

JavaScript基础第06天笔记

1 - 内置对象

1.1 内置对象

JavaScript 中的对象分为3种：自定义对象、内置对象、浏览器对象

前面两种对象是JS 基础 内容，属于 ECMAScript； 第三个浏览器对象属于 JS 独有的，JS API 讲解 内置对象就是指 JS 语言自带的一些对象，这些对象供开发者使用，并提供了一些常用的或是最基本而必要的功能（属性和方法），内置对象最大的优点就是帮助我们快速开发

JavaScript 提供了多个内置对象：Math、Date、Array、String等

1.2 查文档

查找文档：学习一个内置对象的使用，只要学会其常用成员的使用即可，我们可以通过查文档学习，可以通过MDN/W3C来查询。

Mozilla 开发者网络（MDN）提供了有关开放网络技术（Open Web）的信息，包括 HTML、CSS 和万维网及 HTML5 应用的 API。

MDN:<https://developer.mozilla.org/zh-CN/>

1.3 Math对象

Math 对象不是构造函数，它具有数学常数和函数的属性和方法。跟数学相关的运算（求绝对值，取整、最大值等）可以使用 Math 中的成员。

属性、方法名	功能
Math.PI	圆周率
Math.floor()	向下取整
Math.ceil()	向上取整
Math.round()	四舍五入版 就近取整 注意 -3.5 结果是 -3
Math.abs()	绝对值
Math.max()/Math.min()	求最大和最小值
Math.random()	获取范围在[0,1)内的随机值

注意：上面的方法使用时必须带括号

获取指定范围内的随机整数：

```
function getRandom(min, max) {  
    return Math.floor(Math.random() * (max - min + 1)) + min;  
}
```

1.4 日期对象

Date 对象和 Math 对象不一样，Date 是一个构造函数，所以使用时需要实例化后才能使用其中具体方法和属性。Date 实例用来处理日期和时间

- 使用 Date 实例化日期对象
 - 获取当前时间必须实例化：

```
var now = new Date();
```

- 获取指定时间的日期对象

```
var future = new Date('2019/5/1');
```

注意：如果创建实例时并未传入参数，则得到的日期对象是当前时间对应的日期对象

- 使用 Date 实例的方法和属性

方法名	说明	代码
getFullYear()	获取当年	dObj.getFullYear()
getMonth()	获取当月 (0-11)	dObj.getMonth()
getDate()	获取当天日期	dObj.getDate()
getDay()	获取星期几 (周日0 到周六6)	dObj.getDay()
getHours()	获取当前小时	dObj.getHours()
getMinutes()	获取当前分钟	dObj.getMinutes()
getSeconds()	获取当前秒钟	dObj.getSeconds()

- 通过 Date 实例获取总毫秒数
 - 总毫秒数的含义
 - 基于1970年1月1日（世界标准时间）起的毫秒数
 - 获取总毫秒数

```
// 实例化Date对象
var now = new Date();
// 1. 用于获取对象的原始值
console.log(date.valueOf())
console.log(date.getTime())
// 2. 简单写可以这么做
var now = + new Date();
// 3. HTML5中提供的方法，有兼容性问题
var now = Date.now();
```

1.5 数组对象

创建数组的两种方式

- 字面量方式
 - 示例代码如下：

```
var arr = [1, "test", true];
```

- new Array()
 - 示例代码如下：

```
var arr = new Array();
```

注意：上面代码中arr创建出的是一个空数组，如果需要使用构造函数Array创建非空数组，可以在创建数组时传入参数

参数传递规则如下：

- 如果只传入一个参数，则参数规定了数组的长度
- 如果传入了多个参数，则参数称为数组的元素

检测是否为数组

- instanceof 运算符
 - instanceof 可以判断一个对象是否是某个构造函数的实例

```
var arr = [1, 23];
var obj = {};
console.log(arr instanceof Array); // true
console.log(obj instanceof Array); // false
```

- Array.isArray()
 - Array.isArray()用于判断一个对象是否为数组，isArray() 是 HTML5 中提供的方法

```
var arr = [1, 23];
var obj = {};
console.log(Array.isArray(arr));    // true
console.log(Array.isArray(obj));    // false
```

添加删除数组元素的方法

- 数组中有进行增加、删除元素的方法，部分方法如下表

方法名	说明	返回值
push(参数1....)	末尾添加一个或多个元素，注意修改原数组	并返回新的长度
pop()	删除数组最后一个元素，把数组长度减 1 无参数、修改原数组	返回它删除的元素的值
unshift(参数1...)	向数组的开头添加一个或更多元素，注意修改原数组	并返回新的长度
shift()	删除数组的第一个元素，数组长度减 1 无参数、修改原数组	并返回第一个元素的值

注意：push、unshift为增加元素方法；pop、shift为删除元素的方法

数组排序

- 数组中有对数组本身排序的方法，部分方法如下表

方法名	说明	是否修改原数组
reverse()	颠倒数组中元素的顺序,无参数	该方法会改变原来的数组 返回新数组
sort()	对数组的元素进行排序	该方法会改变原来的数组 返回新数组

