第十一课 引用类型

学习目录

- Object 类型
- Array 类型
- Function 类型
- Date 类型
- RegExp 类型
- 基本包装类型
- 单体内置对象

①Object 类型

js 中的 Object 类型通常用来存储和输出数据，同时可以用来设置属性和方法。

新建 Object 类型

```
var obj1 = new Object();
var obj2 = {
    name:'星星课堂',
    'num':100,
    5:'数字属性',
    getSum:function(a,b){return a + b;},
    'is-number':true
}
```

增加删除实例属性和方法

```
obj1.name = '周小周';

obj1.num = 200;

obj1.getDec = function(a,b){

    return a - b;

}

delete obj1.num;
```

访问属性和使用方法

点表示法：只能通过属性名称来访问属性和使用与方法。

```
console.log(obj2.name);

console.log(obj2.getSum(1,2));
```

方括号表示法：可以通过变量来访问属性，属性名称中也可以包含非字符和非数字。

```
var otherNum = 'num';

console.log(obj2['num']);//属性名称可以访问

console.log(obj2[otherNum]);//变量名称也可以访问


console.log(obj2[5]);//可以访问

console.log(obj2.5);//报错


console.log(obj2['is-number']);//可以访问

console.log(obj2.is-number);//报错
```

②Array 类型

数组的作用是使用单独的变量名来存储一系列的值。js 中的数组每个元素可以保存任意类型的数据，因此可以把数组理解成一个大仓库。数组中每个元素都对应一个索引下标，这个索引下标是从 0 开始计数的 ,因此数组的最后一个元素 ,它的索引下标是数组长度减去 1。js 数组中最多内包括 4294967295 个元素。

- **数组的基本用法**

  1.新建数组

  new 关键字方式：

  var arr=new Array(2)

  arr[0]="xingxingketang"

  arr[1]=100

  var arr=new Array("星星课堂",100,true);

  字面量方式：

  var arr = ['星星课堂',200,{x:100},true];

  2.数组长度

  获取数组的成员数量。

  console.log(arr.length)

  3.访问数组元素

  通过指定数组名以及索引下标，你可以访问数组中对应的某个特定的元素。

  console.log(arr[1]);//获取第二个元素

4.修改数组元素

修改已有数组中的值，只要修改指定索引下标对应的值即可

```
arr[0]="xingxingketang";//修改第一个元素
```

- **检测数组**

对于引用类型的数据检测，依靠 typeof 不能满足需求，因此需要使用 instanceof 操作符以及数组的 isArray 方法来解决这个问题，它们的判断结果返回的都是布尔值。

```
console.log(arr instanceof Array);
console.log(Array.isArray(arr));
```

- **数组转换**

toString()　把数组转换为字符串，并返回转换之后的字符串。

toLocaleString()　　把数组转换为本地数组，并返回转换之后的本地环境字符串。

valueOf()　返回数组对象的原始值。

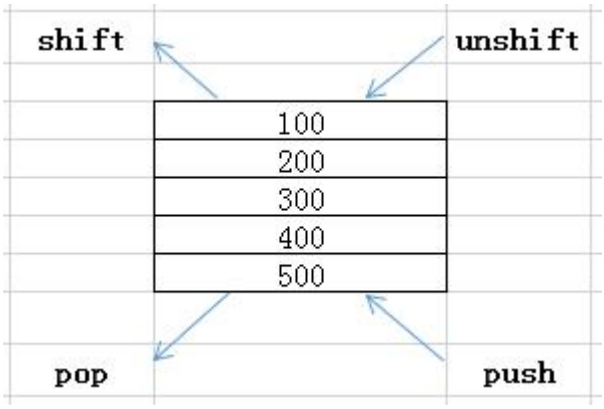
join()　把数组的所有元素拼接成一个字符串，数组元素通过指定的分隔符进行分隔。

toLocaleString()会根据你的本地环境来返回字符串，它和 toString()返回的值在不同的本地环境下使用的时候可能会有不同变化，因此尽量使用 toString()方法。

如果数组中有的值是 null 或者 undefined，那么转换之后的字符串中会以空字符串来表示。

```
var arr = [1,2,undefined,3,null];
console.log(arr.join(','));
```

