JavaScript 基础

第五课

内置对象

自定义构造函数

> Object 是所有JavaScript对象的基础，所以所有的对象都可以通过Object来创建，但是Object创建出来的是一个空的对象。
>
> 通过自定义的构造函数，创建一个对象

```
// 帕斯卡命名     第一个单词 首字母大写，后续每一个单词的首字母大写
function Person(name, age, job){
  // this 动态的给对象添加属性和方法
  // this 指向当前对象
  this.name = name;
  this.age = age;
  this.job = job;
  this.sayHi = function(){
    console.log('Hello,everyBody');
  }
}
var p1 = new Person('张三', 22, 'actor');
var p2 = new Person('李四', 18, 'scientist');

// 比工厂函数  更简洁 ，功能上一样
// 构造函数语义上更规范
```

new关键字

> 构造函数 ，是一种特殊的函数。主要用来在创建对象时初始化对象， 即为对象成员变量赋初始值，总与new运算符一起使用在创建对象的语句中。

1. 构造函数用于创建一类对象，首字母要大写。
2. 构造函数要和new一起使用才有意义。

- new 在执行的时候会做四件事
    1. new会在内存中创建一个新的空对象
    2. new 会让this指向这个新的对象
    3. 执行构造函数 目的：给这个新对象加属性和方法
    4. new会返回这个新对象

instance of

- typeof 返回值是一个字符串，该字符串说明表达式结果的数据类型。
    - 返回结果： `number`、 `string`、 `boolean`、 `object`、 `function` 、 `undefined`

    - 数字 `typeof x = 'number'`
    - 字符串 `typeof x = 'string'`
    - 布尔值 `typeof x = 'boolean'`
    - 对象、数组、null类型： `typeof x = 'object'`
    - 函数：`typeof x = 'function'`
- instanceof 用于判断一个变量是否属于某个构造函数的实例
    - 在采用 typeof 判断一个引用类型的 数据类型的时候，总会返回 object
    - 定义： 用来测试一个对象在其原型链中是否存在一个构造函数的prototype属性
    - 语法： `object instanceof Constructor`

```
// typeof返回值是一个字符串，该字符串说明表达式结果的数据类型。
    var a = 1, b = 'abc', c = true, d, f = null, e = {};
    function g() {};

    console.log(typeof a); // number
    console.log(typeof b); // string
    console.log(typeof c); // boolean
    console.log(typeof d); // undefined
    console.log(typeof e); // object  注意
    console.log(typeof f); // object
    console.log(typeof g); // function


// instanceof 判断一个变量是那个构造函数的实例

    function A (){

    }
    var aa = new A();
    console.log(g instanceof Object);
    console.log(g instanceof Function);
    console.log(aa instanceof A);
    console.log(aa instanceof Object);
```

内置对象

> JavaScript中对象分为三种：内置对象、浏览器对象、自定义对象
>
> JavaScript 提供多个内置对象：Math/Array/Number/String/Boolean...
>
> 对象只是带有**属性**和**方法**的特殊数据类型。
>
> 学习一个内置对象的使用，只要学会其常用的成员的使用（通过查文档学习）
>
> 可以通过MDN/W3C来查询
>
> 内置对象的方法很多，我们只需要知道内置对象提供的常用方法，使用的时候查询文档。

MDN

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

- MDN
- 通过查询MDN学习Math对象的random()方法的使用

如何学习一个方法?

- 方法的功能
- 参数的意义和**类型**
- 返回值意义和**类型**
- demo进行测试

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，都是以静态成员的方式提供

跟数学相关的运算来找Math中的成员（求绝对值，取整）

Math

演示：Math.PI、Math.random()、Math.floor()/Math.ceil()、Math.round()、Math.abs() 、Math.max()

```
Math.PI                     // 圆周率
Math.random()               // 生成随机数
Math.floor()/Math.ceil()     // 向下取整/向上取整
Math.round()                // 取整，四舍五入
Math.abs()                  // 绝对值
Math.max()/Math.min()        // 求最大和最小值

Math.sin()/Math.cos()        // 正弦/余弦
Math.power()/Math.sqrt()     // 求指数次幂/求平方根
```

案例:

1. 求10-20之间的随机数 [10, 20] 整数
2. 随机生成颜色RGB
3. 模拟实现max()/min()

Date对象

创建 `Date` 实例用来处理日期和时间。Date 对象基于1970年1月1日（世界标准时间）起的毫秒数。

```
// 获取当前时间，UTC世界时间，距1970年1月1日（世界标准时间）起的毫秒数
var now = new Date();
console.log(now.valueOf()); // 获取距1970年1月1日（世界标准时间）起的毫秒数

Date构造函数的参数
1. 毫秒数 1498099000356          new Date(1498099000356)
2. 日期格式字符串  '2015-5-1'    new Date('2015-5-1')
3. 年、月、日.....               new Date(2015, 4, 1)   // 月份从0开始
```

