СВОЙСТВО ANIMATION.

Общая информация. Настройка анимации. Ключевые кадры. Настройка нескольких значений.

Общая информация

CSS-анимации позволяют анимировать переходы от одной конфигурации CSS стилей к другой. CSS-анимации состоят из двух компонентов: стилевое описание анимации и набор ключевых кадров, определяющих начальное, конечное и, возможно, промежуточное состояние анимируемых стилей.

Есть три преимущества CSS-анимации перед традиционными способами:

- Простота использования для простых анимаций; вы можете создать анимацию, не зная JavaScript.
- Анимации будут хорошо работать даже при умеренных нагрузках системы. Простые анимации на JavaScript, если они плохо написаны, часто выполняются плохо.
- Позволяет браузеру контролировать последовательность анимации, тем самым оптимизируя производительность и эффективность браузера. Например, уменьшая частоту обновления кадров анимации в непросматриваемых в данный момент вкладках.

Настройка анимации

Чтобы создать CSS-анимацию вы должны добавить в стиль элемента, который хотите анимировать, свойство **animation** или его **подсвойства**. Это позволит вам настроить ускорение и продолжительность анимации, а также другие детали того, как должна проходить анимация.

CSS свойство **animation** это короткая запись для свойств: animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction, animation-fill-mode и animation-play-state.

Порядок важен в каждом определении анимации: первое значение, которое может быть определено как <time> присваивается **animation-duration**, а второе - **animation-delay**.

Ключевые кадры

После того, как вы настроили временн*ы*е свойства (продолжительность, ускорение) анимации, вы должны определить внешний вид анимации. Это делается с помощью указания двух и более ключевых кадров после **@keyframes**. Каждый кадр описывает, как должен выглядеть анимированный элемент в текущий момент.

В то время, как временные характеристики (продолжительность анимации) указываются в стилях для анимируемого элемента, ключевые кадры используют проценты, чтобы определить стадию протекания анимации. **0%** означает начало анимации, а **100%** её конец. Так как эти значения очень важны, то для них придумали специальные слова: **from** и **to**.

Вы также можете добавить ключевые кадры, характеризующие промежуточные состояния анимации.

Подсвойства Animation

animation-name

Определяет имя **@keyframes**, настраивающего кадры анимации.

animation-duration

Определяет время, в течение которого должен пройти один цикл анимации.

animation-timing-function

Настраивает функцию ускорения анимации. Значения такие же как и в transition.

- **ease** - значение по умолчанию, скорость увеличивается к середине перехода, замедляясь в конце.
- **linear** - переходы с неизменной скоростью.
- **ease-in** – переход начинается медленно, с увеличением скорости до полного завершения.
- **ease-out** - переход начинается быстро, с замедлением при завершении.
- **ease-in-out** - начинается медленно, ускоряется, а затем снова замедляется.
- **cubic-bezier(p1, p2, p3, p4)** - кубическая кривая Безье, где значения p1 и p3 должны находиться в диапазоне от 0 до 1.
- **steps(n, <jumpterm>)** - переход с **n** остановок во время перехода, отображая каждую остановку в течение равных промежутков времени. Например, если n равно 5, есть 5 шагов.
    - **jump-start | start** обозначает непрерывную функцию, так что первый скачок происходит, когда начинается переход;
    - **jump-end | end** обозначает непрерывную функцию, так что последний переход происходит, когда анимация заканчивается;
    - **jump-none** прыжка с обоих концов нет. Вместо этого, переход удерживается на отметке 0% и отметке 100%, в течение 1/n продолжительности перехода.
    - **jump-both** включает паузы на отметках 0% и 100%, эффективно добавляя шаг во время перехода.
    - **step-start** эквивалентно steps(1, jump-start)
    - **step-end** эквивалентно steps(1, jump-end)

animation-delay

Настраивает задержку между временем загрузки элемента и временем начала анимации.

animation-iteration-count

Определяет количество повторений анимации; вы можете использовать значение **infinite** для бесконечного повторения анимации.

animation-direction

Свойство задает направление анимации: вперед, в обратном порядке или поочередно.

