20230918

1.作用域

1.定义：

变量在内存中的生命周期（有效范围）

分类：全局变量 局部变量

1. 全局变量，在js文件中声明，在整个js文件中都有效
2. window全局对象
3. 局部变量：声明在代码块中/函数中的变量，为局部变量
4. 不使用var声明

   ！！！不能全使用全局变量，会占用内存空间，降低运行效率
5. 变量的声明提升

   如果代码块/函数中有和全局变量名冲突的时候，那么整个代码块/函数。优先识别局部变量

2.面向对象

所有IT语言分为面向过程和面向对象 两种开发方式

js属于面向对象语言

面向对象：类 对象

1.类和对象

思想：IT建模思想，显示世界和代码的沟通

类，描述现实中的一类事物（如，人类，公交车，学生……）

对象：一类是无阿后者那个的一个具体个体。/类的实例 （如，张三，某辆车）

```
function person(name,age){

            // 声明属性 this.xxx 左侧的xxx为person类的属性 右侧为参数

            this.name = name;
            this.age = age;

        }

        //创建对象实例（创建某个类的对象）
        //格式：var 对象名=new类名（参数）；
        var person01 = new person('zhangsan', '19');
        console.log(person01.name);
        var person02 = new person('lisi', '20')
        person02.name = 'lisan';
        console.log(person02.name);
```

2.类和方法

```
function person (name, age) {

    this.name = name;
    this.age = age;

    this.run=function () {

        console.log(this.name+',is running.')
    };
    this.sleep = function () {

        alert(this.name+' is sleeping');
    }
}


var person01 = new person('lisan',20)
person01.run();
person01.sleep();
```

3.对象的引用

```
function person (name, age) {

    this.name = name;
    this.age = age;

    this.run=function () {

        console.log(this.name+',is running.')
    };
    this.sleep = function () {

        alert(this.name+' is sleeping');
    }
}


var person01 = new person('lisan',20)
//对象的赋值 将person01的引用赋值给person02 此时person01与person02为同一对象
var person02 = person01

console.log(person02.name)

person02.run();

person02.sleep();
```

4.属性添加和删除

```
    function Person(name,age) {

        this.name = name;
        this.age = age;

        this.run = function(){
            console.log(this.name+'is running');
        }
    }

    var person01 = new Person('lisan',19)
    // 添加
    person01.gender = 'male';
    console.log(person01)

    var person02 = new Person('lisirui',20)
    console.log(person02)
    // 删除
    delete person01.gender
    delete person02.age
    person02.run()
```

5.Object类

该类没有实际意义

所有的js的类，都可以认为是该类的子类

作用：通常情况下 不直接使用Object类，当其可以创建空对象 保存数据

```
var obj= new Object();

    obj.val_1='lisan'
    obj.val_2='123456'
    obj.val_3=function(a,b) {

        return a>b?a:b

    }

    console.log(obj)
```

6.字面量对象

传统方式，先声明类，实例化类的对象

字面量对象

```
    //创建了一个person01对象，有以下属性和方法，属于object类
        var person01={

            name: 'John',
            age:'23',
            run : function(){
                console.log(this.name+'is running');
            }
        }

        console.log(person01)
```

7.复杂字面量对象

```
    var person01={

            name:'lisan',
            age:'19',
            score:{
                Math:97,
                English:100
            },
            run:function() {
                console.log(this.name+'is running')
            }
        }

        console.log(person01.score.English)
        console.log(person01['score'])
```

