

变量对象

2020年03月09日, 星期一 09:17

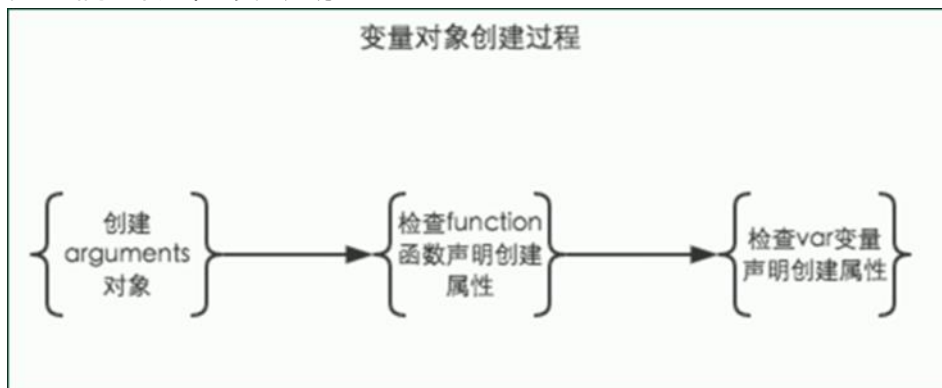
1.

变量对象 (Variable Object)

- 变量对象是与执行上下文相关的数据作用域(scope of data)。
- 变量对象是与上下文关联的特殊对象，用于存储被定义在上下文中的变量(variables) 和函数声明(function declarations)

变量对象创建过程

- 建立 arguments 对象
- 检查当前上下文的函数声明
- 检查当前上下文中的变量声明



2.变量对象

```
function fun(a, b) {  
  var c = 4;  
  function fn() {}  
}  
fun(1, 2, 3)
```

```
funEC.VO={  
  arguments:  
  a:  
  b:  
  c:  
  fn:  
}
```

3.变量对象

```
function foo() {  
  var a = 20;  
  
  function bar() {  
    a = 30;  
    console.log(a);  
  }  
  bar();  
}  
foo();
```

函数 foo 和函数 bar 的变量对象?

```
fooEC.VO = {  
  arguments,  
  a,  
  bar  
};
```

```
barEC.VO = {  
  arguments  
};
```

函数 bar 的变量对象是否包含了变量 a?

否

foo 变量对象中的 a 的值是多少，如何确定?

30

4.代码运行机制

```
console.log(a); //undefined
var a = 2;
console.log(a); //2
```

//从解析器角度看到的代码

```
var a;
console.log(a);
a = 2;
console.log(a);
```

//先创建变量对象，a一开始就存在了，因此第一次的console.log(a)不会报错

5.JavaScript代码运行机制

JavaScript 代码的执行分为两个阶段

代码编译阶段：将代码翻译成可执行代码

代码执行阶段：执行可执行代码

JavaScript 编译和执行过程

全局编译阶段（预解析）

全局顺序执行阶段（变量赋值、函数调用等操作）

当遇到函数调用时，在执行函数内代码前，进行函数范围内的编译

当存在函数嵌套时，以此类推，会进行多次函数预解析

注：**编译和执行**是一个不断交替的过程

6.声明提升

预解析工作之一 —— 声明提升

所有的变量声明和函数声明提升到当前作用域的最前面

声明提升规则

规则1：函数声明整体提前

规则2：变量声明提前，赋值留在原地

规则3：函数会首先被提升，然后才是变量

规则4：函数声明有冲突，会覆盖；变量声明有冲突，会忽略

7.声明提升

```
foo();
function foo() {
  console.log(1);
}
var foo = function() {
  console.log(2);
};
function foo() {
  console.log(3);
}
```

等价于

```
/*
  function foo() { 被覆盖
    console.log(1);
  }
*/
function foo() {
  console.log(3);
}
// var foo; 被忽略
foo();
foo = function() {
  console.log(2);
};
```

四个案例：

```
console.log(add);
var add = 1;
var add = 2;
console.log(add);
```

```
console.log(add);
function add() {
  console.log(1)
}
function add() {
  console.log(2)
}
console.log(add);
```

```
console.log(add);
var add = 1;
function add() {
  console.log(2)
}
console.log(add);
```

```
console.log(add);
function add() {
  console.log(1)
}
var add = 2;
console.log(add);
```

demo: 6.html

输出结果:

```
2
undefined
4
```

8.练习

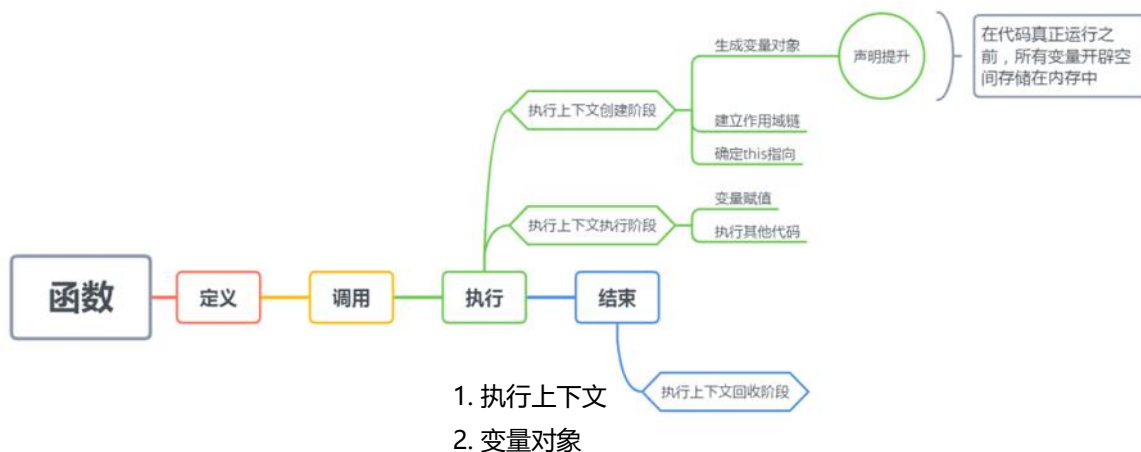
```
var a = 1;
function add(a, b) {
  console.log(a); //2
  console.log(sum); //undefined 此时已经创建变量对象,
  //因此是undefined, 而不是报错。
  return b;
  var sum = 2;
  console.log(b);
}
var c = a + add(2, 3);
console.log(c); //4
```

```
windowEC.VO = window;
```

9.全局上下文的变量对象

- 浏览器中的全局上下文的变量对象
 - 变量对象就是 **window** 对象
 - 在页面关闭前一直存在

10.



11.分析代码 (课下作业)

```
var a = 10,  
    b = 20;  
  
function fn() {  
    var a = 100,  
        c = 200;  
  
    function bar() {  
        var a = 500,  
            d = 600;  
    }  
    bar();  
}  
  
fn();
```

1. 执行上下文
2. 变量对象

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