



中山大學
SUN YAT-SEN UNIVERSITY

《系统分析与设计》

课程设计：互联网电影售票系统

年 级	2014 级
学 院	数据科学与计算机学院
专 业	软件工程
组 号	
组 名	可达鸡不是鸭

版本更新表

日期	版本号	描述	作者
2017.4.8	1.0	编码规范	李文盛

编码规范

一、数据类型

1.原始类型：当你给一个原始类型赋值时，返回的是这个值的本身。

- string
- number
- boolean
- null
- undefined

2.对象类型：当你给一个对象类型赋值时，返回的是这个值的引用。

- Object
- Array
- Function

二、对象

1.新建一个对象的语法

```
//不推荐
var item = new Object();

//推荐
var item = {};
```

2.不要使用保留字作为键值，否则在IE8下面会出现问题(详情)。

```
//不推荐
var superman = {
  default: { clark: 'kent'},
  private: true
};

//推荐
var superman ={
  defaults: { clark: 'kent'},
  hidden: true
};
```

3.使用可读性强的同义词代替保留字

```
//不推荐
var superman = {
  class: 'alien'
};

//不推荐
var superman = {
  klass: 'alien'
};

//推荐
```

```
var superman = {  
  type: 'alien'  
};
```

三、数组

1.新建一个数组的语法

```
//不推荐  
var items = new Array();  
//推荐  
var items = [];
```

2.如果你不知道数组的长度可以使用push将元素加入。

```
var someStack = [];  
//不推荐  
someStack[someStack.length] = 'something';  
//推荐  
someStack.push('something');
```

3.当你需要复制一个数组的时候使用slice。jsPerf

```
var len = items.length,  
    itemsCopy = [],  
    i;  
//不推荐  
for (i = 0; i < len; i++){  
  itemsCopy[i] = items[i];  
}  
//推荐  
itemsCopy = items.slice();
```

4.用slice转换伪数组对象到数组

```
function trigger() {  
  var args = Array.prototype.slice.call(arguments);  
  ...  
}
```

四、String类型

1.使用单引号"

```
//不推荐  
var name = "Bob Parr";  
//推荐  
var name = 'Bob Parr';  
  
//不推荐  
var fullName = "Bob " + this.lastName;  
//推荐  
var fullName = 'Bob ' + this.lastName;
```

2.当字符串长度超过80个时，应该通过字符串连接多行显示。

注意：过度使用字符串连接将会影响性能。

3.当程序建立一个字符串时， 使用join代替字符串连接。特别是在IE下。

函数

五、函数表达式：

1.匿名函数表达式

```
var anonymous = function(){  
    return true;  
}
```

2.命名函数表达式

```
var named = function named() {  
    return true;  
};
```

3.立即执行的函数表达式（IIFE）

```
(function(){  
    console.log('Welcome to the Internet. Please follow me.');})();
```

4.不要将函数声明放在如if/while循环或其他任何语句中。但可以用函数表达式来替代函数声明这么做。一些浏览器可能的确可以在语句中使用函数声明。但是在解析方面的处理各不相同，各种浏览器下兼容性很不好。

5.注意: ECMA-262定义了一系列的语句，但是函数声明并没有被归类为真正的语句。关于这点可查看ECMA-262的文档。

```
//不推荐  
if (currentUser){  
    function test() {  
        console.log('Nope.');    }  
}  
  
//推荐  
if (currentUser){  
    var test = function test() {  
        console.log('Yup.');    }  
}
```

6.arguments 不能作为一个参数的名字， 因为这会覆盖每一个函数内的arguments对象。

六、属性

1.访问一个属性时，使用点的形式取值。

```
var luke = {  
    jedi: true,
```

```
    age: 28
  };
  // 不推荐
  var isJedi = luke['jedi'];
```

```
  // 推荐
  var isJedi = luke.jedi;
```