- **normal** Каждый раз при повторе, анимация сбрасывается в начальное состояние и начинается заново. Это значение по умолчанию.
- **reverse** Каждый раз при повторе, анимация сбрасывается до конечного состояния и начинается заново. Шаги анимации выполняются в обратном порядке, и функции синхронизации также меняются местами. Например, временная функция ease-in становится ease-out.
- **alternate** Анимация меняет направление в каждом цикле, при этом первая итерация воспроизводится вперед. Счетчик для определения, является ли цикл четным или нечетным, начинается с единицы.
- **alternate-reverse** Анимация меняет направление в каждом цикле, при этом первая итерация воспроизводится в обратном направлении. Счетчик для определения, является ли цикл четным или нечетным, начинается с единицы.

animation-fill-mode

Настраивает момент применения стилей, используемых анимацией, до или после ее выполнения.

- **none** Стили, описанные в анимации, применяются к элементу **только** в процессе выполнения. Это значение по умолчанию.
- **forwards** Элемент сохранит стили, установленные последним ключевым кадром, обнаруженным во время выполнения анимации. Последний ключевой кадр зависит от значения **animation-direction** и **animation-iteration-count**
- **backwards** Анимация сразу применит к элементу стили, определенные в первом ключевом кадре. Первый ключевой кадр зависит от значения **animation-direction**.
- **both** Стили будут применены во все время выполнения анимации (даже при задержке animation-delay) и после ее завершения.

animation-play-state

Позволяет приостановить и возобновить анимацию. Возобновление приостановленной анимации запустит анимацию с того места, где она была остановлена во время приостановки, а не с начала последовательности анимации.

Свойство может принимать одно из двух значений: **running –** анимация запущена, **paused –** анимация приостановлена.

Установка нескольких значений свойств анимации

Полнотекстовые свойства CSS анимации могут принимать несколько значений, разделенных запятыми. Эту функцию можно использовать, если вы хотите применить несколько анимаций в одном правиле или установить разные длительности, количество итераций и т. д. для каждой из анимаций.

Например, в коде ниже каждой анимации будет присвоено значение продолжительности и количества итераций с той же позицией, что и имя анимации.

```
animation-name: scale, color, border;
animation-duration: 4s, 6s, 2s;
animation-iteration-count: 2, 1, infinite;
```

Анимации **scale** назначается продолжительность 4s и количество итераций 2, **color** – 6s и 1 повтор, а **border** анимации назначается продолжительность 2s и количество итераций будет бесконечным.

Во втором примере заданы три имени анимации, но есть только одна продолжительность и количество итераций. В этом случае всем трем анимациям присваивается **одинаковая** продолжительность и количество итераций.

```
animation-name: scale, color, border;
animation-duration: 4s;
animation-iteration-count: 2;
```

В этом третьем примере указаны три анимации, но только две продолжительности и количество итераций. В таких случаях, когда в списке недостаточно значений, чтобы назначить отдельное значение для каждой анимации, присвоение значения циклически повторяется от первого к последнему элементу в доступном списке, а затем возвращается к первому элементу.

```
animation-name: scale, color, border;
animation-duration: 4s, 2s;
animation-iteration-count: 2, 1;
```

Итак, **scale** и **border** получают продолжительность 4s, а **color** - 2s, которая является последним значением в списке значений продолжительности. Значения счетчика итераций (и любые другие указанные вами значения свойств) будут назначены таким же образом.

Если несоответствие количества анимаций и значений свойства анимации инвертировано, например, имеется пять значений задержки анимации и только три animation-name, то **лишние** значения не применяются ни к какой анимации и игнорируются.

ЗАДАНИЕ № 22 СЛАЙДЕРЫ

Разберем использование анимации и трансформаций на примере страницы Галерея.

