8. Таймеры, анимация. jQuery

Практика

Статьи

Timeouts and intervals - MDN
Планирование: setTimeout и setInterval

https://jquery.com/
Как создать анимацию в JavaScript за 30 минут

Вебинар: Делаем сайт более живым с помощью анимаций

Таймеры setTimeout и setInterval

[WindowTimers.setTimeout() - MDN]
[Timeouts and intervals - MDN]
[Планирование: setTimeout и setInterval]

«Планирование вызова» функции через методы:

- **setTimeout** позволяет вызвать функцию один раз через определённый интервал времени.

- **setInterval** позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени.

setTimeout

```
let timerId = setTimeout(func|code, [delay], [arg1], [arg2], ...)
```

func|code - Функция или строка кода для выполнения. Обычно это функция. По историческим причинам можно передать и строку кода, но это не рекомендуется. **Функция передается, а не вызывается!**

delay - Задержка перед запуском в миллисекундах (1000 мс = 1 с). Значение по умолчанию – 0 = запустить как можно скорее

arg1, arg2… - Аргументы, передаваемые в функцию, опциональные

```
function sayHi(phrase, who) {
  alert( phrase + ', ' + who );
}
setTimeout(sayHi, 1000, "Привет", "Джон"); // Привет, Джон через 1 секунду
```

Вызов **setTimeout** возвращает «идентификатор таймера» **timerId**, который можно использовать для отмены дальнейшего выполнения:

```
let timerId = setTimeout(...);
clearTimeout(timerId);
```

clearTimeout используется для отмены **setTimeout,** в тч истечения указанного времени таймера.

Задача: Показывать модальное окно со скидкой через 3 секунды после открытия страницы

---

setInterval

```
let timerId = setInterval(func|code, [delay], [arg1], [arg2], ...)
```

Отличие этого метода от setTimeout в том, что функция запускается не один раз, а периодически через указанный интервал времени.

setInterval() продолжает выполнять задачу вечно, если с ней что-то не сделать. Чтобы остановить дальнейшее выполнение функции, необходимо вызвать **clearInterval(timerId)**.

```
let myInterval = setInterval(myFunction, 2000);
clearInterval(myInterval);
```

Пример: показывать текущее время на странице

```
<p class="clock"></p>

function displayTime() {
      let date = new Date();
      let time = date.toLocaleTimeString();
      document.querySelector('.clock').textContent = time;
    }

    displayTime();
    const createClock = setInterval(displayTime, 1000);
```

Анимация

CSS позволяет создавать простые анимации без использования JavaScript.

JavaScript может быть использован для управления такими CSS-анимациями. Это позволяет делать более сложные анимации, используя небольшие кусочки кода.

Вебинар: [Делаем сайт более живым с помощью анимаций]

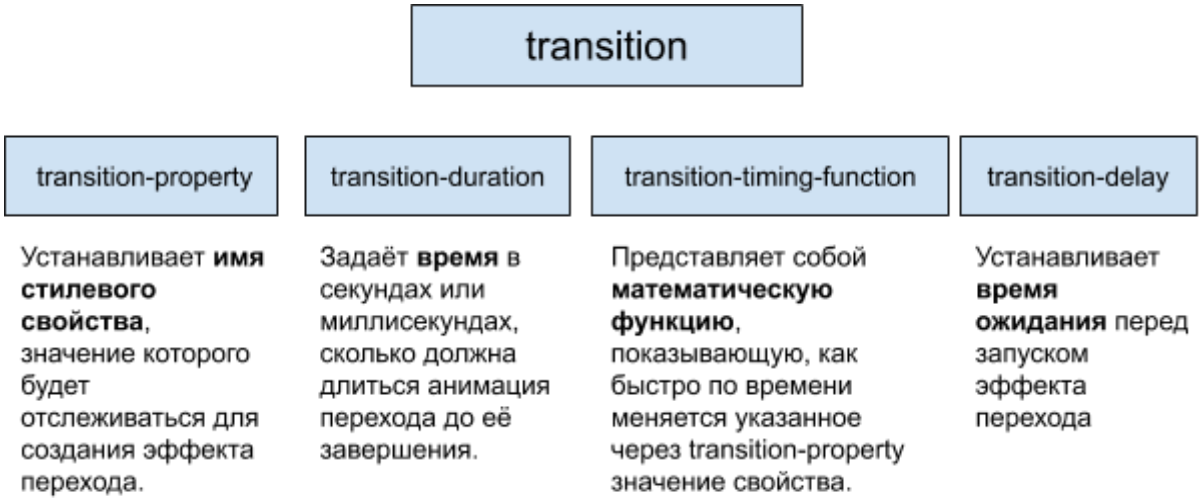
Большинство анимаций может быть реализовано с использованием CSS

