Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

Инструкция
по выполнению лабораторной работы
«Применение функций временной задержки. Анимации в JavaScript»

Минск
2021

Лабораторная работа № 28

Тема работы: Применение функций временной задержки. Анимации в JavaScript

1. Цель работы

Формирование умений применения функций временной задержки для создания анимационных эффектов средствами JavaScript.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

Одной из наиболее типичных областей применения CSS является воспроизведение визуальных анимационных эффектов. Реализовать их можно с помощью метода setTimeout() или setInterval(), используя их для организации многократных вызовов функции, изменяющей встроенный стиль элемента.

Функция setTimeout имеет следующий синтаксис:

```
let timeoutID = window.setTimeout(func, [delay, param1, param2, ...]);
let timeoutID = window.setTimeout(code, [delay]);
```

где:

timeoutID – числовой идентификатор, который может использоваться функцией clearTimeout() для отмены таймера;

func – функция, которая будет выполнена. Ссылка на функцию указывается без оператора вызова «()», иначе в setTimeout будет передан результат выполнения функции;

code (альтернативный синтаксис) – блок кода, который будет выполнен. Код в виде строки передавать не рекомендуется, т.к. такой способ затрудняет чтение (а также сопровождение и/или отладку), внутри используется eval(), что является небезопасным и, наконец, этот способ медленнее остальных, т.к. вызывает интерпретатор JS;

delay – задержка в миллисекундах (1000 мс = 1 с), по истечении которой будет выполнена функция. По умолчанию равна 0;

param1, param2 – параметры, которые будут переданы в качестве аргументов указанной функции.

Функция setTimeout возвращает числовой **id**, который может использоваться функцией **clearTimeout()** для отмены таймера.

Функция **setInterval()** неоднократно вызывает функцию или выполняет переданный код с указанной временной задержкой между каждым вызовом. Функция будет вызываться на исполнение до тех пор, пока не будет закрыто окно с веб-страницей или вызвана функция **clearInterval()**, прерывающая работу **setInterval()**.

Метод setInterval имеет такой же синтаксис как setTimeout:

```
let timernum = setInterval(func[, delay, param1, param2, ...]);
let timernum = setInterval(code[, delay]);
```

Все аргументы имеют такое же значение. Но отличие этого метода от setTimeout в том, что функция запускается не один раз, а периодически через указанный интервал времени.

```
let timernum = setInterval(() => alert('tick-takc'), 5000);
```

Можно отменить таймер после выполнения нескольких итераций.

```
let timernum = setInterval(() => alert('tick-takc'), 1000);
setTimeout(() => { clearInterval(timernum); }, 5000);
```

Помимо **setInterval** регулярный запуск можно реализовать с помощью, так называемого, рекурсивного **setTimeout.**

Рекурсивный **setTimeout** – более гибкий метод, чем **setInterval**. С его помощью последующий вызов может быть задан по-разному в зависимости от результатов предыдущего.

```
<div id ="container">
  <div id ="animate">Моя анимация будет здесь</div>
</div>
#container {
  width: 400px;
  height: 400px;
  position: relative;
  background: yellow;
}
#animate {
  width: 50px;
  height: 50px;
  position: absolute;
  background: red;
}

function myMove() {
    let elem =  document.getElementById("animate");
    let pos = 0;
    let id = setInterval(frame, 5);
    function frame() {
        if (pos == 350) {
             clearInterval(id);
        } else {
             pos++;
             elem.style.top = pos + 'px';
             elem.style.left = pos + 'px';
        }
     }
}
```

Однако, при использовании таймеров для создания анимации, для каждого элемента на странице приходится писать свой таймер. Кроме того, setInterval всегда будет рисовать с одной скоростью, независимо от того, какой у пользователя компьютер, что делает пользователь и активна ли текущая страница.

В HTML5 имеется встроенный API, который используется при создании анимаций на странице – функция **window.requestAnimationFrame** указывает браузеру на то, что необходимо произвести анимацию, и просит его запланировать перерисовку на следующем кадре анимации. В качестве параметра метод получает функцию, которая будет вызвана перед перерисовкой.

```
window.requestAnimationFrame(callback);
```

Объект requestAnimationFrame исключает возможность ненужных отрисовок и может связывать вместе несколько анимаций в одно целое и цикл перерисовки, что делает анимацию плавной и эффективной в плане затрат. Если текущая вкладка браузера перестает быть в фокусе, requestAnimationFrame перестанет выполнять операции по анимации. Еще одно преиму-