- **数组栈与队列方法**



栈方法是一种后进先出的方式，js 中数组就是这么一种栈的概念。栈方法其实就是最新添加的元素会被最早的删除掉。

列队方法就是先进先出的方式，其实就是在数组的末端添加新的元素，从数组的前端删除之前的元素元素。

push()　向数组的末尾添加一个或更多元素，并返回新的长度。

pop()　删除并返回数组的最后一个元素。

unshift()　向数组的开头添加一个或更多元素，并返回新的长度。

shift()　删除并返回数组的第一个元素。

- **数组排序方法**

sort() 从小到大排序，返回排序后的新数组，原数组也会被排序。

```
var arr = [6,9,1,3,11,33,100,0,31];
console.log(box.sort());
```

reverse() 逆向排序，返回排序后的新数组，原数组也会被排序。

```
var arr = [1,2,3,4,5];
console.log(box.reverse())
```

- **数组操作方法**

splice() 向/从数组中添加/删除项目，然后返回被删除的项目。

splice 第一个参数是确定从哪个下标开始删除，

第二个参数是删除元素的个数（如果不传第二个参数，那么默认会删除指定下标位置之后的所有数组元素）

splice 方法会改变原来的数组

splice 方法会返回一个由之前删除元素组成的新数组

splice 方法第三个参数是向数组的指定下标位置添加新的元素

如果第二个参数传 0，那么第三个参数会添加到之前第一个参数下标的位置上，之前的那个元素会自动后移一个位置

```
var arr = [6,9,1,3,11,33,100,0,31];
arr.splice(2,0,"666");
arr.splice(2,2);
```

concat()方法可以合并两个或多个数组并新建一个新数组，该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

```
var arr = [1,2,3,4,5];
arr.concat(666,666);
```

slice() 方法可从已有的数组中返回选定的元素。

slice 方法第一个参数是指定从哪个下标开始取数据，第二个参数是指定取到哪个数组下标为止，

然后返回第一个参数和第二个参数之间的元素，并组成一个新的数组

slice 方法不会改变原数组

```
var arr = [6,9,1,3,11,33,100,0,31];
var newArr = arr.slice(1,3);
```

③Function 类型

1.定义函数

JavaScript 函数是被设计为执行特定任务的代码块，函数会在某代码调用它时被执行，并且可以多次调用执行，函数名可以包含字母、数字、下划线和美元符号（规则与变量名相同）。

函数可能有参数。

函数都可以有返回值。

函数声明的方式定义

js 中通过函数声明定义两个名称相同的函数，后面定义的函数会覆盖前面定义的函数

```
function fn(a, b) {
    return a * b;
}
```

函数表达式定义

```
var fn= function(a, b) {
    return a+ b;
};
```

使用 Function 构造函数

```
var fn= new Function('a', 'b' ,'return a + b');
```

2.调用函数

() 运算符调用函数，访问没有 () 的函数将返回函数定义。

```
fn();
console.log(fn);
```

3.函数参数

函数定义时候的参数是在函数定义中所列的名称，称为形参。函数调用时候的参数是当调用函数时由函数接收的真实的值，称为实参。实参和形参在数量、类型上、顺序上要保持一致， 否则会出现错误。

函数定义参数时不用指定数据类型，函数对传进来的参数不会进行类型检测，函数对传进来的参数的个数不会进行检测。

```
function fn(a,b) {
    return a + b
};
fn(1,2);
```

a，b 就是形参。调用函数 fn(1,2) 1，2 就是实参。

在函数中，参数是局部变量。

函数内部可以通过 arguments 对象来接收传递进来的所有参数集合 ,arguments 对象

的 length 属性可以得到函数执行时传入的参数数量。

```
function fn() {
    console.log(arguments.length);
    return arguments[0] + arguments[1];
}
console.log(fn(100,200,300,400,500,600));
```

4.函数返回值

JavaScript 函数指定到 return 语句的时候，函数将停止执行，return 之后的代码将不会继续执行。函数会得出返回值，并会返回给调用者。

```
function fn(a,b) {
    return a + b
};
var num = fn(1,2);
```