- 获取日期的毫秒形式

```
var now = new Date();
// valueOf用于获取对象的原始值
console.log(date.valueOf())

// HTML5中提供的方法，有兼容性问题
var now = Date.now();

// 不支持HTML5的浏览器，可以用下面这种方式
var now = + new Date();          // 调用 Date对象的valueOf()
```

- 日期格式化方法

```
toString()       // 转换成字符串
valueOf()        // 获取毫秒值
// 下面格式化日期的方法，在不同浏览器可能表现不一致，一般不用
toDateString()
toTimeString()
toLocaleDateString()
toLocaleTimeString()
```

- 获取日期指定部分

```
getTime()     // 返回毫秒数和valueOf()结果一样，valueOf()内部调用的getTime()
getMilliseconds()
getSeconds()  // 返回0-59
getMinutes()  // 返回0-59
getHours()    // 返回0-23
getDay()      // 返回星期几 0周日    6周6
getDate()     // 返回当前月的第几天
getMonth()    // 返回月份，***从0开始***
getFullYear() //返回4位的年份   如 2016
```

案例：

1. 写一个函数，格式化日期对象，返回yyyy-MM-dd HH:mm:ss的形式

```
function formatDate(d) {
  //如果date不是日期对象，返回
  if (!date instanceof Date) {
    return;
  }
  var year = d.getFullYear(),
      month = d.getMonth() + 1,
      date = d.getDate(),
      hour = d.getHours(),
      minute = d.getMinutes(),
      second = d.getSeconds();
  month = month < 10 ? '0' + month : month;
```

```
  date = date < 10 ? '0' + date : date;
  hour = hour < 10 ? '0' + hour : hour;
  minute = minute < 10 ? '0' + minute:minute;
  second = second < 10 ? '0' + second:second;
  return year + '-' + month + '-' + date + ' ' + hour + ':' + minute + ':' +
second;
```

2. 计算时间差，返回相差的天/时/分/秒

```
function getInterval(start, end) {
  var day, hour, minute, second, interval;
  interval = end - start;
  interval /= 1000;
  day = Math.round(interval / 60 /60 / 24);
  hour = Math.round(interval / 60 /60 % 24);
  minute = Math.round(interval / 60 % 60);
  second = Math.round(interval % 60);
  return {
    day: day,
    hour: hour,
    minute: minute,
    second: second
  }
}
```

Array对象

创建数组对象的两种方式

- 字面量方式
- new Array()

```

// 1. 使用构造函数创建数组对象
// 创建了一个空数组
var arr = new Array();
// 创建了一个数组，里面存放了3个字符串
var arr = new Array('zs', 'ls', 'ww');
// 创建了一个数组，里面存放了4个数字
var arr = new Array(1, 2, 3, 4);


// 2. 使用字面量创建数组对象
var arr = [1, 2, 3];

// 获取数组中元素的个数
console.log(arr.length);
```

检测一个对象是否是数组

- instanceof
- Array.isArray()

  函数的参数，如果要求是一个数组的话，可以用这种方式来进行判断

  优于instanceof,但是可能会出现兼容性

toString/valueOf

- toString() 把数组转换成字符串，逗号分隔每一项
- valueOf() 返回数组对象本身

数组常用方法

演示：push()、shift()、unshift()、reverse()、sort()、splice()、indexOf()

```
// 1 栈操作(先进后出)
push()
pop()        //取出数组中的最后一项，修改length属性
// 2 队列操作(先进先出)
push()
shift()      //取出数组中的第一个元素，修改length属性
unshift()    //在数组最前面插入项，返回数组的长度
// 3 排序方法
reverse()    //翻转数组
sort();      //即使是数组sort也是根据字符，从小到大排序
// 带参数的sort是如何实现的?
// 4 操作方法
concat()     //把参数拼接到当前数组
slice()      //从当前数组中截取一个新的数组，不影响原来的数组，参数start从0开始,end从1开始
splice()     //删除或替换当前数组的某些项目，参数start，deleteCount，options(要替换的项目)
// 5 位置方法
indexOf()、lastIndexOf()    //如果没找到返回-1
// 6 迭代方法 不会修改原数组(可选)
every()、filter()、forEach()、map()、some()
// 7 方法将数组的所有元素连接到一个字符串中。
join()
```

清空数组的方法

```
// 方式1 推荐
arr = [];
// 方式2
arr.length = 0;
// 方式3
arr.splice(0, arr.length);
```

