JavaScript HTML DOM 节点列表 JavaScript this 关键字

JavaScript let 和 const

ECMAScript 2015(ECMAScript 6)

ES2015(ES6) 新增加了两个重要的 JavaScript 关键字: **let** 和 **const**。

let 声明的变量只在 let 命令所在的代码块内有效。

const 声明一个只读的常量，一旦声明，常量的值就不能改变。

在 ES6 之前，JavaScript 只有两种作用域： **全局变量** 与 **函数内的局部变量**。

全局变量

在函数外声明的变量作用域是全局的：

实例

```
var carName = "Volvo";
// 这里可以使用 carName 变量
function myFunction() {
// 这里也可以使用 carName 变量
}
```

尝试一下 »

全局变量在 JavaScript 程序的任何地方都可以访问。

局部变量

在函数内声明的变量作用域是局部的（函数内）：

实例

```
// 这里不能使用 carName 变量
function myFunction() {
var carName = "Volvo";
// 这里可以使用 carName 变量
}
// 这里不能使用 carName 变量
```

尝试一下 »

函数内使用 var 声明的变量只能在函数内容访问，如果不使用 var 则是全局变量。

JavaScript 块级作用域(Block Scope)

使用 var 关键字声明的变量不具备块级作用域的特性，它在 {} 外依然能被访问到。

```
{
    var x = 2;
}
// 这里可以使用 x 变量
```

在 ES6 之前，是没有块级作用域的概念的。

ES6 可以使用 let 关键字来实现块级作用域。

let 声明的变量只在 let 命令所在的代码块 {} 内有效，在 {} 之外不能访问。

```
{
    let x = 2;
}
// 这里不能使用 x 变量
```

重新定义变量

使用 var 关键字重新声明变量可能会带来问题。

在块中重新声明变量也会重新声明块外的变量：

实例

```
var x = 10;
// 这里输出 x 为 10
{
var x = 2;
// 这里输出 x 为 2
}
// 这里输出 x 为 2
```

尝试一下 »

let 关键字就可以解决这个问题，因为它只在 let 命令所在的代码块 {} 内有效。

实例

```
var x = 10;
// 这里输出 x 为 10
{
let x = 2;
// 这里输出 x 为 2
}
// 这里输出 x 为 10
```

尝试一下 »

浏览器支持

Internet Explorer 11 及更早版本的浏览器不支持 let 关键字。

下表列出了各个浏览器支持 let 关键字的最低版本号。

				
Chrome 49	IE / Edge 12	Firefox 44	Safari 11	Opera 36
Mar, 2016	Jul, 2015	Jan, 2015	Sep, 2017	Mar, 2016

循环作用域

使用 var 关键字：

实例

```
var i = 5;
for (var i = 0; i < 10; i++) {
// 一些代码...
}
// 这里输出 i 为 10
```

尝试一下 »

使用 let 关键字：

实例

```
let i = 5;
for (let i = 0; i < 10; i++) {
// 一些代码...
}
// 这里输出 i 为 5
```

尝试一下 »

在第一个实例中，使用了 **var** 关键字，它声明的变量是全局的，包括循环体内与循环体外。

在第二个实例中，使用 **let** 关键字， 它声明的变量作用域只在循环体内，循环体外的变量不受影响。

局部变量

在函数体内使用 **var** 和 **let** 关键字声明的变量有点类似。

它们的作用域都是 **局部的**:

```
// 使用 var
function myFunction() {
    var carName = "Volvo";   // 局部作用域
}

// 使用 let
function myFunction() {
```

```
    let carName = "Volvo";   //  局部作用域
}
```

全局变量

在函数体外或代码块外使用 **var** 和 **let** 关键字声明的变量也有点类似。

它们的作用域都是 **全局的**:

```
// 使用 var
var x = 2;       // 全局作用域

// 使用 let
let x = 2;       // 全局作用域
```

HTML 代码中使用全局变量

在 JavaScript 中, 全局作用域是针对 JavaScript 环境。

在 HTML 中, 全局作用域是针对 window 对象。

使用 **var** 关键字声明的全局作用域变量属于 window 对象:

实例

```
var carName = "Volvo";
// 可以使用 window.carName 访问变量
```

尝试一下 »

使用 **let** 关键字声明的全局作用域变量不属于 window 对象:

实例

```
let carName = "Volvo";
// 不能使用 window.carName 访问变量
```

尝试一下 »

重置变量

使用 **var** 关键字声明的变量在任何地方都可以修改：

实例

```
var x = 2;
// x 为 2
var x = 3;
// 现在 x 为 3
```

尝试一下 »

在相同的作用域或块级作用域中，不能使用 **let** 关键字来重置 **var** 关键字声明的变量:

```
var x = 2;       // 合法
let x = 3;       // 不合法

{
    var x = 4;   // 合法
    let x = 5   // 不合法
}
```

在相同的作用域或块级作用域中，不能使用 **let** 关键字来重置 **let** 关键字声明的变量:

```
let x = 2;       // 合法
let x = 3;       // 不合法

{
    let x = 4;   // 合法
    let x = 5;   // 不合法
}
```