CSS-переходы

[CSS-анимации]

Идея CSS-переходов проста: мы указываем, что некоторое свойство должно быть анимировано, и как оно должно быть анимировано. А когда свойство меняется, браузер сам обработает это изменение и отрисует анимацию.

Всё что нам нужно, чтобы начать анимацию – это изменить свойство, а дальше браузер сделает плавный переход сам.




```
.animated {
  transition-property: background-color;
  transition-duration: 3s;
}
```

Пример “Галерея”: https://codepen.io/lipa88/pen/VwaBMaz

При помощи JS можно анимировать элементы по клику:

```
<button class="growing">Нажми меня</button>
```

```
.growing {
  transition: font-size 3s, color 2s;
}
```

```
growing.onclick = function() {
  this.style.fontSize = '36px';
  this.style.color = 'red';
};
```

Или добавлять класс с переходом при каком-то событии:

```
element.classList.add('animate');
```

Событие transitionend

Когда завершается анимация, срабатывает событие transitionend.

Оно широко используется для выполнения действий после завершения анимации, а также для создания последовательности анимаций.

Объект события **transitionend** содержит ряд полезных свойства:

event.propertyName
Имя свойства, анимация которого завершилась. Может быть полезным, если анимировать несколько свойств.

event.elapsedTime
Время (в секундах), которое заняла анимация, без учёта transition-delay.

Большинство анимаций может быть реализовано с использованием CSS, а событие **transitionend** позволяет запускать JavaScript после анимации

Пример: по окончанию анимации скрыть элемент

```
<div class="block"></div>
<button class="toggle-btn">close</button>

.block {
  width: 50px;  height: 50px;  background-color: green;  opacity: 1;
}
.anim-hide {
  opacity: 0;  transition: 2s;
}

btn.addEventListener("click", function () {
  let btnCloseHandler = function () {
    div.style.display = "none";
    div.classList.remove("anim-hide");
    div.removeEventListener("transitionend", btnCloseHandler);
  };

  div.classList.add("anim-hide");
  div.addEventListener("transitionend", btnCloseHandler);
});
```

Покадровая анимация CSS

Более сложная покадровая анимация на веб-странице делается с помощью двух вещей.

1. Продолжительность анимации, задержка перед её выполнением, число повторений и другие параметры указываются через универсальное свойство **animation**.

2. А ключевые кадры и значения свойств элемента определяются правилом **@keyframes**. Затем они связываются между собой с помощью переменной, имя для которой мы придумываем сами.

Свойство animation имеет восемь значений, которые могут быть заданы непосредственно через него или через отдельные свойства, вроде animation-duration, оно устанавливает продолжительность анимации. Подробнее об этих свойствах смотрите в справочнике.

Пример: https://codepen.io/lipa88/pen/rNerZvb
Облака: https://codepen.io/lipa88/pen/VwaBrKq
Градиентный фон: https://codepen.io/lipa88/pen/MWyBOJm

Анимация JS

С помощью JavaScript-анимаций можно делать вещи, которые нельзя реализовать на CSS. Например, движение по сложному пути.

Для этого можно использовать таймеры, например:

```
let start = Date.now();
    let timer = setInterval(function() {
      let timePassed = Date.now() - start;
      element.style.left = timePassed / 5 + 'px';
      if (timePassed > 2000) clearInterval(timer);
    }, 20);
```

Пример перемещения при клике на элемент

requestAnimationFrame ()

requestAnimationFrame() - это специализированная функция цикла, созданная для эффективного запуска множества анимации в браузере.

