JS第一周第二天

`javascript`

1.var，let和const

- var
  1.可以重复声明，后面的值会把前面的覆盖
  2.可以只声明，不赋值，此时变量是undefined
  `in 判断一个属性在不在对象中`
  `console.log("a" in window)`
  3.在全局作用域中，使用var声明的变量会给全局对象window增加一个属性
- let
  1.不可以重复声明
  2.可以只声明，不赋值，此时变量是undefined
  3.使用let声明的变量不会给全局对象win
- const 定义常量
  1.不可以重复声明
  2.使用const声明的常量不会给全局对象window增加属性
  3.赋值之后不可以修改
  4.一旦声明必须赋值

2.toString

任意数据类型都有一个toString方法，只有null和undefined没有这个方法

```
null.toString();// Cannot read property 'toString' of null:  在null
```

```
中没有toString属性
```

- 数字number
  输出为字符串数字
  `console.log(num.toString())`
  数字紧跟着的点表示小数点
  `console.log(2..toString())`输出为字符串”2”
  或者使用小括号将2括起来
  `console.log((2).toString())`
- 布尔boolean
  输出为字符串true和false

`引用数据类型`

- 数组→字符串 ——直接将[ ]变成” “
- 对象{ } →字符串 —— 都会变成”{object Object}”
- 正则→字符串 —— 直接用” “将其包起来
- 时间→字符串 —— 直接用” “将其包起来
- 函数→字符串——直接用” “将其包起来
- `对于正则，时间，函数使用toString变为字符串的意义不大，很少用到。但在后期检测数据类型时会用到`

3.字符串string

1.使用单引号或双引号包起来

2.字符串中的内容没有任何意义，只是一堆字符

3.字符串有length，有索引（从0开始，字符在字符串中的位置表示），可以使用 `字符串[索引]` 获取对应位置的字符，如果索引号超出length，结果为undefined。

4.空格也是有效的字符

5.单双引号不要嵌套同样的引号，如要嵌套同样的引号，需要加/转义，就可以使用一种引号`'I/'m a student'`

6.+遇到字符串表示字符串的拼接 `"1"+"1"变为"11"`
`其他数据类型+字符串：` 也是字符串的拼接，先将其他数据类型变成字符串（默认调用toString方法），再跟字符串进行拼接。因为null和undefined没有toString方法，当遇到字符串拼接时，直接变成字符串（”null”和”undefined”）进行拼接

7.字符串拼接变量，拼接的是变量存储的值

8.eval( )可将字符串转换成可执行的代码

4.字符串拼接

`+"变量"+`再将前后字符串补齐，补齐的字符串不用加单双引号
`ES6中使用ES6的模板字符串进行拼接变量，以后基本都是ES6模板字符串：需要替换的位置使用${变量}`

5.innerHTML

```
*/
<body>
    <ul id="list">
        <li>1111</li>
    </ul>
</body>
*/

<script>
 let list=document.getElementById("list");
    list.innerHTML="<li>2222</li>"
    //'='之前不加'+'会将上面的覆盖
    list.innerHTML+="<li>2222</li>";
    //加了不会覆盖之前内容
</script>
```

6.null和undefined

- null（空）
  初始化或者清空一个值

```
var n=null;//不占内存
```

- undefined（未定义）
  1.只声明未定义的变量 值为undefined
  2.一个函数没有返回值 alert（ ）
  3.一个对象中没有某个属性，获取的时候就是undefined

7.运算符

- “+” 注意字符串为拼接，其他数据类型都是数学运算，除了+之外字符串也是数学运算（ 1*”10”=10 ），不是数字进行数学运算要转为数字（使用number方法）
- NaN和任何值运算结果都是NaN
- “%” 模（取余）

8.布尔boolean()

也是一个方法，将其他数据类型变为布尔值，只有五个为false，其他全为true。
false：0，NaN，”“（空字符串），null， defined
`！和！！的区别：`

- ！表示先转布尔再取反，console.log(!”“)=true
- ！！表示直接转布尔

9.对象

对象：是键值对的组合，是无序的
键值对：属性名（键key）和属性值（value）
{属性名：属性值，属性名：属性值.....}

- 属性名不可以重复，如果重复后面的会将前面的覆盖
- 属性名是字符串，但是写的时候可以省略引号
- 获取属性值可以通过`对象["属性名"]`（中括号里为字符串，如果不写成字符串就是变量）取对应的属性值或`对象.属性名`获取对应的属性值
- 如果对象没有这个属性名，获取出来就是undefined
- 属性名后的属性如果之前没有，就是新增；如果有，就是修改
- 如果属性名为数字，只能使用中括号的形式，而且可以省略引号

10.for In遍历对象

判断属性在不在对象中

```
let obj={}
    for(var key in obj){
        //变量key代表的是属性名
        //对象有几组键值对就会执行几次
        //obj[key]：属性值
        console.log(key, obj[key]);
    }
```

- 遍历对象的时候先遍历属性名是数字的，而且按照从小到大的顺序遍历，剩下的按照书写的顺序遍历

11.数组array

1.有序的集合，放在[ ]中使用”，”隔开
2.有length（数组的长度），有索引（从0开始，表示每一项的位置），通过`数组[索引]`获取对应位置的值
3.数组同对象一样，原来有就是修改，没有就是增加，如果增加的位置超出了数组长度，则中间为空位（一个逗号代表一个空位）
4.用`索引 in 数组`判断这个索引的位置有没有值 true：不是空位 false：是空位

12.基本数据类型和引用数据类型

- 当给变量赋值的时候浏览器会判断是什么数据类型，如果是引用型数据类型，浏览器会默认开辟一个内存地址（堆内存），将地址值赋给变量，将数据存储在堆内存中
- 区别：基本数据类型是操作值的，而引用数据类型是操作地址的