【前端面试】JS严格模式有什么特点?

前言

我们常说 JavaScript 是一门弱类型语言，相对于 Java 这类的强类型语言来说，JavaScript 这种脚本语言的语法规则就松散的多。松散的模式对我们来说有好处也有坏处，好处便是我们写代码的时候不用考虑那么多，普通小白随便学学之后也能快速上手。但是坏处也是显而易见的，这种松散的代码会导致很多不可预测的错误，代码规范也是很大的一个问题。

所以为了让我们编写的代码变得严谨一点，ECAMscript5 提出了”严格模式”，处于严格模式下运行的 JavaScript 代码会遵循更加苛刻的条件。

1.如何开启严格模式?

我们平常写代码的时候可能没太注意严格模式和非严格模式，默认情况下，我们编写的 js 代码都是非严格模式的，想要开启严格模式，我们需要在规定的地方添加’use strict’关键词。

1.1 整个文件开启严格模式

我们需要在脚本文件的第一行加上”use strict”关键词，这样整个脚本文件都会按照严格模式运行，需要注意的是如果不在第一行，那么则无效。

示例代码:

```
<script>
  "use strict";
  console.log("严格模式");
</script>
<script>
  console.log("非严格模式")
</script>
```

1.2 单个函数开启严格模式

除了给整个脚本文件添加严格模式外，我们还可以只针对某个函数开启严格模式，我们只需要将”use strict”关键词放在函数体的第一行即可。

示例代码:

```
function strict() {
  "use strict";
```

```
  return "严格模式";
}

function notStrict() {
  return "正常模式";
}
```

1.3 脚本文件合并问题

通常来说我们的一个项目都有很多个脚本文件组成，在项目打包阶段，我们可能需要将多个脚本文件合并。如果有些脚本文件开启了严格模式，有些脚本文件是正常模式，那么它们合并后的代码属于严格模式还是非严格模式呢?

如果我们不能很好的解决这个问题，那么很容易导致致命错误的出现，因此这里给出两种建议：

建议一：

如果你不能确保是否全局需要开启严格模式或者你是新手的话，建议按照一个个的函数去开启严格模式。

建议二：

如果需要脚本文件全局开启严格模式，建议将所有代码放在一个立即执行的匿名函数中去，示例代码如下：

```
(function (){
  "use strict";
})();
```

2.严格模式的特点

2.1 变量

2.1.1 声明变量必须要有关键词

我们都知道 js 中声明变量是非常简单的，直接使用 var、let 关键词声明即可，甚至你还可以关键词都不需要，如果在非严格模式下，这种变量默认为全局变量，但是在严格模式下则不这样允许。

非严格模式：

```
<script>
  a = '小猪课堂'
  console.log(a); // 小猪课堂
</script>
```

严格模式：

```
<script>
```

```
  "use strict";
  a = '小猪课堂'
  console.log(a); // Uncaught ReferenceError: a is not defined
</script>
```

可以看到非严格模式下不适用关键词声明变量是会报错的。

2.1.2 无法使用 delete 删除变量

我们平常开发中可以使用 delete 关键词删除某个变量或者某个对象属性，但是在严格模式下不能这么操作。

非严格模式：

```
<script>
  a = '小猪课堂'
  console.log(delete a); // true
</script>
```

严格模式：

```
<script>
  "use strict";
  a = '小猪课堂'
  console.log(delete a); // Uncaught SyntaxError: Delete of an unqualified
</script>
```

2.1.3 无法使用关键词作为变量名

在非严格模式下，我们对变量名的取决比较松散，但是在严格模式下，不能使用一些关键词作为变量名，这其实也是为了给以后的 JavaScript 发展坐下铺垫。比如我们常见的一些关键词：implements、interface、let、package 等等。

非严格模式：

```
<script>
  let implements = '小猪课堂'
  console.log(implements); // 小猪课堂
</script>
```

非严格模式：

```
<script>
  "use strict";
  let implements = '小猪课堂'
  console.log(implements); // Uncaught SyntaxError: Unexpected strict mode
</script>
```

2.2 对象

2.2.1 对象属性操作将会更严格

我们有时候会对一些对象属性设置一些限制，比如说只读、禁止添加属性等，这个时候我们再去修改对象的该属性时，非严格模式不会报错，但是严格模式会报错。

非严格模式：

```
<script>
  let obj = {}
  Object.defineProperty(obj, "name", { value: '小猪课堂', writable: false });
  obj.name = "张三"
  console.log(obj); // {name: '小猪课堂'}
</script>
```

严格模式：

```
<script>
  "use strict";
  let obj = {}
  Object.defineProperty(obj, "name", { value: '小猪课堂', writable: false });
  obj.name = "张三"
  console.log(obj); // Uncaught TypeError: Cannot assign to read only proper
</script>
```

非严格模式：

```
<script>
  let obj = {}
  Object.preventExtensions(obj); // 禁止添加属性
  obj.name = "张三"
  console.log(obj); // {}
</script>
```

严格模式：

```
<script>
  "use strict";
  let obj = {}
  Object.preventExtensions(obj);
  obj.name = "张三"
  console.log(obj); // Uncaught TypeError: Cannot add property name, object
</script>
```

