

# JavaScript进阶 (ES6)

---ES6背景知识

## 背景知识 (ES5、ES6)

---

- ECMAScript (ES) 是JavaScript的语法标准
- ECMAScript的版本 (ES5、ES6)
  - ES5 (2009年12月发布)
  - ES6 (2015年06月发布) 增加了许多新特性, 并解决了很多ES5中的缺陷
- 如何查看当前环境对ES6的支持情况
  - <https://ruanyf.github.io/es-checker/index.cn.html>
  - <https://kangax.github.io/compat-table/es6/> 和 <http://node.green/>
- 遇到兼容问题时如何将ES6转为ES5 (<http://babeljs.io/>)

# JavaScript进阶 (ES6)

- ES6中的let与const
- ES6中变量的解构赋值



# JavaScript进阶 (ES6)

---ES6中的let与const



河北师范大学软件学院  
Software College of Hebei Normal University

# 内容提纲

---

- **ES5中的var及其缺陷**
- **ES6中的let与const**
- **let与const的重要特性**



## ES5中的var及其缺陷

- ES5通过var声明变量（无块作用域，可能造成变量污染）

```
var userId = 12;  
document.onclick = function () {  
    console.log("userId = ",userId);  
};
```

//一长串代码后，忘记了上边定义的userId

```
var a=2,b=3;  
if(a<b){  
    var userId = 34;//重复定义，变量污染  
}
```

- ES5中变量生命周期解决方案IIFE

# ES5中的var及其缺陷

- ES5通过var声明变量（可能造成变量的非期望共享）

```
> for (var i = 0; i < 3; i++) {  
    (function(j) { // j = i  
        setTimeout(function() {  
            console.log(new Date, j);  
        }, 1000*i);  
    })(i);  
}
```

< undefined

Fri Sep 01 2017 15:36:02 GMT+0800 (中国标准时间) 0

Fri Sep 01 2017 15:36:03 GMT+0800 (中国标准时间) 1

Fri Sep 01 2017 15:36:04 GMT+0800 (中国标准时间) 2

# 内容提纲

---

- ES5中的var及其缺陷
- ES6中的let与const
- let与const的重要特性





# ES6中的let与const

## •通过let声明变量

- ES6新增了let命令，用于声明变量，用法与var类似
- 其与var的不同在于，用let声明的变量只在 let 命令所在的代码块 { }内有效

```
let userId = 123;  
document.onclick = function () {  
    console.log("userId = ",userId);  
};
```

```
let a=2,b=3;  
if(a<b){  
    let userId = 234;//不会造成变量污染  
}
```

# ES6中的let与const

- 使用let声明变量，可有效避免变量共享缺陷

```
for (let i = 0; i < 3; i++) {  
    setTimeout(function() {  
        console.log(new Date, i);  
    }, 1000*i);  
}  
3
```

Tue Sep 05 2017 16:29:26 GMT+0800 (中国标准时间) 0

Tue Sep 05 2017 16:29:27 GMT+0800 (中国标准时间) 1

Tue Sep 05 2017 16:29:28 GMT+0800 (中国标准时间) 2

# ES6中的let与const

- ES6使用const来声明常量，也常用来声明不变的函数

```
const PI = 3.1415926;  
console.log(PI);  
PI = 3; //给常量再赋值 报错
```

```
//const声明时必须赋值  
//一旦声明必须立即初始化  
const foo; //报错  
const foo = 123; //ok
```

```
//const作用域同let  
if(true){  
    const MAX = 5;  
    console.log(MAX); //5  
}  
console.log(MAX); //报错
```

```
//const也常用来声明不变的函数  
const fee = function () {};
```

- const指向的对象引用不可变，但其属性是可变的

# 内容提纲

---

- ES5中的var及其缺陷
- ES6中的let与const
- **let与const的重要特性**



## ES6 let变量和const常量的特性--不进行变量提升特性

- var的 '声明提升' 的特性（声明前可以使用该变量）

```
console.log(a);  
var a = 1;  
console.log(a);
```

等  
价  
于  
=>

```
var a;  
console.log(a); // undefined  
a = 1;  
console.log(a); // 1
```

- 用let声明变量时，在声明前不能使用该变量（let不提升）

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

► Uncaught ReferenceError: a is not defined

## ES6 函数块作用域内提升特性

- ES5、ES6中函数提升的不同（注意：ES6中的块）

```
function f() {  
    console.log("outside")  
};  
  
{  
    f();  
    {  
        function f() {console.log("inside");}  
    }  
}
```

# ES6 let变量和const常量的特性--暂时性死区特性

## •暂时性死区 (temporal dead zone) 特性

- 只要块级作用域内存在let，它所声明的变量就“绑定”在这个区域，不再受外部影响
- let对这个块从一开始就形成了封闭的作用域，凡是在声明之前使用该变量，就会报错

```
var tmp = 123;  
if(true){  
    tmp = "abc";//Reference Error  
    let tmp;  
}
```

typeof 将不再是绝对安全的操作（参见实例），养成良好习惯，在使用变量之前定义变量

► Uncaught ReferenceError: tmp is not defined

## •var可以声明重名变量，let和const则不可以



# 总结

---

- ES5中的var及其缺陷
- ES6中的let与const
- let与const的重要特性







Thank You!