注意：sort方法需要传入参数来设置升序、降序排序

- 如果传入“function(a,b){ return a-b;}”，则为升序
- 如果传入“function(a,b){ return b-a;}”，则为降序

数组索引方法

- 数组中有获取数组指定元素索引值的方法，部分方法如下表

方法名	说明	返回值
indexOf()	数组中查找给定元素的第一个索引	如果存在返回索引号 如果不存在，则返回-1。
lastIndexOf()	在数组中的最后一个的索引，	如果存在返回索引号 如果不存在，则返回-1。

数组转换为字符串

- 数组中有把数组转化为字符串的方法，部分方法如下表

方法名	说明	返回值
toString()	把数组转换成字符串，逗号分隔每一项	返回一个字符串
join('分隔符')	方法用于把数组中的所有元素转换为一个字符串。	返回一个字符串

注意：join方法如果不传入参数，则按照“,”拼接元素

其他方法

- 数组中还有其他操作方法，同学们可以在课下自行查阅学习

方法名	说明	返回值
concat()	连接两个或多个数组 不影响原数组	返回一个新的数组
slice()	数组截取slice(begin, end)	返回被截取项目的新数组
splice()	数组删除splice(第几个开始,要删除个数)	返回被删除项目的新数组 注意，这个会影响原数组

1.6 字符串对象

基本包装类型

为了方便操作基本数据类型，JavaScript 还提供了三个特殊的引用类型：String、Number和Boolean。

基本包装类型就是把简单数据类型包装成为复杂数据类型，这样基本数据类型就有了属性和方法。

```
// 下面代码有什么问题？
var str = 'andy';
console.log(str.length);
```

按道理基本数据类型是没有属性和方法的，而对象才有属性和方法，但上面代码却可以执行，这是因为

js 会把基本数据类型包装为复杂数据类型，其执行过程如下：

```
// 1. 生成临时变量，把简单类型包装为复杂数据类型
var temp = new String('andy');
// 2. 赋值给我们声明的字符变量
str = temp;
// 3. 销毁临时变量
temp = null;
```

字符串的不可变

指的是里面的值不可变，虽然看上去可以改变内容，但其实是地址变了，内存中新开辟了一个内存空间。

当重新给字符串变量赋值的时候，变量之前保存的字符串不会被修改，依然在内存中重新给字符串赋值，会重新在内存中开辟空间，这个特点就是字符串的不可变。

由于字符串的不可变，在大量拼接字符串的时候会有效率问题

根据字符返回位置

字符串通过基本包装类型可以调用部分方法来操作字符串，以下是返回指定字符的位置的方法：

方法名	说明
indexOf('要查找的字符', 开始的位置)	返回指定内容在元字符串中的位置， 如果找不到就返回 -1， 开始的位置是 index 索引号
lastIndexOf()	从后往前找， 只找第一个匹配的

案例：查找字符串"abcfoxyozzopp"中所有o出现的位置以及次数

1. 先查找第一个o出现的位置
2. 然后 只要indexOf 返回的结果不是 -1 就继续往后查找
3. 因为indexOf 只能查找到第一个， 所以后面的查找， 利用第二个参数， 当前索引加1， 从而继续查找

根据位置返回字符

字符串通过基本包装类型可以调用部分方法来操作字符串， 以下是根据位置返回指定位置上的字符：

方法名	说明	使用
charAt(index)	返回指定位置的字符(index 字符串的索引号)	str.charAt(0)
charCodeAt(index)	获取指定位置处字符的ASCII码 (index索引号)	str.charCodeAt(0)
str[index]	获取指定位置处字符	HTML5, IE8+支持 和charAt()等效

在上述方法中， charCodeAt方法返回的是指定位置上字符对应的ASCII码， ASCII码对照表如下：

ASCII表																										
(American Standard Code for Information Interchange 美国标准信息交换代码)																										
高四位	ASCII控制字符													ASCII打印字符												
	0000						0001						0010	0011	0100	0101	0110	0111								
	0						1						2	3	4	5	6	7								
低四位	十进制	字符	Ctrl	代码	转义	字符解释	十进制	字符	Ctrl	代码	转义	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl	
0000	0	0		^@	NUL	\0 空字符	16	▶	^P	DLE		数据块转义	32		48	0	64	@	80	P	96	`	112	p		
0001	1	1	☺	^A	SOH	标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q		
0010	2	2	☹	^B	STX	正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r		
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s		
0100	4	4	♦	^D	EOF	传输结束	20	☐	^T	DC4		设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t		
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAE		否定应答	37	%	53	5	69	E	85	U	101	e	117	u		
0110	6	6	♠	^F	ACK	肯定应答	22	—	^V	STN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v		
0111	7	7	•	^G	BEL	响铃	23	↕	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w		
1000	8	8	☐	^H	BS	\b 退格	24	↑	^X	CAN		取消	40	(56	8	72	H	88	X	104	h	120	x		
1001	9	9	○	^I	HT	\t 横向制表	25	↓	^Y	EN		介质结束	41)	57	9	73	I	89	Y	105	i	121	y		
1010	A	10	☐	^J	LF	\n 换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z		
1011	B	11	♂	^K	VT	\v 纵向制表	27	←	^[ESC	\e 溢出	43	+	59	;	75	K	91	[107	k	123	{			
1100	C	12	♀	^L	FF	\f 换页	28	└	^[_	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124			
1101	D	13	♪	^M	CR	\r 回车	29	↔	^J	GS		组分隔符	45	-	61	=	77	M	93]	109	m	125	}		
1110	E	14	🎵	^N	SD		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^_	US		单元分隔符	47	/	63	?	79	O	95		111	o	127	␣	*Backspace 代码: DEL

