JS

Лабораторная работа 5.

Создадим папку lr5. Откроем ее в vsCode (файл – открыть папку…). Это будет проектом. Создадим в нем файлы Index.html и script.js.

Index.html

<html>

<head>

<meta charset="UTF-8">

    <script src="script.js"></script>

<title>Лабораторная работа 5</title>

</head>

<body>

</body>

</html>

**Выполнение.**

1 Методы работы с датой и временем

В JS за работу с временем и датой отвечает объект Date. При создании объекта конструктором по умолчанию создается метка текущего времени (кол-во миллисекунд с 1 января 1970)

let now = new Date();

console.log(now);

При выводе объекта Date автоматически срабатывает метод toString().

1.1Метод toLocaleString преобразовывает дату и время в строку в соответствии с региональными настройками пользователя.

Метод toLocaleDateString преобразовывает только дату в строку в соответствии с региональными настройками пользователя.

Метод toLocaleTimeString преобразовывает только время в строку в соответствии с региональными настройками пользователя.

let now = new Date();

document.write(now.toLocaleString() + '<br>');

document.write(now.toLocaleDateString() + '<br>');

document.write(now.toLocaleTimeString() + '<br>');

1.2Метод getDate вернет число

Метод getMonth вернет номер месяца (январь – 0 месяц)

Метод getDay вернет номер дня в неделе

Метод getFullYear вернет год (getYear вернет год – 1900)

Метод getHours вернет количество часов

Метод getMinutes вернет количество минут

Метод getSeconds вернет количество секунд

Метод getMilliseconds вернет количество миллисекунд

let now = new Date();

document.write(now.getDate() + '.');

document.write(now.getMonth() + 1 + '.');

document.write(now.getFullYear() + '<br>');

document.write(now.getHours() + ':');

document.write(now.getMinutes() + ':');

document.write(now.getSeconds() + ':');

document.write(now.getMilliseconds());

Для того, чтобы получить время UTC есть аналогичные методы GetUTC

let now = new Date();

document.write(now.getUTCDate() + '.');

document.write(now.getUTCMonth() + 1 + '.');

document.write(now.getUTCFullYear() + '<br>');

document.write(now.getUTCHours() + ':');

document.write(now.getUTCMinutes() + ':');

document.write(now.getUTCSeconds() + ':');

document.write(now.getUTCMilliseconds());

1.3Метод setFullYear установит год

Метод setMonth установит месяц

Метод setDate установит дату

Метод setHours установит часы

Метод setMinutes установит минуты

Метод setSeconds установит секунды

Метод setMilliseconds установит миллисекунды

Метод setTime установит кол-во миллисекунд с 1970 года.

Аналогично есть методы UTC.

Если задать слишком большое значение, то остаток добавиться в старшее значение, например

let now = new Date();

now.setDate(35)

document.write(now.toLocaleString() + '.');

1.4Можно задать дату в конструкторе.

let now = new Date();

let newYear = new Date(now.getFullYear() + 1, 0, 1, 0, 0, 0);

document.write(newYear.toLocaleString() + '.');

Так же в конструктор можно задать время в миллисекундах.

1.5Т.к даты представляют собой кол-во миллисекунд – с ними можно производить арифметические действия.

let now = new Date();

let newYear = new Date(now.getFullYear() + 1, 0, 1, 0, 0, 0);

document.write('До нового года осталось ' + parseInt((newYear - now) / (1000 \* 60 \* 60 \* 24)) + '.');

1.6Статический метод Date.parse преобразует строку с датой в миллисекунды с 1970

2Методы для таймеров

2.1setTimeout позволяет вызвать функцию через заданный период времени.

function myFunc(param) {

    document.write(param);

}

setInterval(myFunc, 2000, 'параметр');

2.2setInterval позволяет создать таймер для вызова функции.

function myFunc(param) {

    console.log(param);

}

setInterval(myFunc, 500, 'параметр');

Функция setTimeout и SetInterval возвращает числовой идентификатор таймера timerId, который можно использовать для отмены действия.

function myFunc(param) {

    console.log(param);

}

let myTimerId = setInterval(myFunc, 500, 'параметр');

setTimeout(function () {

    clearTimeout(myTimerId);

}, 5000);

2.3Рекурсивный setTimeout

let timerId = setTimeout(function tick() {

    alert( "тик" );

    timerId = setTimeout(tick, 2000);

}, 2000);

Рекурсивный setTimeout гарантирует паузу между вызовами функции, setInterval нет. Это происходит потому, что setInterval не учитывает время, которое тратится на обработку функции и если ее обработка занимает больше времени, чем пауза между ними – вызов будет вообще без пауз. При использовании рекурсивного setTimeout задержка всегда фиксирована.