02_JavaScript常量与变量

一.JavaScript常量

在编程语言中，一般固定值称为常量，如 3.14。

1.数字（Number）常量

可以是整数或者是小数，或者是科学计数(e)。
如3.14,1001,123e5

2.字符串（String）常量

可以使用单引号或双引号:
"John Doe" 'John Doe'

3.表达式常量

用于计算：5 + 6 5 * 10

4.数组（Array）常量

定义一个数组:
[40, 100, 1, 5, 25, 10]

5.对象（Object）常量

定义一个对象:
{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

6.函数（Function）常量

定义一个函数:
function myFunction(a, b) { return a * b;}

二.JavaScript 变量

JavaScript是弱类型编程语言,定义变量都使用 var 定义,与 Java这种强类型语言有区别.
在定义后可以通过typeOf()来获取JavaScript中变量的数据类型.

1.变量介绍

在编程语言中，变量用于存储数据值。
JavaScript 使用关键字 var 来定义变量， 使用等号来为变量赋值：

```
var x, length
x = 5
length = 6
```

变量可以通过变量名访问。在指令式语言中,变量通常是可变的,常量是一个恒定的值。

注意:

变量是一个名称,常量是一个值。

2.变量命名规则

变量必须以字母开头;
变量也能以 $ 和 _ 符号开头（不过我们不推荐这么做）;
变量名称对大小写敏感（y 和 Y 是不同的变量）.

注意:

JavaScript 语句和 JavaScript 变量都对大小写敏感。

3.JavaScript 数据类型

字符串（String）、数字(Number)、布尔(Boolean)、数组(Array)、对象(Object)、空（Null）、未定义（Undefined）。

3.1.JavaScript 拥有动态类型

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

```
var x; // x 为 undefined
var x = 5; // 现在 x 为数字
var x = "John"; // 现在 x 为字符串
```

3.2.字符串

字符串是存储字符（比如 "Bill Gates"）的变量。
字符串可以是引号中的任意文本。您可以使用单引号或双引号：

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

```
var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
```

3.3.数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

```
var x1=34.00; //使用小数点来写
var x2=34; //不使用小数点来写
```

3.4.布尔

布尔（逻辑）只能有两个值：true 或 false。

```
var x=true;
var y=false;
```

3.5.数组

下面的代码创建名为 cars 的数组：

```
var cars=new Array();
cars[0]="Saab";
cars[1]="Volvo";
cars[2]="BMW";
```

或者 (condensed array):

```
var cars=new Array("Saab","Volvo","BMW");
```

或者 (literal array):

```
var cars=["Saab","Volvo","BMW"];
```

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

3.6.对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

空格和折行无关紧要。声明可横跨多行：

```
var person={
firstname : "John",
lastname : "Doe",
id : 5566
};
```

对象属性有两种寻址方式：

```
name=person.lastname;
name=person["lastname"];
```

3.7.Undefined 和 Null

Undefined 这个值表示变量不含有值。
可以通过将变量的值设置为 null 来清空变量。

```
cars=null;
person=null;
```

3.8.声明变量类型

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

```
var carname=new String;
var x= new Number;
var y= new Boolean;
var cars= new Array;
var person= new Object;
```

注意:

JavaScript 变量均为对象,当声明一个变量时，就创建了一个新的对象。

4.JavaScript变量的声明

在 JavaScript 中创建变量通常称为"声明"变量。
我们使用 var 关键词来声明变量：

```
var carname;
```

变量声明之后，该变量是空的（它没有值）。
如需向变量赋值，请使用等号：

```
carname="Volvo";
```

不过，您也可以在声明变量时对其赋值：

```
var carname="Volvo";
```

一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

```
var lastname="Doe", age=30, job="carpenter";
```

声明也可横跨多行：

```
var lastname="Doe",
age=30,
job="carpenter";
```

Value = undefined

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。
在执行过以下语句后，变量 carname 的值将是 undefined：

```
var carname;
```

重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：
在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

```
var carname="Volvo";
var carname;
```

4.变量的作用域

在 JavaScript 中, 对象和函数同样也是变量。
在 JavaScript 中, 作用域为可访问变量，对象，函数的集合。
JavaScript 函数作用域: 作用域在函数内修改。

4.1.JavaScript 局部作用域

变量在函数内声明，变量为局部作用域。
局部变量：只能在函数内部访问。

```
// 此处不能调用 carName 变量
function myFunction() {
var carName = "Volvo";
// 函数内可调用 carName 变量
}
```

