变量定义

为什么要有新的变量定义方式

现在一般定义变量都用 `var`，但是 `var` 会有很多的问题：

- `var` 没有块级作用域，定义后在当前闭包中都可以访问，如果变量名重复，就会覆盖前面定义的变量，并且也有可能被其他人更改。
- `var` 在 `for` 循环标记变量共享，一般在循环中使用的 `i` 会被共享，其本质上也是由于没有块级作用域造成的，由于情况比较多见和特殊，单独算一个问题。

因此有了两种新的变量定义方式 `let` 和 `const` 。

用 `let` 定义的变量具有块级作用域。 用 `const` 定义的变量是静态变量，定以后不可修改。

`let` 的好处和用法

- `let` 拥有块级作用域
- `let` 生命的全局变量不会影响 `window`
- 循环时不会共享 `let` 定义的变量

友好的报错

> `'use strict';` 用了严格模式才有有友好的报错

- 重新定义 `let` 的变量会报错
- 使用未定义的变量会报错（包括不在自己作用域中的变量）
- `let` 没有变量提升(变量预定义)，但是在一个块级作用域中，不可以先使用未定义的变量。

块级作用域是什么

在用 `var` 定义变量的时候，变量是通过闭包进行隔离的，现在用了 `let` ，不仅仅可以通过闭包隔离，还增加了一些块级作用域隔离。

> `{}`

在 `for` 、 `if` 、 `switch` 、 `function` 等的大括号都会当成一个独立的块级作用域。

闭包的新写法：

```
{

}
```

不必再是：

```
;(function () {

})();
```

const的好处和用法

`const` 是静态变量，定以后不允许再被修改或者重新定义。

- 在严格模式下，重新定义或者修改静态变量会报错。
- 不同的块级作用域下，可以定义相同名字的静态变量。