдером. Из важных стилей здесь — ширина и высота, равные 100%, а также box-sizing: content-box.

3. Swiper-slide — это дивы-слайды. Здесь всё просто, им по умолчанию задан flex-shrink: 0, чтобы флексбокс не пытался сжимать их, а они брали размеры от контейнера и располагались как нужно.
4. Swiper-pagination — изначально пустой див, который автоматически заполняется пагинацией, если такая настройка включена. Если не включена — див можно удалить.
5. Swiper-button-prev, swiper-button-next — дивы для кнопок. По факту тут может быть любой класс, и лучше сделать через тег button.

Всё это перечислено тут не случайно, ведь не всегда есть возможность использовать именно эти классы. Чтобы при использовании сторонних классов слайдер адекватно работал (при учёте соответствующих настроек типа slideClass), нужно делать те же стили новым классам.

Подключение файлов

Перед тем как инициализировать плагин, нужно его правильно подключить.
Будем использовать cdn, то есть просто ссылки из интернета, чтобы было быстрее.

Чтобы подключить css-файл плагина, используем данную строку:

<link rel="stylesheet" href="https://unpkg.com/swiper/swiper-bundle.min.css" />

Вспоминая правила подключения файлов и правила весов селекторов, подключим этот файл раньше файла стилей.

Со скриптами всё аналогично — подключаем скрипт плагина раньше файла JS и обязательно держим все скрипты перед закрывающим боди:

<script src="https://unpkg.com/swiper/swiper-bundle.min.js"></script>

В итоге подключение будет выглядеть примерно так:

```
<title>Слайдер</title>
<link rel="stylesheet" href="https://unpkg.com/swiper/swiper-bundle.min.css" />
<link rel="stylesheet" href="style.css">
```

```
  <script src="https://unpkg.com/swiper/swiper-bundle.min.js"></script>
  <script src="script.js"></script>
</body>
```

JS

Теперь напишем код, который инициализирует плагин слайдера. По большому счёту, за нас уже всё написали в документации, будем пользоваться ей.
Чтобы плагин заработал, нужно передать ему селектор нашего контейнера, а также ряд настроек. Давайте это сделаем.

```
const swiper = new Swiper('.swiper-container', {
  slidesPerView: 1,
  loop: true,

  // пагинация
  pagination: {
    el: '.swiper-pagination',
  },

  // навигация
  navigation: {
    nextEl: '.swiper-button-next',
    prevEl: '.swiper-button-prev',
  },
});
```

Здесь мы создали переменную swiper (она может пригодиться в работе со слайдером, но не обязательна) и вызвали плагин через запись new Swiper. Далее внутри в кавычках передали селектор контейнера — класс swiper-container. Напоминаю, что здесь может быть и ваш кастомный класс.

Добавим картинки — и получим вполне рабочий слайдер со стрелками и точками для переключения.

Настройки плагина

У любого плагина, как правило, большое количество настроек. Конечно, применять их все не обязательно, всё зависит от конкретной задачи.
Используя синтаксис объектов (фигурные скобки, а в них пара «свойство: значение»), можно легко задавать слайдеру абсолютно разное поведение. Например, мы захотим сделать не один слайд видимым сразу, а три. Как это сделать? За это отвечает свойство slidesPerView, уже написанное в нашем коде. Также, чтобы слайды не были слипшимися, добавим им отступ. За это отвечает свойство spaceBetween.

```
slidesPerView: 3,
spaceBetween: 30,
```

Таким образом мы сказали, что хотим видеть три слайда, а отступ между ними — 30 пикселей. И всё прекрасно отработало.

В нашем коде также присутствует настройка loop: true, которая отвечает за то, чтобы слайдер не останавливался при достижении конца слайдов, а начинался с начала, то есть зацикливался.

Отдельно остановимся на навигации и пагинации.

```
// навигация
  navigation: {
    nextEl: '.swiper-button-next',
    prevEl: '.swiper-button-prev',
  },
```

Как видите, навигация как бы сама является неким свойством, в которое вложены два других свойства. В целом, даже логически понятно, что здесь перечисляются селекторы для кнопок слайдера: nextEl — кнопка, которая будет переключать слайдер вправо, prevEl — влево. Здесь, конечно, можно указывать любые классы.