因为局部变量只作用于函数内，所以不同的函数可以使用相同名称的变量。
局部变量在函数开始执行时创建，函数执行完后局部变量会自动销毁。

4.2.JavaScript 全局变量

变量在函数外定义，即为全局变量。
全局变量有 全局作用域: 网页中所有脚本和函数均可使用。

```
var carName = " Volvo";
// 此处可调用 carName 变量
function myFunction() {
// 函数内可调用 carName 变量
}
```

4.3.注意

如果变量在函数内没有声明（没有使用 var 关键字），该变量为全局变量
以下实例中 carName 在函数内，但是为全局变量。

```
// 此处可调用 carName 变量
function myFunction() {
carName = "Volvo";
// 此处可调用 carName 变量
}
```

5.变量生命周期

JavaScript 变量生命周期在它声明时初始化。
局部变量在函数执行完毕后销毁。
全局变量在页面关闭后销毁。

6.函数参数

函数参数只在函数内起作用，是局部变量。

7.HTML中的全局变量

在 HTML 中, 全局变量是 window 对象: 所有数据变量都属于 window 对象。

```
//此处可使用 window.carName
function myFunction() {
carName = "Volvo";
}
```

注意:

我们自己定义的全局变量，或者函数，可以覆盖 window 对象的变量或者函数;
我们自己定义的局部变量，包括 window 对象可以覆盖全局变量和函数。

总结

局部变量：在函数中通过var声明的变量。
全局变量：在函数外通过var声明的变量。
没有声明就使用的变量，默认为全局变量，不论这个变量在哪被使用。

函数内未声明即使用的变量情况：

```
function func(){
undefined_var=110
}
```

在 func() 被第一次调用之前， undefinedvar 变量是不存在的即 undefined。
func() 被调用过之后，undefinedvar 成为全局变量。

8.变量提升

JavaScript 中，函数及变量的声明都将被提升到函数的最顶部。
JavaScript 中，变量可以在使用后声明，也就是变量可以先使用再声明。

变量提升：

函数声明和变量声明总是会被解释器悄悄地被"提升"到方法体的最顶部。

JavaScript初始化的变量不会提升

JavaScript 只有声明的变量会提升，初始化的不会。

注意;

JavaScript 严格模式(strict mode)不允许使用未声明的变量。

三.JavaScript严格模式(use strict)

JavaScript 严格模式（strict mode）即在严格的条件下运行。

使用 "use strict" 指令

"use strict" 指令在 JavaScript 1.8.5 (ECMAScript5) 中新增。
它不是一条语句，但是是一个常量表达式，在 JavaScript 旧版本中会被忽略。
"use strict" 的目的是指定代码在严格条件下执行。
严格模式下你不能使用未声明的变量。

支持严格模式的浏览器:

Internet Explorer 10 +、 Firefox 4+、Chrome 13+、 Safari 5.1+、 Opera 12+。

严格模式声明

严格模式通过在脚本或函数的头部添加 "use strict"; 表达式来声明。

```
<script>
"use strict";
x = 3.14; // 报错 (x 未定义)
</script>
```

为什么使用严格模式?

1.消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
2.消除代码运行的一些不安全之处，保证代码运行的安全;
3.提高编译器效率，增加运行速度;
4.为未来新版本的Javascript做好铺垫。
5."严格模式"体现了Javascript更合理、更安全、更严谨的发展方向，包括IE 10在内的主流浏览器，都已经支持它，许多大项目已经开始全面拥抱它。
6.另一方面，同样的代码，在"严格模式"中，可能会有不一样的运行结果；一些在"正常模式"下可以运行的语句，在"严格模式"下将不能运行。掌握这些内容，有助于更细致深入地理解Javascript，让你变成一个更好的程序员。

严格模式的限制

1.不允许使用未声明的变量;
2.不允许删除变量或对象;
3.不允许删除函数;
4.不允许变量重名;
5.不允许使用八进制;
6.不允许使用转义字符:
7.不允许对只读属性赋值;
8.不允许对一个使用getter方法读取的属性进行赋值;
9.不允许删除一个不允许删除的属性;
10.变量名不能使用 "eval" 字符串;
11.变量名不能使用 "arguments" 字符串;
12.不允许使用以下这种语句:

```
"use strict";
with (Math){x = cos(2)}; // 报错
```

13.由于一些安全原因，在作用域 eval() 创建的变量不能被调用:

```
"use strict";
eval ("var x = 2");
alert (x); // 报错
```

14.禁止this关键字指向全局对象。

保留关键字

为了向将来Javascript的新版本过渡，严格模式新增了一些保留关键字：implements,interface,let,package,private,protected,public,static,yield