Перед началом работы изучите файл **gallery.html** в папке **sample**. Закомментируйте предыдущие настройки страницы в файле **style.css**.

Каждый из наших слайдеров будет находиться в своем контейнере и заменит один из разделов Галереи.

Слайдер с автоматической сменой изображений

Первым будем создавать слайдер с автоматической сменой изображений по времени. Он заменит нам первый раздел галереи Весенние пейзажи.

1. В стилях опишем класс-обертку **.container_slider**.

```
div.container_slider {
    margin: 50px auto;
    width: 680px;
    height: 340px;
    overflow: hidden;
    position: relative;
  }
```

Жесткие размеры, скрытие содержимого за его пределами и относительное позиционирование позволят управлять содержимым контейнера и отображать его нужным нам образом.

2. Добавим классы изображениям и избавимся от гиперссылок (адреса изображений и содержимое alt расставьте самостоятельно).

```
<div class="col-12">Весенние пейзажи</div>
  <div class="container_slider time">
```

```
    <img class="time_slider" src=".../1.jpg" alt="...">
    <img class="time_slider" src=".../2.jpg" alt="...">
    <img class="time_slider" src=".../3.jpg" alt="...">
    <img class="time_slider" src=".../4.jpg" alt="...">
    <img class="time_slider" src=".../5.jpg" alt="...">
    <img class="time_slider" src=".../6.jpg" alt="...">
    <img class="time_slider" src=".../7.jpg" alt="...">
    <img class="time_slider" src=".../8.jpg" alt="...">
</div>
```

Смена изображений будет осуществляться с помощью изменения прозрачности с задержкой. Т.е. все изображения загружаются на страницу одновременно, но проявляются через заданные (для каждого изображения свои) промежутки времени. Реализуем этот механизм с помощью свойства **animation**, ключевых кадров и псевдоклассов. Но обо всём по порядку.

3. Настроим изображения в слайдере. При загрузке страницы все изображения на странице не видны. Длительность цикла анимации зависит от количества изображений. В нашем случае их 8, видимым каждое из них будет 4 секунды, итого – 32 секунды.

```
div.container_slider .time_slider {
  /* изображения находятся одно под другим */
  position: absolute;
  /* имя keyframe:round, цикл 32 секунды, бесконечный цикл */
  animation: round 32s infinite;
  /* изображения не видны */
  opacity: 0;
  width: 100%;
}
```

После настройки временных свойств (продолжительность, ускорение) анимации, необходимо определить ее внешний вид. Это делается с помощью двух и более ключевых кадров после @keyframes. Каждый кадр описывает, как должен выглядеть анимированный элемент в текущий момент.

4. Для нашей анимации опишем состояние изображений в 2 момента времени. На 1/8 от общего времени анимации объект станет видимым.

```
@keyframes round {
  12.5% { opacity: 1; }
  20% { opacity: 0; }
}
```

5. Далее настроим задержку анимации для каждого изображения в слайдере. Используем для этого псевдоклассы с номером элемента и свойство animation-delay.

```
div.container_slider img:nth-child(1) {animation-delay: 28s;}
div.container_slider img:nth-child(2) {animation-delay: 24s;}
div.container_slider img:nth-child(3) {animation-delay: 20s;}
div.container_slider img:nth-child(4) {animation-delay: 16s;}
div.container_slider img:nth-child(5) {animation-delay: 12s;}
div.container_slider img:nth-child(6) {animation-delay: 8s;}
div.container_slider img:nth-child(7) {animation-delay: 4s;}
div.container_slider img:nth-child(8) {animation-delay: 0s;}
```

Слайдер со сменой изображений при клике на радиокнопку

Второй слайдер сложнее по количеству кода. Технически это даже не слайдер, потому что самостоятельно он менять фото не может. Механизм смены фото основан на перемещении фото при кликах пользователя на радиокнопки.

1. Создадим обертку для контейнера и разместим внутри нее радиокнопки с подписями.

