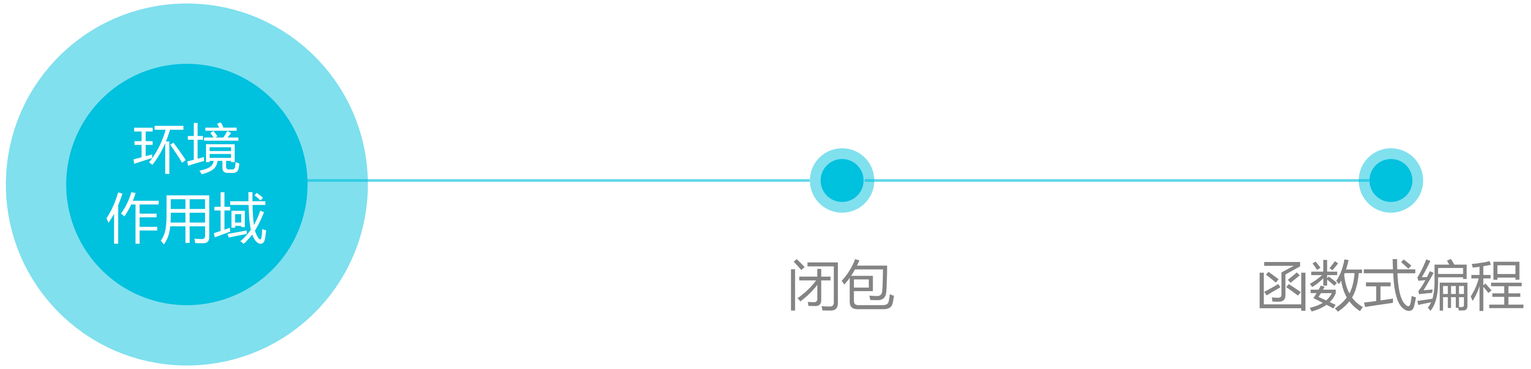
JavaScript

函数

逸翾 2018.04.12

函数

环境&作用域



函数

环境&作用域

1、执行环境是JavaScript中最重要的一个概念，执行环境定义了变量或者函数有权访问的其他数据。
全局环境是最外围的一个执行环境，在Web浏览器中，全局执行环境被认为是window对象。
2、每个函数都有自己的执行环境。
3、当代码在一个环境中执行时，会创建变量对象的一个作用域链。

函数

环境&作用域

作用域最大的用处就是隔离变量，不同作用域下同名变量不会有冲突。
作用域中变量的值是在执行过程中产生的确定的，而作用域却是在函数创建时就确定了。

函数

块级作用域-var

使用var声明的变量会自动添加最接近的环境中。
不论var声明的变量处于当前作用域的第几行，都会提升到作用域的头部，并被初始化为undefined。

```
console.log(i);
for (var i = 0; i < 10; i++) {

}
console.log(i)
```

```
undefined
```

```
10
```

函数

块级作用域-let/const

let不允许在相同作用域内，重复声明同一个变量
只要块级作用域内存在let命令，它所声明的变量就“绑定”这个区域，不再受外部的影响，不存在变量提升

Var 的作用域是函数作用域
Let 的作用域是 {}

函数

环境&作用域

```
var x = 10;
function fn() {
  console.log(x);
}
function show(f) {
  var x = 20;
  (function () {
    f()
  })()
}
show(fn);
```

