JavaScript进阶

---JS数据类型、值与类型转换

---Boolean、Number、String进阶

河北师范大学软件学院
Software College of Hebei Normal University

JavaScript进阶

---JS数据类型、值与类型转换

参见《深入理解JS》第8章、《JS权威指南》第3章

河北师范大学软件学院
Software College of Hebei Normal University

内容刚要

- JS的数据类型
- 不同类型的值
- 数据类型转换

数据类型（参考《深入理解JS》8.1节）

- JavaScript（ES5）数据类型（6种）及其划分（2类）
  - 基本（原始）类型（Number、String、Boolean、Null、Undefined）
  - 引用（对象）类型（Object（Array、Function、Date、Error等））

注意：定义为引用类型的变量，其引用分配在栈区或堆区，引用的对象分配在堆区

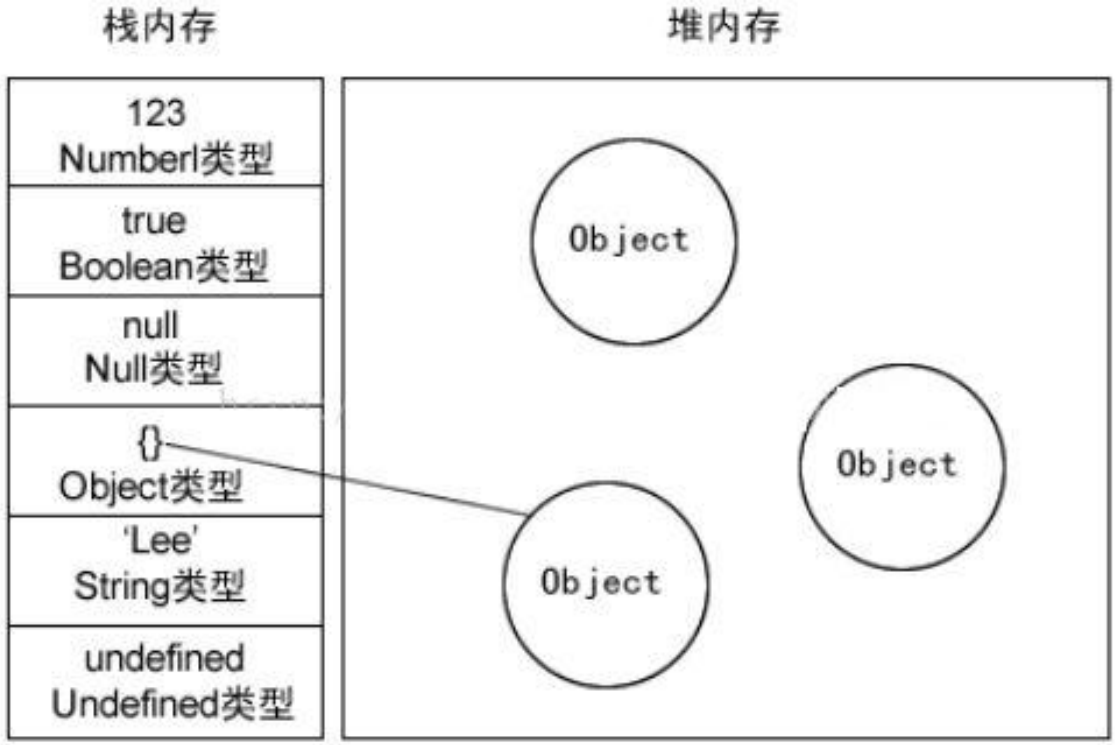
注意：定义为基本类型的函数局部变量分配在栈区

- 数据类型检测方法（typeof、instanceof）参见实例demo01

参考链接：http://www.jianshu.com/p/75057391ad51

基本类型与引用类型的区别

- 内存分配方式不同
  - 堆区与栈区、存值与存地址、影响变量的生命周期（自动清除、垃圾回收）
  - 函数内定义的基本数据类型的临时变量分配在栈区
  - 引用数据类型的变量的引用（地址）存储在栈区或堆区，被引用（指向）的对象存储在堆区



思考：对象的属性如果是基本类型，那么该属性是分配在堆区还是栈区

栈区常用来存储函数局部临时变量，一般数据量较小
堆区常用来存储更为复杂的数据结构的对象

参见实例demo02

基本类型与引用类型的区别

- 赋值时不同　　参见实例demo03
  - 赋值、赋地址、深拷贝与浅拷贝
- 判等时的不同　　参见实例demo04
  - 值类型是判断变量的值是否相等（值比较）
  - 引用类型是判断所指向的内存空间是否相同（引用比较）
- 函数参数传递时的不同　　参见实例demo05
  - 值传递
  - 引用传递