案例：

- 将一个字符串数组输出为|分割的形式，比如“刘备|张飞|关羽”。使用两种方式实现

```
function myJoin(array, seperator) {
  seperator = seperator || ',';
```

```
  array = array || [];
  if (array.length == 0){
    return '';
  }
  var str = array[0];
  for (var i = 1; i < array.length; i++) {
    str += seperator + array[i];
  }
  return str;
}
var array = [6, 3, 5, 6, 7, 8, 0];
console.log(myJoin(array, '-'));

console.log(array.join('-'))
```

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/Boolean

```
// 下面代码的问题?
// s1是基本类型，基本类型是没有方法的
var s1 = 'zhangsan';
var s2 = s1.substring(5);

// 当调用s1.substring(5)的时候，先把s1包装成String类型的临时对象，再调用substring方法，最后销毁临时对象，相当于:
var s1 = new String('zhangsan');
var s2 = s1.substring(5);
s1 = null;
```

```
// 创建基本包装类型的对象
var num = 18;                   //数值，基本类型
var num = Number('18');         //类型转换
var num = new Number(18);       //基本包装类型，对象
// Number和Boolean基本包装类型基本不用，使用的话可能会引起歧义。例如:
var b1 = new Boolean(false);
var b2 = b1 && true;            // 结果是什么
```

String对象

字符串的不可变

```
var str = 'abc';
str = 'hello';
// 当重新给str赋值的时候，常量'abc'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
```

创建字符串对象

```
var str = new String('Hello World');

// 获取字符串中字符的个数
console.log(str.length);
```

字符串对象常用的方法

字符串所有的方法，都不会修改字符串本身(字符串是不可变的)，操作完成会返回一个新的字符串

```
// 1 字符方法
charAt()        //获取指定位置处字符
charCodeAt()    //获取指定位置处字符的ASCII码
str[0]          //HTML5，IE8+支持 和charAt()等效
// 2 字符串操作方法
concat()        //拼接字符串，等效于+，+更常用
slice()         //从start位置开始，截取到end位置，end取不到
substring()     //从start位置开始，截取到end位置，end取不到
substr()        //从start位置开始，截取length个字符
// 3 位置方法
indexOf()       //返回指定内容在元字符串中的位置
lastIndexOf()   //从后往前找，只找第一个匹配的
// 4 去除空白
trim()          //只能去除字符串前后的空白
// 5 大小写转换方法
to(Locale)UpperCase()   //转换大写
to(Locale)LowerCase()   //转换小写
// 6 其它
search()
replace()
split()
fromCharCode()
// String.fromCharCode(101, 102, 103);   //把ASCII码转换成字符串
```

案例:

- 截取字符串"我爱中华人民共和国"，中的"中华"

```
var s = "我爱中华人民共和国";
s = s.substr(2,2);
console.log(s);
```

- "abcoefoxyozzopp"查找字符串中所有o出现的位置

```
var s = 'abcoefoxyozzopp';
var array = [];
do {
  var index = s.indexOf('o', index + 1);
  if (index != -1) {
    array.push(index);
  }
} while (index > -1);
console.log(array);
```

- 把字符串中所有的o替换成!

```
var s = 'abcoefoxyozzopp';
do {
  s = s.replace('o', '');
} while (s.indexOf('o') > -1);
console.log(s);

console.log(s.replace(/o/gi, ''));
```

作业

1. 重写 Math对象的max和min方法
2. 格式化时间 "2019/01/19 18:28:00 周*"
3. 倒计时 几天几小时几分几秒
4. 数组排序 ['ad','abc','a','bdcdef']; 按照元素字符串个数 降序排列
5. 改写原来的冒泡排序函数，通过自定义方法来控制排序方式【仿Array.sort()】
6. 模拟array.reverse函数
7. 工资数组[1500, 1200, 2000, 2100, 1800]，把工资低于2000的元素生成新数组(可用多种关于数组的方法)
8. ['c', 'a', 'z', 'a', 'x', 'a'] 找到每一个 a出现的位置 【indexOf/lastIndexOf】
9. 数组去重 ['c', 'a', 'z', 'a', 'x', 'a']

给定一个字符串如：“abaasdffggghhjjkkgfddssssss3444343”问题如下：

1. 字符串的长度
2. 取出指定位置的字符，如：0,3,5,9等
3. 查找指定字符是否在以上字符串中存在，如：i，c，b等
4. 替换指定的字符，如：g替换为22,ss替换为b等操作方法
5. 截取指定开始位置到结束位置的字符串，如：取得1-5的字符串
6. 找出以上字符串中出现次数最多的字符和出现的次数
7. 遍历字符串，并将遍历出的字符两头添加符号“@”输出至当前的文档页面。
8. 判断一个字符串中出现次数最多的字符，统计这个次数
9. 获取一个 url中?后边的内容：http://www.baidu.com/?name=zs&age=18&a=1&b=2