5.函数的属性与方法

length 属性

length 属性表示函数要接收的参数的个数。

```
function fn(a, b) {
    alert(a+ b);
}
console.log(fn.length);
```

prototype 属性

js 中的每个函数都有 prototype 属性，它返回对象类型原型的引用，保存所有实例方法，也就是原型。toString()等方法实际上都保存在 prototype 下，只不过是通过各自的对象实例来访问。

```
console.log(fn.prototype);
```

apply()方法与 call()方法

apply()方法接收两个参数：第一个参数是调用该函数的执行对象，第二个是参数数组。其中。

```
function fn1(a, b) {
    console.log(a+ b);
}
```

第二个参数可以是数组

```
var obj = {x:100,y:200,z:300};
fn1.apply(obj,[1,2]);
```

第二个参数也可以是 arguments 对象

```
function fn2(c, d) {
    return fn1.apply(this, arguments);
}
fn2(1,2);
```

call()方法与 apply()方法的作用相同，区别在于接收参数的方式不同。call()方法而言，第一个参数与 apply()方法相同，不同的是函数的参数需要逐个列举出来传递给函数。

```
var obj = {x:100,y:200,z:300};

fn1.call(obj,1,2);
```

apply()方法与 call()方法它们主要是扩展函数的调用执行作用域，也就是说，obj 对象和 fn1()方法之间没有多少关联，也就是对象不用和函数方法发生任何耦合，比如 obj.fn1= fn1。

6.函数内的属性

this

this 引用的是函数执行的环境对象，当函数被调用时，此时的 this 对象引用的是当前调用函数的对象。**简单来说就是谁调用函数，函数的内部属性 this 就代表谁**。当在全局作用域中调用函数时，this 代表的就是 window。

```
fn1()//window

obj.fn1()//obj

var f1 = new fn1()//f1

fn1.call(obj)//obj
```

arguments.callee

arguments 对象有一个 callee 的属性，这个属性是一个指针，指向拥有这个 arguments 对象的函数本身。

函数内部有可能会调用自身，调用自身的函数定义可以认为是递归函数。

```
function fn1(a) {
    if (a <= 1) {
        return 1;
    } else {
        return a * fn1(a - 1);
    }
}
console.log(fn1(3)) ;
```

加入改变函数名 fn1，内部的自身调用就要修改，因此可以使用 arguments.callee 来代替函数本身来执行递归调用。

```
function fn1(a) {
    if (a <= 1) {
        return 1;
    } else {
        return a * arguments.callee(a - 1);
    }
}
console.log(fn1(3));
```

④Date 类型

Date 对象用于处理日期和时间，通过操作 Date 对象我们可以获取和设置时间信息。

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

不传递参数的时候会自动获取当前的时间和日期

```
var dateTime = new Date('2019/03/25 11:30:23');
```

传递参数的时候会获取指定的时间和日期

ps：这个数据在不同浏览器上会有不同的显示效果

1.时间戳

日期类型在 js 中也会以时间戳的形式出现和使用，时间戳是指从格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒**(北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒)**起到现在的毫秒数，这个基准时间也被称为国际协调时间**（UTC）**，或者说是计算机元年时间。时间戳位数是 10 位数的时候要*1000，时间戳位数是 13 位数的话可以直接使用转换。

获取时间戳（时间转时间戳）

```
var dateTime = new Date();
var timestamp = dateTime.getTime();
console.log(timestamp);//1479785483000
```

获取时间（时间戳转时间）

```
var dateTime = new Date(1479785483000);
console.log(dateTime);//Tue Nov 22 2016 11:31:23 GMT+0800 (中国标准时间)
```

ps：时间戳用来比较大小非常方便

2.获取 Date 中的指定格式日期时间

getDate()　从 Date 对象返回一个月中的某一天 (1 ~ 31)。

getDay()　从 Date 对象返回一周中的某一天 (0 ~ 6)。

getMonth() 从 Date 对象返回月份 (0 ~ 11)。

getFullYear()从 Date 对象以四位数字返回年份。

getHours() 返回 Date 对象的小时 (0 ~ 23)。

getMinutes()　返回 Date 对象的分钟 (0 ~ 59)。

getSeconds()　返回 Date 对象的秒数 (0 ~ 59)。

3.简单获取指定格式显示时间的函数方法

```
function getTime(timestamp){
    var date = new Date(timestamp);
    var year = date.getFullYear();
    var month = date.getMonth() + 1;
    var day = date.getDate();
    var hour = date.getHours();
    var min = date.getMinutes();
    var sec = date.getSeconds();
    return year+'-'+month+'-'+day+' '+hour+':'+min+':'+sec;
}
console.log(getTime(1479785483000));//2016-11-22 11:31:23
```