Она выполняет указанный блок кода до того, как браузер перерисовывает изображение, позволяя запускать анимацию с подходящей частотой кадров независимо от среды, в которой она выполняется.

```
let requestId = requestAnimationFrame(callback)
```

Такой вызов планирует запуск функции callback на ближайшее время, когда браузер сочтет возможным осуществить анимацию.

Пример: плавное появление и скрытие элемента при клике на кнопку

Если в **callback** происходит изменение элемента, тогда оно будет сгруппировано с другими requestAnimationFrame и CSS-анимациями. Таким образом браузер выполнит один геометрический пересчёт и отрисовку, вместо нескольких.

Значение **requestId** может быть использовано для отмены анимации:

```
// отмена запланированного запуска callback
cancelAnimationFrame(requestId);
```

Дополнительно: Более сложная функция анимации

```
function animate({timing, draw, duration}) {

  let start = performance.now();

  requestAnimationFrame(function animate(time) {
    // timeFraction изменяется от 0 до 1
    let timeFraction = (time - start) / duration;
    if (timeFraction > 1) timeFraction = 1;

    // вычисление текущего состояния анимации
    let progress = timing(timeFraction);

    draw(progress); // отрисовать её

    if (timeFraction < 1) {
      requestAnimationFrame(animate);
    }

  });
}
```

Функция animate имеет три аргумента, которые описывают анимацию:

duration - Продолжительность анимации.

timing(timeFraction) - Функция расчёта времени, которая будет вычислять прогресс анимации в зависимости от прошедшего времени (0 в начале, 1 в конце).

draw(progress) - Функция отрисовки, которая получает аргументом значение прогресса анимации и отрисовывает его. Значение progress=0 означает, что анимация находится в начале, и значение progress=1 – в конце. Эта та функция, которая на самом деле и рисует анимацию.

В отличие от CSS-анимаций, можно создать любую функцию расчёта времени и любую функцию отрисовки.

---

jQuery

[Основы jQuery]

Библиотека jQuery предоставляет функцию jQuery, которая позволяет вам выбирать элементы с помощью селекторов CSS

При вызове функции **$()** и передаче в неё селектора вы создаете новый объект jQuery.

```
var listItems = jQuery( 'li' );
```

или

```
var listItems = $( 'li' );
```

Перед безопасным использованием jQuery для выполнения чего-либо на странице, нужно убедиться, что страница готова к манипуляциям.
Для этого нужно поместить наш код в функцию, а затем эту функцию в **$(document).ready()**

```
$( document ).ready(function() {
  console.log( 'готов!' );
});
```

Создание новых элементов

```
$( '<p>Привет!</p>' ); // создание нового элемента <p> с текстом
```

Изменение стиля

```
padding = 30;
$( elem ).css( 'padding-left', padding + 'px' );
```

```
$( 'li' ).eq( 1 ).css({
  'font-size': '20px',
  'padding-left': '20px'
});
```

Добавление и удаление классов

```
$( 'li' ).addClass( 'hidden' );
$( 'li' ).removeClass( 'hidden' );
$( 'li' ).toggleClass( 'hidden' );
```

Подробнее о других методах можно посмотреть тут: [Обход и манипуляция]

События

jQuery упрощает реагирование на действия пользователя с веб-страницей.

```
$( 'li' ).on( 'click', function( event ) {
  console.log( 'clicked', $( this ).text() );
});
```

Так же есть сокращенные методы событий:

```
$( 'li' ).click();
```

.click() .keydown() .mouseover() .scroll() .focus()

Или с именованной функцией:

```
var handleClick = function() {
  console.log( 'на что-то щёлкнули' );
};

$( 'li' ).on( 'click', handleClick );
 $( 'h1' ).on( 'click', handleClick );
```

Отвязать событие можно с помощью метода **.off()**

```
$( 'li' ).off( 'click' );
```

Анимация с jQuery

Часто используемые эффекты встроены в jQuery как методы, которые вы можете вызвать для любого объекта jQuery:

.show() — показать выбранные элементы;
.hide() — скрыть выбранные элементы;
.fadeIn() — анимация прозрачности выбранных элементов до 0%;
.fadeOut() — анимация прозрачности выбранных элементов до 100%;
.slideDown() — отображение выбранных элементов с помощью вертикального скользящего движения;
.slideUp() — сокрытие выбранные элементы с помощью вертикального скользящего движения;
.slideToggle() — показать или скрыть выбранные элементы с вертикальным скользящим движением в зависимости от того, видны элементы в данный момент или нет.