- Количество радиокнопок равно количеству фотографий.
- У каждой радиокнопки есть свой уникальный идентификатор.
- Каждая кнопка связана со своей подписью с помощью атрибута **for**. Благодаря этому кнопка будет становиться активной не только при нажатии непосредственно на нее, но и на соответствующую ей надпись

```
<div class="container_btn">
    <input type="radio" name="switch" id="btn1" checked>
    <input type="radio" name="switch" id="btn2">
    <input type="radio" name="switch" id="btn3">
    <input type="radio" name="switch" id="btn4">
    <input type="radio" name="switch" id="btn5">
    <input type="radio" name="switch" id="btn6">
    <input type="radio" name="switch" id="btn7">
    <input type="radio" name="switch" id="btn8">

    <div class="switch">
      <label for="btn1"></label>
      <label for="btn2"></label>
      <label for="btn3"></label>
      <label for="btn4"></label>
      <label for="btn5"></label>
      <label for="btn6"></label>
      <label for="btn7"></label>
      <label for="btn8"></label>
    </div>

</div>
```

2. Настроим оформление для контейнера, расположение подписей и скроем значки радиокнопок.

```
div.container_btn {
    position: relative;
    width: 680px;
    margin: 20px auto 80px;
  }

  /* скрываем стандартные радиокнопки */
  div.container_btn input[name="switch"] {
    display: none;
  }

  /* позиционируем блок с подписями к радиокнопкам под фото */
  div.container_btn .switch {
    position: absolute;
    left: 0;
    bottom: -40px;
    text-align: center;
    width: 100%;
  }
```

3. К сожалению, внешний вид радиокнопок настроить нельзя, поэтому будем оформлять подписи. Текста внутри не будет, но зададим подписи границу, заливку и закруглим края.

```
div.container_btn .switch label {
    display: inline-block;
    width: 16px;
    height: 16px;
    cursor: pointer;
    margin: 0 3px;
    border-radius: 50%;
    border: 5px solid #2f363c;
    background-color: #738290;
}
```

4. Для того чтобы выделить текущий переключатель воспользуемся псевдоклассом **checked** и зададим белую заливку элементам **label**, с соответствующим **for** в соседнем с выбранной радиокнопкой блоке **switch**.

```
div.container_btn #btn1:checked~.switch label[for="btn1"],
div.container_btn #btn2:checked~.switch label[for="btn2"],
div.container_btn #btn3:checked~.switch label[for="btn3"],
div.container_btn #btn4:checked~.switch label[for="btn4"],
div.container_btn #btn5:checked~.switch label[for="btn5"],
div.container_btn #btn6:checked~.switch label[for="btn6"],
div.container_btn #btn7:checked~.switch label[for="btn7"],
div.container_btn #btn8:checked~.switch label[for="btn8"]
{
  background-color: white;
}
```

5. Добавим блок с фотографиями. Разместим его внутри контейнера .container_btn после блока с подписями радиокнопок (адреса изображений и содержимое alt расставьте самостоятельно).

```
<div class="slider-inner">
 <div class="slides">
  <img src=".../1.jpg" alt="...">
  <img src=".../2.jpg" alt="...">
  <img src=".../3.jpg" alt="...">
  <img src=".../4.jpg" alt="...">
  <img src=".../5.jpg" alt="...">
  <img src=".../6.jpg" alt="...">
  <img src=".../7.jpg" alt="...">
  <img src=".../8.jpg" alt="...">
 </div>
</div>
```

6. Настроим внешний вид блока с фото – размеры, перекрытие и длительность анимации.

```
div.container_btn .slider-inner {
    overflow: hidden;
}

div.container_btn .slides {
    width: 800%;
    transition: all 0.5s;
}

div.container_btn .slides img {
    width: 680px;
    float: left;
}
```

Размеры блока с фото 800% - по 100% на каждую фотографию. НО видимая часть всего 680px - по ширине одного изображения.