2.3 函数

2.3.1 函数不能有重名参数

非严格模式下，如果一个函数的参数有重名的不会报错，严格模式下会报错。

非严格模式：

```
<script>
  function say(name, name) {
    console.log(name, name)
  }
  say('小猪课堂'); // undefined undefined
</script>
```

严格模式：

```
<script>
  "use strict";
  function say(name, name) {
    console.log(name, name)
  }
  say('小猪课堂'); // Uncaught SyntaxError: Duplicate parameter name not allo
</script>
```

2.3.2 函数 arguments 有限制

在严格模式下，针对 arguments 对象有诸多的限制，主要有下面几点：

1.无法修改 arguments

非严格模式：

```
<script>
  function say(name) {
    arguments = '23'
    console.log(arguments)
  }
  say('小猪课堂'); // 23
</script>
```

严格模式：

```
<script>
  "use strict";
  function say(name) {
    arguments = '23'
    console.log(arguments)
  }
  say('小猪课堂'); // Uncaught SyntaxError: Unexpected eval or arguments in s
</script>
```

2.arguments 不在追踪参数变化

加入我们在函数内部修改了参数的值，非严格模式下 arguments 的值也是会变化的，但是在严格模式下，arguments 与参数值独立开来了，互不影响。

非严格模式：

```
<script>
  function say(name) {
```

```
    name = '张三'
    console.log(name);
    console.log(arguments);
  }
  say('小猪课堂'); // 张三 ,  Arguments ['张三']
</script>
```

严格模式：

```
<script>
  "use strict";
  function say(name) {
    name = '张三'
    console.log(name);
    console.log(arguments);
  }
  say('小猪课堂'); // 张三 ,  Arguments ['小猪课堂']
</script>
```

3.禁止使用 arguments.callee

callee 算是一个一个历史遗留问题了，它的作用主要是调用函数自身，通常我们也不会使用。

非严格模式：

```
<script>
  function say(name) {
    arguments.callee();
  }
  say('小猪课堂'); // 正常调用，只不过要考虑内存溢出问题
</script>
```

严格模式：

```
<script>
  "use strict";
  function say(name) {
    arguments.callee();
  }
  say('小猪课堂'); // Uncaught TypeError: 'caller', 'callee', and 'arguments'
</script>
```

2.3.3 函数必须声明在顶层

其实这和 JavaScript 中的块级作用域是比较相符的，在严格模式下，我们不能把函数声明在 if 或者 for 语句中。

非严格模式：

```
<script>
  if (true) {
    function say(name) {
      console.log(name);
```

```
    }
  }
  say('小猪课堂'); // 小猪课堂
</script>
```

严格模式：

```
<script>
  "use strict";
  if (true) {
    function say(name) {
      console.log(name);
    }
  }
  say('小猪课堂'); // Uncaught ReferenceError: say is not defined
</script>
```

2.4 禁止使用 eval()

eval 一直都是备受大家争议的，它可以直接执行一段代码，这会导致非常多的问题，所以在严格模式下直接禁用它了。

非严格模式：

```
<script>
  // "use strict";
  function say(name) {
    eval("var age = 50");
    console.log(age);
  }
  say('小猪课堂'); // 50
</script>
```

严格模式：

```
<script>
  "use strict";
  function say(name) {
    eval("var age = 50");
    console.log(age);
  }
  say('小猪课堂'); // Uncaught ReferenceError: age is not defined
</script>
```

2.5 禁止使用 with 语句

with 语句主要用来扩展作用域链，它也备受争议，因为它无法在编译时确定属性到底属于哪个对象，所以我们目前是不建议使用它，所以严格模式下也直接禁止使用它了。

非严格模式：

```
<script>
  let obj = {
    name: '小猪课堂'
  }
  with (obj) {
    name = '张三';
  }
  console.log(obj); // {name: '张三'}
</script>
```

严格模式：

```
<script>
  "use strict";
  let obj = {
    name: '小猪课堂'
  }
  with (obj) {
    name = '张三';
  }
  console.log(obj); // Uncaught SyntaxError: Strict mode code may not includ
</script>
```

2.6 this 无法指向全局

我们平常会使用 bind、apply 等方法修改 this 指向，但是当我们没有显示执行 this 指向时，this 默认会指向全局，在严格模式下这是不允许的。

非严格模式：

```
<script>
  function say() {
    console.log(this)
  }
  say.call(null); // Window {window: Window, self: Window, document: documer
</script>
```

严格模式：

```
<script>
  "use strict";
  function say() {
    console.log(this)
  }
  say.call(null); // null
</script>
```

2.7 禁止八进制写法

在正常模式下，如果某个整数的第一位是 0，那么表示该数是八进制数，会自动进行转换，但是在严格模式下无法进行有效转换。

非严格模式：

```
<script>
  var a = 0100
  console.log(a); // 64
</script>
```

严格模式：

```
<script>
  "use strict";
  var a = 0100
  console.log(a); // Uncaught SyntaxError: Octal literals are not allowed ir
</script>
```

总结

严格模式给我们带来了规范，但是同时也给我们带来了一些不便，但是为了未来的发展，我们最好还是按照严格模式来，这样不仅可以最大程度的减少我们出错的机率，也能够让我们的程序能够走得更远。