河北师范大学软件学院  
Software College of Hebei Normal University

# JavaScript进阶 (ES6)

## ---ES6中变量的解构赋值



河北师范大学软件学院  
Software College of Hebei Normal University

# 内容提纲

---

- **数组、对象的解构赋值**
- **字符串、数字的解构赋值**
- **函数参数的解构赋值**
- **解构赋值的常见应用及注意事项**



# 数组、对象的解构赋值

## • 什么是解构赋值 (Destructuring)


- ES6允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构赋值
- 这种写法属于“模式匹配”，只要等号两边的模式相同，左边的变量就会被赋予对应的值

## • 数组的解构赋值

```
let [ , , third] = ["foo", "bar", "baz"];  
console.log(third); // "baz"
```

```
let [x, , y] = [1, 2, 3];  
console.log(x,y); // 1 3
```

```
let [head, ...tail] = [1, 2, 3, 4];  
console.log(head,tail); // 1 [2, 3, 4]
```

```
 let [d, e, ...f] = ['a'];  
console.log(d,e,f); // "a" undefined []
```

参见实例demo09 数组的解构赋值

# 数组、对象的解构赋值

## • 对象的解构赋值

```
var { bar2, foo2 } = { foo2: "ccc", bar2: "ddd" };  
console.log(foo2, bar2);
```

```
var { baz3 } = { foo3: "ccc", bar3: "ddd" };  
console.log(baz3);
```

```
var { foo4: baz4 } = { foo4: 'aaa', bar4: 'bbb' };  
console.log(baz4); // "aaa"
```

ccc ddd

undefined

aaa

解构赋值时，左侧为键值对时  
要注意键值对赋值时的对应关系

如果是键值对的情况，键只用于  
匹配，真正赋给的是对应的值

# 内容提纲

---

- 数组、对象的解构赋值
- 字符串、数字的解构赋值
- 函数参数的解构赋值
- 解构赋值的常见应用及注意事项



# 字符串、数字的解构赋值

## • 字符串的解构赋值

```
const [a, b, c, d, e] = 'hello';  
console.log(a); // "h"  
console.log(b); // "e"  
console.log(c); // "l"  
console.log(d); // "l"  
console.log(e); // "o"
```

## • 数字的解构赋值

```
let {toString: s1} = 123;  
console.log(s1); //  
s1 === Number.prototype.toString // true  
  
function toString() { [native code] }  
  
true
```

解构赋值的规则是，只要等号右边的值不是对象，就先将其转为对象

思考：左侧123改为true，结果如何？

# 内容提纲

---

- 数组、对象的解构赋值
- 字符串、数字的解构赋值
- 函数参数的解构赋值
- 解构赋值的常见应用及注意事项





# 函数参数的解构赋值

## • 参数的解构赋值 案例一

```
function move1({x = 0, y = 0} = {}) {  
    return [x, y];  
}  
console.log(move1({x: 3, y: 4})); // [3, 4]  
console.log(move1({x: 3})); // [3, 0]  
console.log(move1({})); // [0, 0]  
console.log(move1()); // [0, 0]
```

## • 参数的解构赋值 案例二

```
function move2({x, y} = { x: 0, y: 0 }) {  
    return [x, y];  
}  
console.log(move2({x: 3, y: 8})); // [3, 8]  
console.log(move2({x: 3})); // [3, undefined]  
console.log(move2({})); // [undefined, undefined]  
console.log(move2()); // [0, 0]
```

参见实例demo12 函数参数解构赋值



# 内容提纲

---

- 数组、对象的解构赋值
- 字符串、数字的解构赋值
- 函数参数的解构赋值
- 解构赋值的常见应用及注意事项



# 解构赋值的常见应用 一

- 交换变量（写法简洁，语义清晰）

```
var [x,y] = ["a","b"];  
[x, y] = [y, x];  
console.log(x,y); //b,a
```

- 从函数中返回多个值

- 函数只能返回一个值，如果要返回多个值，只能将它们放在数组或对象里返回
- 有了解构赋值，取出这些值就非常方便

```
function example() {  
    return [1, 2, 3];  
}  
var [a, b, c] = example();  
console.log(a,b,c); //1,2,3
```

## 解构赋值的常见应用 二

- 提取JSON数据

```
var jsonData = {id: 42, status: "OK", data: [867, 5309]};  
let { id, status, data: number } = jsonData;  
console.log(id, status, number); // 42, "OK", [867, 5309]
```

- 给函数指定默认参数

```
jQuery.ajax = function (url, {  
    async = true,  
    beforeSend = function () {},  
    cache = true,  
    // ... more config  
}) {  
    // ... do stuff  
};
```

## 解构赋值的注意事项（括号问题）

- 对于编译器来说，一个式子到底是模式，还是表达式，没有办法从一开始就知道，必须解析到（或解析不到）等号才能知道
- 由此带来的问题是，如果模式中出现圆括号怎么处理，ES6的规则是，只要有可能导致解构的歧义，就不得使用圆括号
- 建议只要有可能，就不要在模式中放置圆括号

The background of the slide is decorated with various abstract shapes in shades of green and yellow. These shapes, which include circles, ovals, and irregular blobs, are scattered across the top and right sides of the slide, creating a modern, organic feel.

# Thank You!



河北师范大学软件学院  
Software College of Hebei Normal University

## 作业

---

- 复习本章的课件和练习
- 阅读ES6 标准入门的1-3章