案例：判断一个字符串 'abcfoxyozzopp' 中出现次数最多的字符，并统计其次数

1. 核心算法：利用 charAt() 遍历这个字符串

2. 把每个字符都存储给对象，如果对象没有该属性，就为1，如果存在了就 +1

3. 遍历对象，得到最大值和该字符

注意：在遍历的过程中，把字符串中的每个字符作为对象的属性存储在对象总，对应的属性值是该字符出现的次数

字符串操作方法

字符串通过基本包装类型可以调用部分方法来操作字符串，以下是部分操作方法：

方法名	说明
<code>concat(str1,str2,str3...)</code>	<code>concat()</code> 方法用于连接两个或多个字符串。拼接字符串，等效于 <code>+</code> ， <code>+</code> 更常用
<code>substr(start,length)</code>	从 <code>start</code> 位置开始（索引号）， <code>length</code> 取的个数 重点记住这个
<code>slice(start, end)</code>	从 <code>start</code> 位置开始，截取到 <code>end</code> 位置， <code>end</code> 取不到（他们俩都是索引号）
<code>substring(start, end)</code>	从 <code>start</code> 位置开始，截取到 <code>end</code> 位置， <code>end</code> 取不到 基本和 <code>slice</code> 相同 但是不接受负值

`replace()`方法

`replace()` 方法用于在字符串中用一些字符替换另一些字符，其使用格式如下：

```
字符串.replace(被替换的字符串, 要替换为的字符串);
```

`split()`方法

`split()`方法用于切分字符串，它可以将字符串切分为数组。在切分完毕之后，返回的是一个新数组。

其使用格式如下：

```
字符串.split("分割字符")
```

2 - 简单数据类型和复杂数据类型

2.1 简单数据类型

简单类型（基本数据类型、值类型）：在存储时变量中存储的是值本身，包括`string`，`number`，`boolean`，`undefined`，`null`

2.2 复杂数据类型

复杂数据类型（引用类型）：在存储时变量中存储的仅仅是地址（引用），通过 `new` 关键字创建的对象（系统对象、自定义对象），如 `Object`、`Array`、`Date`等；

2.3 堆栈

- 堆栈空间分配区别：

1、栈（操作系统）：由操作系统自动分配释放存放函数的参数值、局部变量的值等。其操作方式类似于数据结构中的栈；

简单数据类型存放到栈里面

2、堆（操作系统）：存储复杂类型(对象)，一般由程序员分配释放，若程序员不释放，由垃圾回收机制回收。



● 简单数据类型的存储方式

值类型变量的数据直接存放在变量（栈空间）中

		栈	堆
var usrAge = 12;	0x11	12	
var usrName = '小白';	0x20	小白	
var usrSex = 'true';	0x32	true	

● 复杂数据类型的存储方式

引用类型变量（栈空间）里存放的是地址，真正的对象实例存放在堆空间中



2.4 简单类型传参

函数的形参也可以看做是一个变量，当我们把一个值类型变量作为参数传给函数的形参时，其实是把变量在栈空间里的值复制了一份给形参，那么在方法内部对形参做任何修改，都不会影响到的外部变量。

```
function fn(a) {
  a++;
  console.log(a);
}
var x = 10;
fn(x);
console.log(x);
```

运行结果如下：

11

10

2.5 复杂数据类型传参

函数的形参也可以看做是一个变量，当我们把引用类型变量传给形参时，其实是把变量在栈空间里保存的堆地址复制给了形参，形参和实参其实保存的是同一个堆地址，所以操作的是同一个对象。

```
function Person(name) {  
    this.name = name;  
}  
function f1(x) { // x = p  
    console.log(x.name); // 2. 这个输出什么 ?  
    x.name = "张学友";  
    console.log(x.name); // 3. 这个输出什么 ?  
}  
var p = new Person("刘德华");  
console.log(p.name); // 1. 这个输出什么 ?  
f1(p);  
console.log(p.name); // 4. 这个输出什么 ?
```

运行结果如下：

刘德华

刘德华

张学友

张学友
