## 函数的定义

### 函数总共有三种定义方式：

#### 函数式声明（命名函数）：

使用function关键字，

function fnMethodName(x){alert(x);}

##### 函数的执行顺序：

函数式声明中，会存在函数作用域的提升；

**函数声明在JS解析时进行函数提升**，因此在同一个作用域内，不管函数声明在哪里定义，该函数都可以进行调用。

因为：用函数声明创建的函数可以在**函数解析后调用**（解析时进行等逻辑处理）

即：

test1();

function test1() { //函数声明

alert("1111");

}

// “1111”

是因为函数声明实在函数解析完成后调用的，进行了函数提升，test1()才能调用。

#### 2、函数字面量，函数表达式(Function Literals)：

函数表达式的值是在JS运行时确定，并且在表达式赋值完成后，该函数才能调用。

var fnMethodName = function(x){alert(x);}

##### 函数的执行顺序：

函数表达式创建的函数是在运行时进行赋值，且要等到表达式赋值完成后才能调用。

即：

test2();

var test2 = function(){ //函数字面量

alert("2222");

}

//error: test2 is not a function

，，，是因为函数字面量不是函数声明，函数字面量表示方法是在运行时解析的，在调用test2()之前，函数字面量的函数体还没有声明

#### 3、Function()构造函数：

var fnMethodName = new Function(‘x','alert(x);') // 由Function构造函数的参数个数可变。最后一个参数写函数体，前面的参数写入参。

##### 函数的执行顺序：

即：

test3();

var test3 = new Function("alert(3333);");

//error: test3 is not a function

##### ES6中的箭头函数和普通函数有什么区别？

1、**普通函数中的this总是指向调用它的那个对象**，

**箭头函数没有自己的this，他的this永远指向其定义环境**，任何方法都改变不了其指向，如call（）、bind（）、apply（）。（正是因为它没有this，所以也就不能用作构造函数，也没有原型对象）

2、**箭头函数不能当作构造函数**，也就是说，**不能使用new命令**，否则会报错。

3、**箭头函数不能使用yield命令**，因此箭头函数不能用作genertor函数。

4、**箭头函数没有原型属性**。

5、**箭头函数不能使用arguments对象**，该对象在函数体内不存在。如果要用，可以用rest参数代替。

6、**变量提升**：**由于js的内存机制，function的级别最高，而用箭头函数定义函数的时候，需要var(let、const）关键字，而var所定义的变量不能得到变量提升。故箭头函数一定要定义于调用之前。**

**this的指向问题？**

1、**普通函数中，this指向其函数的直接调用者**；

2、**箭头函数中，this指向其定义环境**，任何方法都改变不了其指向，如call（）、bind（）等；

3、构造函数中，如果不使用new,则this指向window，如果使用new创建了一个实例，则this指向该实例。

4、**window内置函数中，如setInterval,setTimeout等，其内部的this指向Window。**

5、**匿名函数的this指向Window**。

6、**apply()、call()、bind()可以改变this的指向**