3.js常用内置对象

定义：独立于宿主环境的ECMAScrip对象

特点：js内置u底线是浏览器开发商依据标准，使用原生j所开发的对象（function）

js内置对象与浏览器，网页上元素无关

js内置对象在页面加载之前就可以使用

分类：

本地对象：js开发的引用类型String Function

内置对象：已被实例化的对象，如：Math

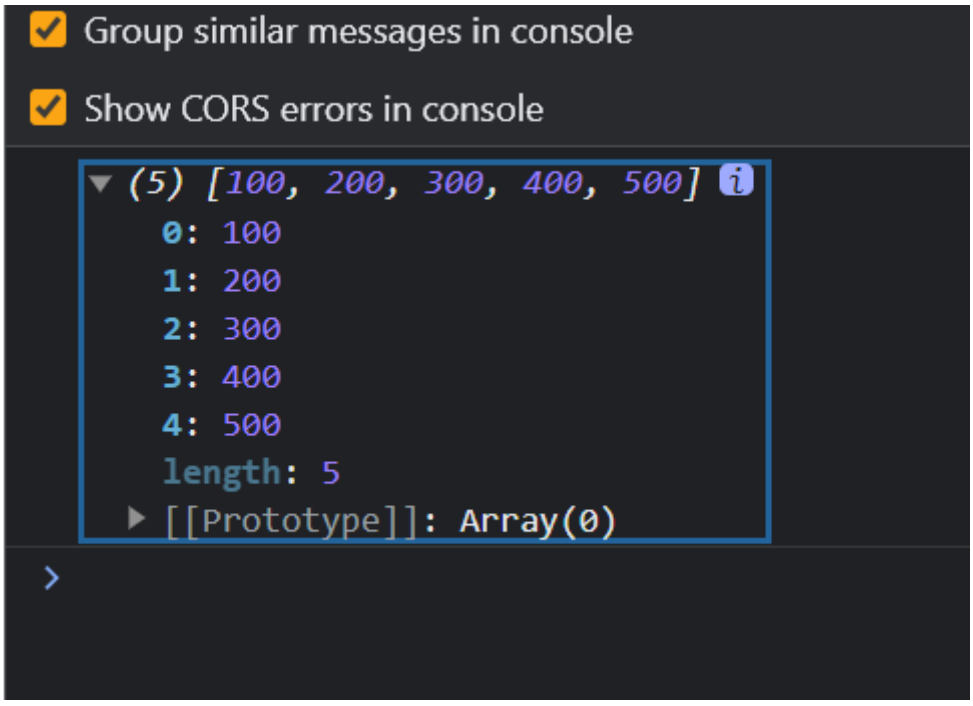
1.数组

数组对象Array

案例：

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        var arr = new Array();

        arr[0] = 100;
        arr[1] = 200;
        arr[2] = 300;
        arr[3] = 400;
        arr[4] = 500;

        console.log(arr);

    </script>
</body>
</html>
```



数组是js的存储介质，可以同时存储不同类型的数据。

1.数组的创建

格式：

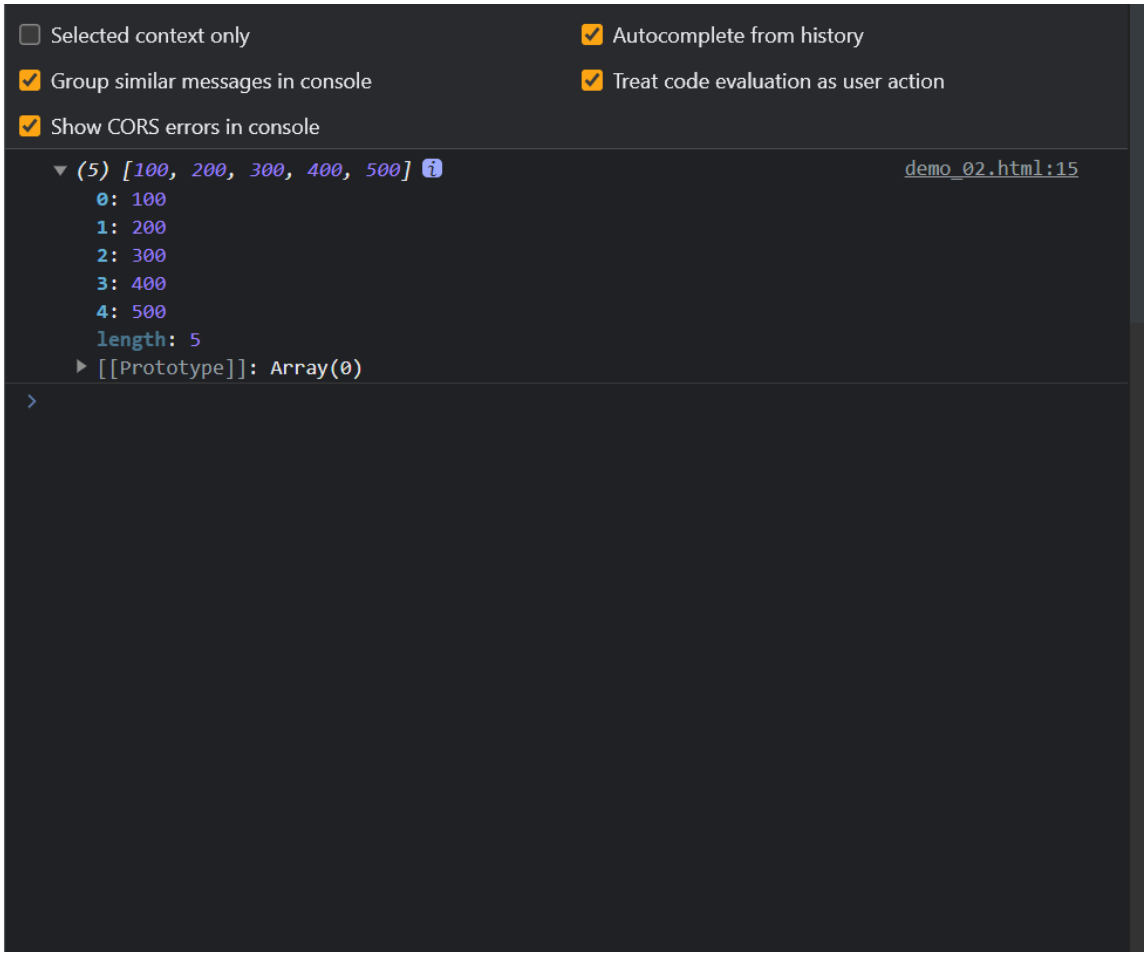
var 数组名 = new Array()

见上图

2.数组的创建2

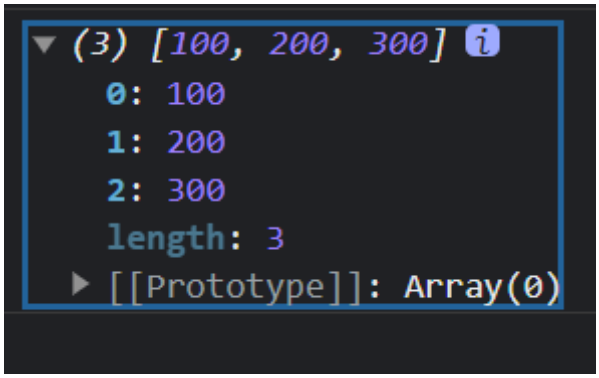
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>


    <script>

        var arr = new Array(100,200,300,400,500);

        console.log(arr);

    </script>

</body>
</html>
```