2. 需要一个变量访问一个属性时，使用“[]”来取值。

```
var luke = {
  jedi: true,
  age: 28
};

function getProp(prop) {
  return luke[prop];
}
```

```
var isJedi = getProp('jedi');
```

七、变量

1. 总是使用 `var` 来定义变量。如果不这么做将定义一个全局变量出来。我们希望避免全局命名空间的污染。

2. 使用一个 `var` 声明多个变量，并且每声明一个变量就换一行。

声明多个变量时，把不赋值的变量放在后面。这样做是有好处的，如果日后你想给未赋值变量赋值的时候，可能要引用到上面已经赋值的变量。

3. 在一个作用域的顶部给一个变量赋值。这样有助于避开，变量声明和声明提前的分配问题。

八、声明提前

1. 不管你在何处给一个变量声明或赋值，`javascript` 解析器都会事先在作用域的顶端做声明提前（`Hoisting`）。

```
// 我们知道下面将不能正常运行（假设没有全局变量）
function example() {
  console.log(notDefined); // => 抛出一个引用错误
}
```

// 在引用这个变量之后，给这个变量赋值将不会抛异常，这是因为 `javascript` 解析器有声明提前。

// 注意：赋的“`true`”值，不会被提前。

```
function example() {
  console.log(declaredButNotAssigned); // => undefined
  var declaredButNotAssigned = true;
}
```

// javascript解析器,会在作用域的顶部提前声明变量。

// 用代码描述出来, 其实就等同于下面这种情况。

```
function example() {  
  var declaredButNotAssigned;  
  console.log(declaredButNotAssigned); // => undefined  
  declaredButNotAssigned = true;  
}
```

匿名函数表达式将该变量名做了提前声明, 没有给该变量赋值函数。

```
function example() {  
  console.log(anonymous); // => undefined  
  
  anonymous(); // => 抛出异常, anonymous 不是一个函数  
  
  var anonymous = function() {  
    console.log('anonymous function expression');  
  };  
}
```

2.和匿名一样, 有名函数表达式将该变量名做了提前声明, 没有给该变量赋值函数名和函数体。

```
function example() {  
  console.log(named); // => undefined  
  
  named(); // => 抛出异常, named 不是一个函数  
  
  superPower(); // => 抛出异常, superPower 没定义  
  
  var named = function superPower() {  
    console.log('Flying');  
  };  
}
```

// 把函数名改成和变量名一样, 也得出同样的结果。

```
function example() {  
  console.log(named); // => undefined  
  
  named(); // => 抛出异常, named 不是一个函数  
  
  var named = function named() {  
    console.log('named');  
  };  
}
```

函数声明会将函数名和函数体声明提前。

```
function example() {  
  superPower(); // => Flying
```

```
function superPower() {  
    console.log('Flying');  
}  
}
```

九、条件表达式与等式

1.使用 `===` 和 `!==` 代替 `==` 和 `!=`。

2.条件表达式 会通过 `ToBoolean` 来进行强制转化，而且遵循以下的规则：

- 对象被转化为`true`
- `Undefined`被转化为`false`
- `Null`被转化为`false`
- 布尔值被转化为相应的布尔值
- 数字当值为`**+0**`,-0或`NaN`时转化为`false`，其他的转化为`true`
- `Strings`类型如果为空时转化为`false`,否则转化为`true`

十、使用快捷方式

```
// 不推荐  
if (name !== '') {  
    // ...stuff...  
}
```

```
// 推荐  
if (name) {  
    // ...stuff...  
}
```

```
// 不推荐  
if (collection.length > 0) {  
    // ...stuff...  
}
```

```
// 推荐  
if (collection.length) {  
    // ...stuff...  
}
```

十一、块

1.给多行的块，使用大括号

十二、注释

1.使用 `/**...*/` 进行多行注释。注释要包括描述、指定类型、参数值和返回值。

2.使用 // 进行单行注释。注释单独占一行，并写在需要注释对象的上面。在注释的上面留一个空行。