JavaScript 作用域 09

JS

标识符

> 所谓标识符，就是指变量名、函数名、对象属性名、函数参数名。

标志识别的符号。

作用域的概念

> 作用域就是用来管理标识符可以在哪里被访问的一套规则。

- 全局作用域：在全局内声明的函数之外的部分。
- 函数作用域：函数内部的范围。

浏览器的JS引擎在运行JS代码之前，会经过2个步骤（无论是全局作用域还是函数作用域）：

1. 将所有的标识符提升
2. 逐行执行代码

例如下面这个例子：

```
// 可以想象成 var a; 先被放到这里了。

alert(a);  // 只是提升标识符，而不会赋值。 所以这里弹出 undefined

var a = 1;
```

作用域链

> 作用域链规定了，当作用域嵌套时候标识符的访问规则。它的规则是，函数作用域优先使用自身作用域的标识符，如果没有找到就去上一级去找，直到找到全局作用域，如果全局作用域还没有，那么就会抛出一个引用错误。同时还规定了，外面的作用域无法访问函数内的作用域。

```
<script>
    // 全局作用域

    /*var a = 10;

    function fn(){
      // 函数作用域
      alert(a);
    }

    fn();*/

    /*function fn1(){
      var a = 1;
    }

    alert(a);*/

    /*var a = 1;

    function fn(){
      // var b;
      // function fn1(){....}
      var b = 2;
      function fn1(){
        // var c;
        var c = 3;
        alert(b);  // 2
        alert(a);  // 1
      }
      fn1();
    }*/

    //fn();

    /*alert(b)
    alert(c);*/

```

```
39.       // var fn3;
40.       //
41.       /*alert(abc);
42.
43.       fn3();
44.
45.       var fn3 = function (){
46.         alert(1);
47.       };*/
48.
49.       function fn1(){
50.         var a = 1;
51.         fn2();
52.       }
53.
54.       function fn2(){
55.         alert(a);
56.       }
57.
58.       fn1();
59.     </script>
```

浏览器报错说明

- ReferenceError ： 引用错误，通常指标识符没有找到。
- TypeError： 类型错误，通常指对指定了的类型执行了错误的操作，例如一个不是函数的标识，执行了“()”执行操作符。
- Tokn： 词法错误，通常是因为写错了 `;` 和 `(`。

本文在线地址