12. Следующим шагом настроим перемещение картинок при выборе кнопки. При выборе того или иного переключателя наш блок шириной 800% будет смещаться влево на ширину одного или нескольких изображений, таким образом создавая эффект слайдера.

```
div.container_btn #btn1:checked ~ .slider-inner .slides {
    transform: translate(0);
}
div.container_btn #btn2:checked ~ .slider-inner .slides {
    transform: translate(-680px);
}
div.container_btn #btn3:checked ~ .slider-inner .slides {
    transform: translate(-1360px);
}
div.container_btn #btn4:checked ~ .slider-inner .slides {
    transform: translate(-2040px);
}
div.container_btn #btn5:checked ~ .slider-inner .slides {
```

```
    transform: translate(-2720px);
}
div.container_btn #btn6:checked ~ .slider-inner .slides {
    transform: translate(-3400px);
}
div.container_btn #btn7:checked ~ .slider-inner .slides {
    transform: translate(-4080px);
}
div.container_btn #btn8:checked ~ .slider-inner .slides {
    transform: translate(-4760px);
}
```

Слайдер со сменой изображений при клике на стрелку

Третий слайдер еще более объемный в части кода. Смена слайдов осуществляется за счет включения и отключения видимости изображений при нажатии на стрелки.

Структура блока будет состоять из нескольких составляющих:

- Радиокнопки для реализации смены состояний.
- Блоки с фото.
- Блоки со стрелками Налево и Направо для каждого фото.

1. Разместим в блоке с классом .container_arrow необходимые элементы (адреса изображений и содержимое alt расставьте самостоятельно).

```
<div class="container_arrow">

 <input type="radio" class="slide_img" name="slide_img" id="img_1"
checked>
 <input type="radio" class="slide_img" name="slide_img" id="img_2">
 <input type="radio" class="slide_img" name="slide_img" id="img_3">
 <input type="radio" class="slide_img" name="slide_img" id="img_4">
```

```
 <input type="radio" class="slide_img" name="slide_img" id="img_5">
 <input type="radio" class="slide_img" name="slide_img" id="img_6">
 <input type="radio" class="slide_img" name="slide_img" id="img_7">
 <input type="radio" class="slide_img" name="slide_img" id="img_8">

 <div class="slides" id="i_1"><img src=".../1.jpg" alt="..."></div>
 <div class="slides" id="i_2"><img src=".../2.jpg" alt="..."></div>
 <div class="slides" id="i_3"><img src=".../3.jpg" alt="..."></div>
 <div class="slides" id="i_4"><img src=".../4.jpg" alt="..."></div>
 <div class="slides" id="i_5"><img src=".../5.jpg" alt="..."></div>
 <div class="slides" id="i_6"><img src=".../6.jpg" alt="..."></div>
 <div class="slides" id="i_7"><img src=".../7.jpg" alt="..."></div>
 <div class="slides" id="i_8"><img src=".../8.jpg" alt="..."></div>
</div>
```

2. Чтобы в следующем блоке появились стрелки, добавим в раздел head ссылку на особый шрифт, а в label специальные блоки для их отображения

```
    <link rel="stylesheet" href="style/fonts/fa/css/font-
awesome.css">
```

```
<div id="arrow_next">
 <label for="img_1">
  <i class="fa fa-angle-right" aria-hidden="true"></i>
 </label>

 <label for="img_2"><i class="fa fa-angle-right" aria-
hidden="true"></i></label>
     ...
 <label for="img_8"><i class="fa fa-angle-right" aria-
hidden="true"></i></label>
</div>

<div id="arrow_prev">
 <label for="img_1">
  <i class="fa fa-angle-left" aria-hidden="true"></i>
 </label>
```

```
 <label for="img_2"><i class="fa fa-angle-left" aria-
hidden="true"></i></label>
     ...
 <label for="img_8"><i class="fa fa-angle-left" aria-
hidden="true"></i></label>
</div>
```