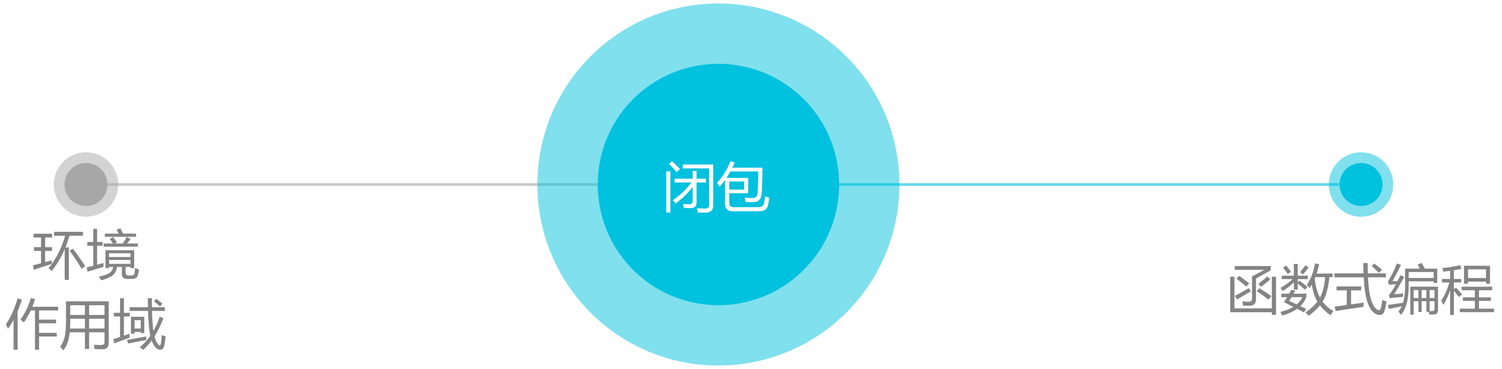
```
10
```

在作用域中使用的变量，却没有在当前作用域中声明，就需要到另一个作用域中取。
要到『创建』这个函数的那个作用域中取值，而不是『调用』这个函数的作用域中取值。
如果在上一个作用域中也没找到就要继续向上跨，直到跨到全局作用域为止。

作用域链

函数

闭包



函数

闭包

函数内部可以直接读取全局变量
函数外部自然无法读取函数内的局部变量

如何从外部读取局部变量?

在函数的内部，再定义一个函数!

```
function f1() {
  var n = 999;
  function f2() {
    console.log(n);
  }
  return f2;
}
var result = f1();
result();
```

```
999
```

函数

闭包

优势：

闭包就是将函数内部和函数外部连接起来的一座桥梁，
闭包使得函数能够读取其他函数内部变量的函数。

隐患：

由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包。
闭包会在父函数外部，改变父函数内部变量的值。

函数

函数式编程

函数

函数式编程

函数式编程（functional programming）或称函数程序设计，又称泛函编程，是一种编程典范，它将电脑运算视为数学上的函数计算，并且避免使用程序状态以及易变对象。函数式编程强调程序执行的结果而非执行的过程，倡导利用若干简单的执行单元让计算结果不断渐进，逐层推导复杂的运算，而不是设计一个复杂的执行过程。

——维基百科

函数

函数式编程

函数是第一等公民

1、语义更加清晰
2、可复用性更高
3、代码更少，可维护性更好
4、作用域局限，副作用少

纯函数
函数柯里化
高阶函数

函数

函数式编程

纯函数。
对于相同的输入，永远会得到相同的输出，而且没有任何可观察的副作用，也不依赖外部环境的状态。

```
//不纯的
var min = 18;
var checkage = function (age) {
  return age > min;
}
//纯函数
var checkage = age => age > 18;
```

函数

函数式编程

函数柯里化

给函数分步传递参数，每次传递参数后,部分应用参数，并返回一个更具体的函数接受剩下的参数，中间可嵌套多层这样的接受部分参数函数，逐步缩小函数的适用范围，逐步求解，直至返回最后结果。

```
function add(x, y) {
  return x + y;
}
add(2, 4)
```

```
function add(x) {
  return function (y) {
    return x + y;
  }
}
let a = add(2)
a(4);//6
```

```
let add = x => y => x + y;
var add2 = add(2);
add2(4);// 6
```

函数

函数式编程

高阶函数

指函数可以作为参数被传递或者作为返回值输出。

map、reduce、filter

```
let a = [1, 2, 3, 4, 5]
a.map(function (x) { return x * 3 })

// 箭头函数
let a = [1, 2, 3, 4, 5]
a.map(x => x * 3)
```

为了无法计算的价值 | 阿里云