⑤RegExp 类型

RegExp 对象表示正则表达式，被用来匹配、搜索、替换那些符合某个指定规则的文本。主要用来验证用户提交信息是否符合指定规则，过滤字符串信息等作用。

test 检索字符串中指定的值是否符合指定规则，返回 true 或 false。

1.RegExp 对象实例的创建

new 操作符方式

```
var rp = new RegExp();//不指定里面的规则可以匹配任何值
var rp = new RegExp('xxkt');//只能匹配包含 xxkt 的字符串
```

字面量方式

```
var rp = /xxkt/;//字面量的形式
```

第二个参数是一个可选的字符串，包含属性 "g"、"i"，分别用于指定全局匹配、区分大小写的匹配。

i 忽略大小写的影响

g 全局匹配，查找所有匹配而非在找到第一个匹配后停止

```
var rp = new RegExp('xxkt','ig');
var rp = /xxkt/ig;
```

2.RegExp 对象方法

exec 检索字符串中指定的值。返回找到的值，并确定其位置。

test 检索字符串中指定的值。返回 true 或 false。

3.支持正则表达式的 String 的方法

search 检索与正则表达式相匹配的值。

match 找到一个或多个正则表达式的匹配。

replace 替换与正则表达式匹配的子串。

split 把字符串分割为字符串数组。

```
var str = 'xxkt,123,xxkt';

var rp = /xxkt/ig;

var rp1 = /,/ig;

console.log(str.match(rp));//["xxkt", "xxkt"]

console.log(str.search(rp));//0

console.log(str.replace(rp,'星星课堂'));//星星课堂,123,星星课堂

console.log(str.split(rp1));//["xxkt", "123", "xxkt"]
```

4.匹配字符

方括号

方括号用于查找某个范围内的字符。

[abc] 查找方括号之间的任何字符。

[^abc] 查找任何不在方括号之间的字符。

[0-9] 查找任何从 0 至 9 的数字。

[a-z] 查找任何从小写 a 到小写 z 的字符。

[A-Z] 查找任何从大写 A 到大写 Z 的字符。

[A-z] 查找任何从大写 A 到小写 z 的字符。

(red|blue|green) 查找任何指定的选项。

元字符

元字符是拥有特殊含义的字符。

. 查找单个字符，匹配除换行符外的任意字符

\d 查找匹配数字

\D 查找匹配非数字

\w 查找匹配字母和数字及_

\W 查找匹配非字母和数字及_

\s 查找匹配空白字符。

\S 查找匹配非空白字符。

\b 查找匹配空格字符

\n 查找换行符

匹配量词

n+ 匹配任何包含至少一个 n 的字符串。

n* 匹配任何包含零个或多个 n 的字符串。

n? 匹配任何包含零个或一个 n 的字符串。

(abc)+ 匹配至少一个(abc)

n{a,b} 匹配最少 a 个、最多 b 个 n

n$ 匹配任何结尾为 n 的字符串，从行首开始匹配。

^n 匹配任何开头为 n 的字符串，必须匹配到行尾。

查找匹配身份证号码是否符合规则

身份证号码为 15 位或者 18 位，15 位肯定全部是数字，18 位前 17 位肯定为数字，最后一位可能是数字也有可能是字符 x。

```
var rp= /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/;

var str = 340403198506030263;

console.log(rp.test(str));
```

⑥基本包装类型

按照常理说基本类型是不应该有方法的，方法应该是对象才有的，因此为了便于操作基本类型值，ECMAScript 提供了 3 个特殊的引用类型：Boolean、Number 和 String，并为这三个应用类型提供了属性与方法。