3. Перейдем к файлу стилей и сначала запишем общие настройки для контейнера.

```
div.container_arrow {
    position: relative;
    margin: 20px auto 80px;
    width: 680px;
    height: 480px;
}

div.container_arrow .slides {
    position: absolute;
    width: 100%;
    overflow: hidden;
    visibility: hidden;
    opacity: 0;      /*блоки с фото по умолчанию не видны на
странице*/
    transition: .5s; /*смена состояний будет происходить с
задержкой*/
}

div.container_arrow .slides img {
    width: 100%;
}

div.container_arrow .slide_img { /*радиокнопки скрыты*/
    display: none;
}
```

4. Настроим отображение стрелок.

```
div.container_arrow #arrow_next label, #arrow_prev label {
    color: rgba(0, 0, 0, 0.7);
    position: absolute;
    visibility: hidden; /*подписи скрыты*/
    cursor: pointer;
    top: 34%;
    font-size: 80px;
}

/*смещаем стрелки за пределы изображения*/
div.container_arrow #arrow_prev label {left: -50px;}
div.container_arrow #arrow_next label {right: -50px;}
```

5. Теперь займемся настройкой смены изображений.

```
/* текущие настройки будут применяться к изображениям при выборе
соответствующей радиокнопки, они будут становиться видимыми и
непрозрачными */
div.container_arrow #img_1:checked ~ #i_1,
div.container_arrow #img_2:checked ~ #i_2,
div.container_arrow #img_3:checked ~ #i_3,
div.container_arrow #img_4:checked ~ #i_4,
div.container_arrow #img_5:checked ~ #i_5,
div.container_arrow #img_6:checked ~ #i_6,
div.container_arrow #img_7:checked ~ #i_7,
div.container_arrow #img_8:checked ~ #i_8
{
    visibility: visible;
    opacity: 1;
}
```

6. Настроим отображение стрелок для каждого изображения. Например, для изображения с id=i_3 должны быть видимыми четвертая стрелка из списка label внутри блока #arrow_next и вторая из группы #arrow_prev.

```
div.container_arrow #img_1:checked ~ #arrow_next label:nth-child(2),
div.container_arrow #img_2:checked ~ #arrow_next label:nth-child(3),
div.container_arrow #img_3:checked ~ #arrow_next label:nth-child(4),
div.container_arrow #img_4:checked ~ #arrow_next label:nth-child(5),
div.container_arrow #img_5:checked ~ #arrow_next label:nth-child(6),
div.container_arrow #img_6:checked ~ #arrow_next label:nth-child(7),
div.container_arrow #img_7:checked ~ #arrow_next label:nth-child(8),
div.container_arrow #img_8:checked ~ #arrow_next label:nth-child(1),
div.container_arrow #img_1:checked ~ #arrow_prev label:nth-child(8),
div.container_arrow #img_2:checked ~ #arrow_prev label:nth-child(1),
div.container_arrow #img_3:checked ~ #arrow_prev label:nth-child(2),
div.container_arrow #img_4:checked ~ #arrow_prev label:nth-child(3),
div.container_arrow #img_5:checked ~ #arrow_prev label:nth-child(4),
div.container_arrow #img_6:checked ~ #arrow_prev label:nth-child(5),
div.container_arrow #img_7:checked ~ #arrow_prev label:nth-child(6),
div.container_arrow #img_8:checked ~ #arrow_prev label:nth-child(7) {
    visibility: visible;
}
```

На самом деле, создание слайдеров на чистом css дело достаточно сложное и на практике мало применимое. Особую сложность представляют слайдеры, связанные с позиционированием элементов. В них при добавлении нового изображения в слайдер или при изменении размеров придется переписывать добрую половину кода и как бонус заниматься дополнительными расчетами.

Нашей задачей в этой работе было изучить подходы к организации слайдеров в целом.

В качестве самостоятельной работы можете попробовать настроить получившиеся слайдеры: изменить количество изображений, их размеры, стиль маркеров и т.д.