Вы также можете указать длительность встроенных эффектов:
`$( '.hidden' ).show( 300 );`

Пример применения функций

Если требуется сделать что-то после завершения анимации

```
$( 'p.old' ).fadeOut( 300, function() {
  $( this ).remove(); //удалить
});
```

this указывает на исходный элемент DOM, к которому применялся эффект

Смотрите более подробную информацию об эффектах jQuery в документации.

Задача: аккордеон на jQuery
Пример и пример

Произвольные эффекты .animate()

У метода .animate() есть три аргумента:

- объект, определяющий свойства для анимации;
- продолжительность анимации в миллисекундах;
- функция обратного вызова, которая будет вызываться после окончания анимации.

Метод **.animate()** может анимировать до указанного конечного значения или увеличить существующее значение.

```
$( '.funtimes' ).animate({
   left: '+=50', // увеличить на 50
   opacity: 0.25,
   fontSize: '12px'
 },
 300,
 function() {
   // выполняется, когда анимация завершена
 }
);
```

Также есть два важных метода для управления анимацией:

- .**stop()** — останавливает выполняемую в данное время анимацию для выбранных элементов.

- **.delay()** — пауза перед выполнением следующей анимации. В качестве аргумента передаётся желаемое время ожидания в миллисекундах.

jQuery также предлагает методы для управления очередью эффектов, создание произвольных очередей и добавление своих функций в эти очереди.

Задача: модальное окно на jQuery с анимацией появления
Пример

Практика

Авто-закрытие модального окна JS

Для обычного модального окна допишите его автозакрытие через 5 секунд после того, как окно было открыто пользователем.

Секундомер JS

Нужно отображать время, на странице, а так же добавить:

- Кнопка «Пуск» для запуска секундомера.
- Кнопка «Стоп», чтобы приостановить / остановить его.
- Кнопка «Сброс» для сброса времени на 0.
- Отображение времени, показывающее количество прошедших секунд, а не фактическое время.

Для этого потребуется создать переменную, которая начинается с 0, а затем увеличивается на единицу каждую секунду с использованием постоянного цикла.

Используй **startTime = Date.now()** чтобы получить метку времени, когда именно пользователь нажал кнопку запуска, а затем **Date.now() - startTime**, чтобы получить количество миллисекунд после нажатия кнопки запуска.

Показывайте результат в часах, минутах и секундах. Количество секунд в часе 3600. Количество минут - это количество секунд, оставшихся после удаления всех часов, разделенное на 60. Количество секунд будет количеством секунд, оставшихся после удаления всех минут.

Чтобы остановить секундомер, нужно очистить интервал. Для этого нужно вернуть счетчик в 0, очистить интервал, а затем немедленно обновить отображение времени.

Также следует отключить кнопку запуска после ее нажатия и снова включить ее после того, как вы остановили / сбросили ее. В противном случае многократное нажатие кнопки запуска приведет к применению нескольких setInterval(), что приведет к неправильному поведению.

Пример

Вывод каждую секунду JS

Напишите функцию **printNumbers(from, to)**, которая выводит число каждую секунду, начиная от from и заканчивая to.

Фотогалерея с увеличением фото при клике JS

При нажатии картинка изменяет размеры с 40x24px до 400x240px (увеличивается в 10 раз).
Время анимации 3 секунды.
По окончании анимации добавить картинке тень.
Если во время анимации будут дополнительные клики по картинке – они не должны ничего «сломать».

Еще одна галерея на jQuery

Создать фотогалерею с эффектом при клике на изображение

Вкладки на jQuery

Переписать имеющиеся вкладки при помощи jQuery

Аккордеон на jQuery

Переписать имеющийся аккордеон при помощи jQuery