使用方式是通过 new 运算符来新建基本包装类型对象，不过通常是使用字面量的形式。

Boolean 类型

只是通过 new 操作符定义一个布尔值，没有其他属性和方法。

Number 类型

属性：之前学习过的最大值、最小值、正无穷、负无穷。

方法：

toFixed()　　保留浮点数小数点后指定位数并转成字符串

String 类型

属性：之前学习的长度属性。

方法：

concat(a,b) 连接字符串，把 a,b 字符串合并到到调用该方法的字符串

charAt(n) 查找并返回指定索引下标位置的字符

charCodeAt(n) 查找并以 Unicode 编码形式返回指定索引下标位置的字符

slice(a,b) 提取字符串的片断，并在新的字符串中返回下标 a 到下标 b 的字符串

substring(a,b) 提取字符串中两个指定的索引号之间的字符

substr(a,b) 从起始索引号提取字符串中指定数目的字符

toLocaleLowerCase() 把字符串转换为小写。

toLocaleUpperCase()把字符串转换为大写。

toLowerCase() 把字符串转换为小写。

toUpperCase() 把字符串转换为大写。

ps：需要注意的是，JavaScript 的字符串是不可变的，String 类型定义的方法都不能改变字符串的内容。他们只是返回全新的字符串，而不是修改原来的字符串。

⑦单体内置对象

内置对象：由 ECMAScript 提供的、不需要依赖于宿主环境的对象，这些对象在 ECMAScript 程序执行之前就已经存在了。

Global

Global 作为 js 的全局对象，这个对象无法直接访问的，但是在浏览器中浏览器是将这个对象当做是 window，全局变量和函数都是 Global 对象的属性。

属性：Infinity、NaN、undefined

方法：

encodeURL()　　用于整个 url 且改方法不会对特殊字符进行编码

encodeURLComponent() 用于 url 的某一个片段，且会对任何非标准字符进行编码

decodeURI() 非特殊解码

decodeURIComponent() 特殊解码

```
var url = "http://www.xingxingclassroom.com/index/课程列表";
var encodedUrl1 = encodeURI(url);
var encodedUrl2 = encodeURIComponent(url);
```

evel()

该函数可以计算并执行以字符串表示的 JavaScript 代码，尽量不要用，某些情况下也不会执行。

Math

Math 对象用于执行数学任务，Math 对象并不像 Date 和 String 那样是对象的构造函数，因此没有 new Math()，Math.sin() 这样的函数只是函数，不是某个对象的方法。通过把 Math 作为对象使用就可以调用其所有属性和方法。

属性：

E　返回算术常量 e，即自然对数的底数（约等于 2.718）。

LN2返回 2 的自然对数（约等于 0.693）。

LN10　返回 10 的自然对数（约等于 2.302）。

LOG2E　返回以 2 为底的 e 的对数（约等于 1.414）。

LOG10E 返回以 10 为底的 e 的对数（约等于 0.434）。

PI　返回圆周率（约等于 3.14159）。

SQRT1_2　　返回返回 2 的平方根的倒数（约等于 0.707）。

SQRT2　返回 2 的平方根（约等于 1.414）。

方法：

abs(x)　返回数的绝对值。

acos(x)　返回数的反余弦值。

asin(x)　返回数的反正弦值。

atan(x)　以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。

atan2(y,x)　　返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。

ceil(x)　对数进行上舍入。

cos(x)　返回数的余弦。

exp(x)　返回 e 的指数。

floor(x)　对数进行下舍入。

log(x)　返回数的自然对数（底为 e）。

max(x,y)返回 x 和 y 中的最高值。

min(x,y) 返回 x 和 y 中的最低值。

pow(x,y)　　返回 x 的 y 次幂。

random()　　返回 0 ~ 1 之间的随机数。

round(x)　　把数四舍五入为最接近的整数。

sin(x)　返回数的正弦。

sqrt(x)　返回数的平方根。

tan(x)　返回角的正切。

谢谢观看！

如果你有自己难以解决的问题，请发送邮件到 xingxingclassroom@aliyun.com，并备注

星星课堂问题咨询这几个字+问题介绍为邮件标题，谢谢。

我是星星课堂老师：周小周