С пагинацией дела чуть-чуть сложнее, хотя код — проще.

```
pagination: {
    el: '.swiper-pagination',
  },
```

Всё дело в том, что данный код просто создаёт точки внутри элемента swiper-pagination. Но если попытаться на них нажать — ничего не произойдёт. Разработчики слайдера специально сделали это отдельной командой, чтобы у вас была гибкость в настройке. Команда clickable: true поможет сделать это.
В итоге код пагинации изменится.

```
pagination: {
    el: '.swiper-pagination',
    clickable: true
  },
```

Все эти настройки находятся в документации, для всех достаточно подробно описан функционал и возможные значения свойств, поэтому всё, что нужно, — изучать документацию и находить необходимые свойства и их значения, чтобы заставить слайдер работать так, как нужно вам.

Также не забывайте о том, что у swiper есть довольно большой раздел Demos, где можно посмотреть часто встречающиеся вариации слайдеров с готовым кодом.

Стилизация

Помимо дополнительных настроек, будет полезно знать, как менять стили плагина под себя. Вряд ли в нашем дизайне когда-нибудь встретится слайдер на 100% такой же, какой предлагает сам плагин, поэтому стоит понять, как его менять.

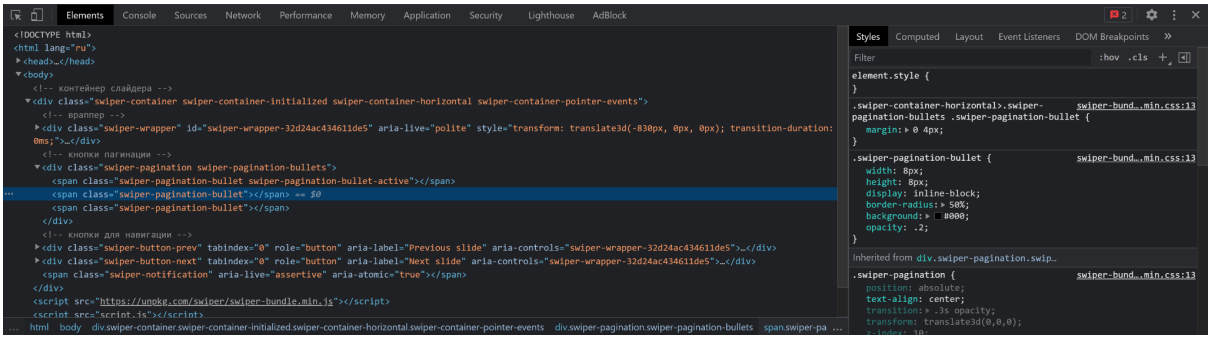
Для начала вспомним, как у нас подключены CSS-файлы:

<link rel="stylesheet" href="https://unpkg.com/swiper/swiper-bundle.min.css" />
<link rel="stylesheet" href="style.css">

Сперва идёт файл стилей библиотеки, а затем — наш. Это очень важно. Иначе вы просто не сможете перестилизовать плагин.

По большому счёту, всё, что нам нужно на данном этапе, — это изменять стили точек слайдера, изменять кнопки. Как же это делать?
Чтобы это сделать — используйте DevTools. Важный момент — даже если мы указывали какие-то свои классы для стилизации, плагин наверняка добавит свои классы и свои стили на эти классы. И чтобы соответствовать весам селектора из плагина, проще всего использовать селекторы плагина, а не свои. А узнать селекторы можно как раз в DevTools.

Чтобы изменить, например, размер точки слайдера, мы должны взять **в точности** такой же селектор.



Мы видим в DevTools селектор swiper-pagination-bullet, а у него — width и height. Значит, используя тот же селектор и свои размеры, мы сможем изменить точки слайдера.

```
.swiper-pagination-bullet {
  width: 12px;
  height: 12px;
}
```

```
.swiper-pagination-bullet {                    style.css:9
    width: 12px;
    height: 12px;
}
.swiper-pagination-bullet {                    swiper-bund….min.css:13
    width: 8px;
    height: 8px;
    display: inline-block;
    border-radius: ▶ 50%;
    background: ▶ #000;
    opacity: .2;
}
```

Теперь в DevTools мы видим, что наши стили перебили стили плагина и точки стали больше — как мы и планировали. Именно в таком ключе и стоит действовать далее, чтобы правильно сделать стилизацию плагина под себя.