函数

- 简介函数语法
- 函数名命名规则
- 函数包含哪几部分
- 函数调用的三种情况
- 简述函数返回值
- 下面代码的执行结果为

```
var x = myFunction(7, 8);
  function myFunction(a, b) {
      return a * b;
  }
```

- 为什么要使用函数
- 简单介绍局部变量和全局变量
- 下面两段代码的执行结果分别是什么 并说明原因

```
function myFunction() {
  var carName = "Volvo";
}
console.log(carName)
```

```
function myFunction() {
  var carName = "Volvo";
  console.log(carName)
}
```

- 函数如何返回值，如何接收
- 下面代码执行结果是什么 为什么

```
function sum(iNum1, iNum2) {
    return iNum1 + iNum2;
  }
```

- 下面代码执行结果是什么 为什么

```
function sum(iNum1, iNum2) {
    console.log(iNum1 + iNum2);
  }
```

- 下面代码执行结果是什么 为什么

```
function sum(iNum1, iNum2) {
    console.log(iNum1 + iNum2);
  }
sum()
```

- 下面代码执行结果是什么 为什么

```
  function sum(iNum1, iNum2) {
      return iNum1 + iNum2;
    }
    var iResult = sum(1,1);
  alert(iResult);
```

- 下面代码执行结果是什么 为什么

```
  function sum(iNum1, iNum2) {
    return iNum1 + iNum2;
    alert(iNum1 + iNum2);
  }
```

- 下面代码的执行结果为

```
function sayHi(sMessage) {
  if (sMessage == "bye") {
    return;
  }

  alert(sMessage);
}
```

- 如何检测参数的个数
- 下面代码的执行结果是多少

```
  function howManyArgs() {
    alert(arguments.length);
  }

  howManyArgs("string", 45);
  howManyArgs();
  howManyArgs(12);
```

- 什么是arguments 对象
- 下面代码的执行结果

```
function sayHi() {
if (arguments[0] == "bye") {
  return;
}

alert(arguments[0]);
}
```

- 下面代码的执行结果是什么

```
function howManyArgs() {
alert(arguments.length);
}

howManyArgs("string", 45);
howManyArgs();
howManyArgs(12);
```

- 什么是函数重载
- 下面代码的执行结果是什么

```
function doAdd() {
  if(arguments.length == 1) {
    alert(arguments[0] + 5);
  } else if(arguments.length == 2) {
    alert(arguments[0] + arguments[1]);
  }
}

doAdd(10);
doAdd(40, 20);
```

- 通过new function() 的系形式重新定义下面的对象

```
function sayHi(sName, sMessage) {
  alert("Hello " + sName + sMessage);
}
```

- Function对象的属性和方法
- 编写一个函数，计算三个数字的大小，按从小到大顺序输出
- 编写一个函数，计算任意两个数字之间所能组成的奇数个数，数字必须是个位数
  比如： 计算0-3之间能组成的奇数个是01、21、03、13、23、31
- 某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的，
  加密规则如下：每位数字都加上5,然后用除以10的余数代替该数字，再将第一位和第四位交换，
  第二位和第三位交换，请编写一个函数，传入原文，输出密文
- 用*实现等边三角形
- 编写一个函数来验证输入的字符串是否是有效的 IPv4 地址。

  如果是有效的 IPv4 地址，返回 "IPv4" ；
  如果不是上述类型的 IP 地址，返回 "Neither" 。

IPv4 地址由十进制数和点来表示，每个地址包含 4 个十进制数，其范围为 0 - 255， 用(".")分割。比如，172.16.254.1；