注意：真正决定这几种不同的是数据类型，而不是内存分配方式，内存分配方式决定的是变量的生命周期

内容刚要

- JS的数据类型
- 不同类型的值
- 数据类型转换

基本数据类型的值（原始值、参考教程8.2、8.3节）

- Number类型的值

  参见实例demo06

  - 整数与浮点数
  - NaN、Infinity、-Infinity、+0、-0

- String类型的值

  参见实例demo07

  - 空字符、字符和字符串、转义字符

- Boolean类型的值

  参见实例demo08

  - true、false

- Null与Undefined

  -null、undefined

引用数据类型的值（对象、参考教程8.2节）

- 简单对象
  - 例：var obj = {name： “Jack”， age：20};
- 数组
  - 例：var arr = [1,2,true,“Hi”];
- 函数对象　　参见实例demo09
  - 例：var foo = function(x,y){...};//函数也是对象（可执行的对象），也有属性和方法
- 正则对象
  - 例：var reg = /^a+b+$/;

包装对象（参见《深入理解JS》8.4节）

- 包装对象
  - 数字、布尔、字符串等基本数据类型都有对应的包装对象类型，可以将其包装成对象
  - 例：new Number(20)； new String('SomeStr');//装箱
  - 存储或读取基本类型（字符串、数字、布尔）值的属性时，会创建临时包装对象
  - 例: console.log('Hello，World'.length);
  - 基本类型其属性不能被改变、添加或删除（原始值不可变性）
- 临时对象在使用之后立即释放
  - 例：var str=”test”；

    str.p = 4；//设置临时对象属性

    var t = str.p; // 临时对象已释放，t为undefined

参见实例demo10

内容刚要

- JS的数据类型
- 不同类型的值
- **数据类型转换**

JavaScript数据类型转换

- 其他类型转换为Boolean类型

值	转换成的布尔值
undefined	False
null	False
布尔值	与输入相同（不用转换）
数字	0，NaN 转换成 false，其他数字转换成 true
字符串	"转换成 false，其他字符串转换成 true
对象	总是为 true

- 转换方式
  - Boolean（）、value? true：false、！！value

参见实例demo11 Part1

JavaScript数据类型转换

- 其他类型转换为Number类型

值	结　果
undefined	NaN
null	0
布尔值	false 转换成 0，true 转换成 1
数字	保持不变（没什么好转换的）
字符串	解析字符串中的数字（忽略开头和结尾的空格）；空字符转换成 0。比如 '3.141'转换成 3.141
对象	调用 ToPrimitive(value, number)（参见 8.5.3“算法：ToPrimitive() ——将值转换为原始值”）并转换生成的原始类型

- 转换方式
  - Number（）、+value、parseFloat、parseInt

参见实例demo11 Part2

JavaScript数据类型转换

- 其他类型转换为String类型

值	结　果
undefined	'undefined'
null	'null'
布尔值	false->'false' true->'true'
数字	(例如，3.141->'3.141')
字符串	输出即输入（无须转换）
对象	调用 ToPrimitive(value,String)（参见 8.5.3“算法：ToPrimitive()——将值转换为原始值”）并将原始值结果转换为字符串

- 转换方式

  - String（）、 ‘’ +value、 value.toString();

参见实例demo11 Part3

JavaScript数据类型转换

值	转换为：			
	字符串	数字	布尔值	对象
undefined	"undefined"	NaN	false	throws TypeError
null	"null"	0	false	throws TypeError
true	"true"	1		new Boolean(true)
false	"false"	0		new Boolean(false)
""(空字符串)		0	false	new String("")
"1.2"(非空,数字)		1.2	true	new String("1.2")
"one"(非空,非数字)		NaN	true	new String("one")
0	"0"		false	new Number(0)
-0	"0"		false	new Number(-0)
NaN	"NaN"		false	new Number(NaN)
Infinity	"Infinity"		true	new Number(Infinity)
-Infinity	"-Infinity"		true	new Number(-Infinity)
1(无穷大,非零)	"1"		true	new Number(1)
{}(任意对象)	参考3.8.3节	参考3.8.3节	true	
[](任意数组)	""	0	true	
[9](1个数字元素)	"9"	9	true	
['a'](其他数组)	使用join()方法	NaN	true	
function(){}(任意函数)	参考3.8.3节	NaN	true	