щество использования requestAnimationFrame – то, что он работает с частотой развертки изображения на мониторе – параметром, который определяет, сколько времени требуется монитору на то, чтобы отобразить новый кадр.

```
<!DOCTYPE html>
<html>
<head>
<style>
    #myball {

    position: absolute;
    width: 100px;
    height: 100px;
    border-radius: 50%;
    background: linear-gradient(45deg, red, green);
}
#mybal {

    position: absolute;
    width: 100px;
    height: 100px;
    border-radius: 50%;
    background: linear-gradient(45deg, rgb(0, 68, 255), rgb(245, 241, 11));
}
    </style>
</head>
<body>
  <div id="myball"></div>
  <div id="mybal"></div>
<script>
  let el=document.getElementById('myball');
let elem=document.getElementById('mybal');
pos=0;
sop=0;
function round() {
pos+=10;
sop-=10;
w=document.documentElement.clientWidth-document.getElementById('mybal').offsetWidth;;
if (pos>=w) {
    pos=0;
}
el.style.transform=`rotate(${pos}deg)`;
el.style.left=pos+'px';
elem.style.transform=`rotate(${sop}deg)`;
requestAnimationFrame(round);
}
requestAnimationFrame(round);
</script>
</body>
</html>
```

Чтобы анимация не остановилась, callback метод сам должен вызвать requestAnimationFrame().Вызов необходимо осуществлять всякий раз, когда необходимо обновить анимацию на экране, чтобы запросить планирование анимации. Обычно запросы происходят 60 раз в секунду, но чаще всего совпадают с частотой обновления экрана.

Для отмены анимаци предназначен метод cancelAnimationFrame. Этот метод должен принять requestID для кадра, который необходимо отменить.

```
let num=requestAnimationFrame(round);
cancelAnimationFrame(num);
```

В случае, если в браузер requestAnimationFrame не поддерживает, можно воспользоваться версией с префиксом. Например, Chrome использует webkitRequestAnimationFrame, а

Mozilla поддерживает mozRequestAnimationFrame. Чтобы исправить это, можно воспользоваться скриптом Пола Ириша. Он просто соединяет разные варианты в новой функции: requestAnimFrame.

```
// setTimeout в качестве запасного варианта
window.requestAnimFrame = (function(){
  return  window.requestAnimationFrame       ||
          window.webkitRequestAnimationFrame ||
          window.mozRequestAnimationFrame    ||
          window.oRequestAnimationFrame      ||
          window.msRequestAnimationFrame     ||
          function( callback ){
            window.setTimeout(callback, 1000 / 60);
          };
})();
```

5. Порядок выполнения работы

1. Добавить на веб-страницу элемент <figure>, содержащий название картинки. В папку pictures сохранить 5 различных изображений. Реализовать смену изображений (по очереди из имеющихся пяти) в соответствии с интервалом, указанным на соответствующей кнопке (3, 5, 10)сек.

2. Реализуйте визуальные анимационные эффекты в соответствии с вариантом, таблица 28.1.

Таблица 28.1. – Варианты заданий

Вариант	Задание
Вариант 1	Добавьте на веб-страницу таблицу. При получении фокуса строки таблицы поочередно должны выделяться (например, цвет, шрифт, размер и т.д.). Реализуйте горизонтальное двухуровневое меню, раскрывающееся при наведении указателя мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.
Вариант 2	Добавьте на страницу 5 ссылок. Реализуйте поочередное вращение всех ссылок. Разместите на странице два изображения и кнопку «Переставить». При нажатии на данную кнопку изображения должны поменяться местами.
Вариант 3	Создайте кнопку, которая при наведении плавно меняет свой цвет. Реализуйте полет снежинок.
Вариант 4	Создайте индикатор выполнения процесса от 0 до 100%. Разместите на странице четыре изображения, два поля ввода и кнопку «Переставить». При нажатии на данную кнопку изображения с номерами, введенными в текстовые поля, должны поменяться местами.
Вариант 5	Создайте анимацию мигания разделителя (13:27, : - разделитель) электронных часов Реализуйте полет мухи.
Вариант 6	Создайте плавно выпадающее меню. Добавьте на страницу изображение; при наведении указателя мыши оно должно постепенно увеличиваться в размерах, создавая иллюзию приближения изображения (предельные значения высоты и ширины установите самостоятельно).
Вариант 7	Создайте анимацию движения Луны вокруг Земли. Организуйте движение изображения слева направо.
Вариант 8	Создайте иконку меню, которая из трех полос при наведении преобразуется в круг с крестиком. Горизонтальное двухуровневое меню, раскрывающееся при щелчке левой кнопкой мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.

Вариант 9	Создайте анимацию движения текста по кругу. Добавьте на страницу изображение. При наведении указателя мыши оно должно постепенно уменьшаться в размерах, создавая иллюзию удаления изображения (предельные значения высоты и ширины установите самостоятельно).
Вариант 10	Создайте анимацию выцветания текста. Создайте текстовое двухуровневое меню: при выборе определенного пункта меню пункты верхнего уровня раздвигаются и вставляются подпункты выбранного пункта, то есть пункт меню «раскрывается».

4.* Реализовать часы: должны присутствовать секундная, минутная и часовая стрелки.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____
_____
_____

7. Контрольные вопросы и задания

1. Какие подходы используются для работы со стилевыми свойствами элементов в JavaScript?

2. Как можно получить значения всех стилей, которые в данный момент применены к элементу (с учётом внешних файлов стилей и тега <style>)?

3. Перечислите известные Вам способы создания визуальных анимационных эффектов?

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: https://learn.javascript.ru. – Дата доступа: 04.03.2020.

2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.

3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.

4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.