在相同的作用域或块级作用域中，不能使用 **var** 关键字来重置 **let** 关键字声明的变量:

```
let x = 2;       // 合法
var x = 3;       // 不合法

{
    let x = 4;   // 合法
    var x = 5;   // 不合法
}
```

let 关键字在不同作用域，或不同块级作用域中是可以重新声明赋值的:

```
let x = 2;       // 合法

{
    let x = 3;   // 合法
}

{
    let x = 4;   // 合法
}
```

变量提升

JavaScript 中，var 关键字定义的变量可以在使用后声明，也就是变量可以先使用再声明（JavaScript 变量提升）。

实例

```
// 在这里可以使用 carName 变量
var carName;
```

尝试一下 »

let 关键字定义的变量则不可以在使用后声明，也就是变量需要先声明再使用。

```
// 在这里不可以使用 carName 变量

let carName;
```

const 关键字

const 用于声明一个或多个常量，声明时必须进行初始化，且初始化后值不可再修改：

实例

```
const PI = 3.141592653589793;
PI = 3.14; // 报错
PI = PI + 10; // 报错
```

尝试一下 »

const定义常量与使用let 定义的变量相似：

- 二者都是块级作用域
- 都不能和它所在作用域内的其他变量或函数拥有相同的名称

两者还有以下两点区别：

- const声明的常量必须初始化，而let声明的变量不用
- const 定义常量的值不能通过再赋值修改，也不能再次声明。而 let 定义的变量值可以修改。

```
var x = 10;
// 这里输出 x 为 10
{
    const x = 2;
    // 这里输出 x 为 2
}
// 这里输出 x 为 10
```

const 声明的常量必须初始化：

```
// 错误写法
const PI;
PI = 3.14159265359;

// 正确写法
const PI = 3.14159265359;
```

并非真正的常量

const 的本质: const 定义的变量并非常量，并非不可变，它定义了一个常量引用一个值。使用 const 定义的对象或者数组，其实是可变的。下面的代码并不会报错：

实例

```
// 创建常量对象
const car = {type:"Fiat", model:"500", color:"white"};
// 修改属性:
car.color = "red";
// 添加属性
car.owner = "Johnson";
```

尝试一下 »

但是我们不能对常量对象重新赋值：

实例

```
const car = {type:"Fiat", model:"500", color:"white"};
car = {type:"Volvo", model:"EX60", color:"red"}; // 错误
```

尝试一下 »

以下实例修改常量数组：

实例

```
// 创建常量数组
const cars = ["Saab", "Volvo", "BMW"];
// 修改元素
cars[0] = "Toyota";
// 添加元素
cars.push("Audi");
```

尝试一下 »

但是我们不能对常量数组重新赋值：

实例

```
const cars = ["Saab", "Volvo", "BMW"];
cars = ["Toyota", "Volvo", "Audi"]; // 错误
```

尝试一下 »

浏览器支持

Internet Explorer 10 及更早版本的浏览器不支持 const 关键字。

下表列出了各个浏览器支持 const 关键字的最低版本号。

				
Chrome 49	IE / Edge 11	Firefox 36	Safari 10	Opera 36
Mar, 2016	Oct, 2013	Feb, 2015	Sep, 2016	Mar, 2016

重置变量

使用 **var** 关键字声明的变量在任何地方都可以修改：

实例

```
var x = 2; // 合法
var x = 3; // 合法
x = 4; // 合法
```

在相同的作用域或块级作用域中，不能使用 **const** 关键字来重置 **var** 和 **let**关键字声明的变量:

```
var x = 2;          // 合法
const x = 2;        // 不合法
{
    let x = 2;      // 合法
    const x = 2;    // 不合法
}
```

在相同的作用域或块级作用域中，不能使用 **const** 关键字来重置 **const** 关键字声明的变量:

```
const x = 2;        // 合法
const x = 3;        // 不合法
x = 3;              // 不合法
var x = 3;          // 不合法
let x = 3;          // 不合法

{
    const x = 2;    // 合法
    const x = 3;    // 不合法
    x = 3;          // 不合法
    var x = 3;      // 不合法
    let x = 3;      // 不合法
}
```

const 关键字在不同作用域，或不同块级作用域中是可以重新声明赋值的:

```
const x = 2;       // 合法

{
    const x = 3;   // 合法
}

{
    const x = 4;   // 合法
}
```

变量提升

JavaScript var 关键字定义的变量可以在使用后声明，也就是变量可以先使用再声明（JavaScript 变量提升）。

实例

```
carName = "Volvo"; // 这里可以使用 carName 变量
var carName;
```

尝试一下 »

const 关键字定义的变量则不可以在使用后声明，也就是变量需要先声明再使用。

```
carName = "Volvo";    // 在这里不可以使用 carName 变量
const carName = "Volvo";
```

← JavaScript HTML DOM 节点列表 JavaScript this 关键字 →