JavaScript数据类型转换

- 隐式类型转换　　参见实例demo12 Part1
  - 使用关系运算符时的转换（==、>、<、引用类型和基本类型比较时）
  - 使用算数运算符时的转换（'img'+ 3 + '.jpg';　“25”　-0;）
  - 使用逻辑运算符时的转换（ !!0; ）
  - 执行流程语句时的转换（if(obj){...}）
- 显式类型转换（使代码更清晰）　　参见实例demo12 Part2
  - Boolean（）、Number（）、String（）、Object（）　参见权威教程50-54页
  - 数转为字符串（toString()、toFixed()、toPrecision()、toExponential()）
  - 字符串转为数字（parseInt()、parseFloat()）
  - 对象转换为原始值（toString()、valueOf()）

Have a Break!

思考

- 类对象与内置对象
- Object、Array、Date等（是对象？、是构造函数？、是类型？）
- Math、JSON（是对象？、是构造函数？、是类型?）

- console.log(typeof Boolean);
- console.log(typeof Number);
- console.log(typeof String);

JavaScript进阶

---Boolean、Number、String进阶

参见《深入理解JS》第10章、11章、12章

河北师范大学软件学院
Software College of Hebei Normal University

内容刚要

- **Boolean**
- **Number**
- **String**

Boolean进阶

- 所有对象都是真值

```
> Boolean({});
< true
> Boolean([]);
< true
> Boolean(new Boolean(false));
< true
> Boolean(function foo(){});
< true
```

- Boolean函数的使用
  - 类型转换、实例化布尔对象（原始值的包装对象）

其他部分参见《深入理解JS》第10章

内容刚要

- Boolean
- Number
- String

Number进阶

- Number构造器属性（静态属性）
  - Number.MAX_VALUE、Number.MIN_VALUE
  - Number.NaN、Number.NEGATIVE_INFINITY、Number.POSITIVE_INFINITY
- Number原型方法(Number对象继承的方法) 参见实例demo13
  - Number.prototype.toFixed(...)、Number.prototype.toPrecision(...)
  - Number.prototype.toString(...)、Number.prototype.toExponential(...)
- Number函数的使用
  - 类型转换、实例化Number对象（原始值的包装对象）

其他部分参见《深入理解JS》第11章

内容刚要

- Boolean
- Number
- String

String进阶

- 字符串比较
  - "B" > "A"、"B" > "a"
  - "B".localeCompare("A"); //考虑本地化的字符排序，返回0或非0
- 字符串拼接
  - 合并：加号运算符（+、+=）
  - 合并：拼接字符串数组（通过数组方法push、join等）
- String函数的使用
  - 类型转换、实例化字符串对象

参见实例demo14 Part0

String进阶

- 字符串构造器方法（静态方法）
  - String.fromCharCode(97,98,99);
  - String.fromCharCode.apply(null,[97,98,99]);
- 字符串原型方法（String对象继承的方法）之提取字符串
  - String.prototype.charAt(pos);
  - String.prototype.charCodeAt(pos);
  - String.prototype.slice(start,end?);
  - String.prototype.substr(start,length?)
  - String.prototype.substring(start,end?);
  - String.prototype.split(separator?,limit?);

参见实例demo14 Part1

String进阶

- 字符串原型方法（String对象继承的方法）之字符串变换
  - String.prototype.trim( );
  - String.prototype.concat(str1?,str2?);
  - String.prototype.toLowerCase( ); String.prototype.toLocaleLowerCase( );
  - String.prototype.toUpperCase( ); String.prototype.toLocaleUpperCase( );

参见实例demo14 Part2

- 字符串原型方法（String对象继承的方法）之检索和比较
  - String.prototype.indexOf(searchingString,position?);
  - String.prototype.lastIndexOf(searchingString,position?);
  - String.prototype.localeCompare(other);

参见实例demo14 Part3

String进阶

- 字符串原型方法（String对象继承的方法）之可正则方法
  - String.prototype.search(regexp);
  - String.prototype.match(regexp);
  - String.prototype.replace(regexp);

参见实例demo14 Part4

其他部分参见《深入理解JS》第12章

Thank You!

作业

- 复习本章课件及练习
- 完成freecodecamp在线学习网站中JS基础任务的50%