2.1.数组创建方式3

方式2的简写版



```
//常用写法
        var arr_1 = [100,200,300]

        console.log(arr_1);
```

3.下标以及长度

```
        var arr = [100,'abc',,,,,,,,true,,,,,]

        console.log(arr.length)

        //下标/角标/下角标/序列号

        // 格式：数组名【下标】第下标值+1个元素 arr[0]表述数组中的第一个元素

        console.log(arr[1])
        // !!!js中，下标超出实际数组范围取值时，显示undifferentiated
        console.log(arr[100])
```

4.数组的遍历

1. 方式1

   通过遍历数组，可以拿到数组中的每一个元素

   本质：循环下标（0~数组长度减1）

2. 方式2：for-in 循环

   格式：var XXX in 数组名

   在循环体中，XXX表示下标值

3. 方式3：for-of循环

   格式：var xxx of 数组名

   在循环体中，xxx就表示数组中的每一个元素

```
        var arr = [100,'abc',true]

        for (var i=0; i<arr.length; i++){
            console.log(arr[i])
        }

        for (var index in arr){

            console.log(arr[index])
        }
        for (var e of arr){
            console.log(e);
        }
```

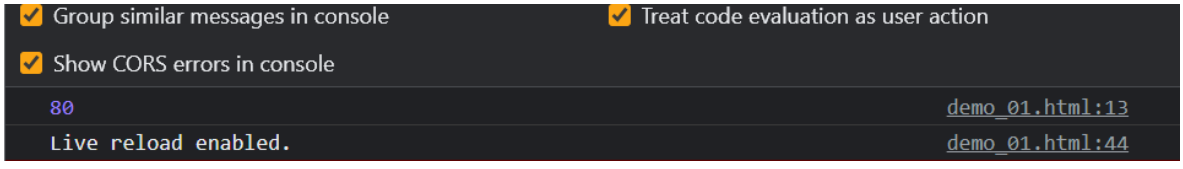
5.数组常用方法

```
        // 数组的的头部尾部添加删除
        var arr = [1,2,3,4,5,6]
        // .push尾部添加
        arr.push('x')
        arr.push('y')
        arr.push('z')
        // .pop尾部删除
        arr.pop()
        // .unshift头部添加
        arr.unshift('z')
        // shift头部删除
        arr.shift()
        console.log(arr)
```

作业

1.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        var arr = Math.max(3,14,6,80,9)

        console.log(arr)

    </script>
</body>
</html>
```



2.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
```

```
        var arr = [15,7,2,9,21,8,20,18]

        for (var i=0; i<arr.length; i++)
            {
                if(arr[i] == 7){

                    console.log(i);

                }
            }

        var c ;
        c==arr[3];
        arr[3]==arr[6]
        arr[6]==c;
        console.log(arr[3],arr[6]);

    </script>
</body>
</html>
```