ES6 学习笔记

作者：wenyl

原文链接：https://ld246.com/article/1683526606401

来源网站：链滴

概念

ECMAScript是浏览器脚本语言的规范，Java script是规范的具体实现。

参考：1.1 ES6 教程 | 菜鸟教程 (runoob.com)

变量

let和var

let 是在代码块内有效，var 是在全局范围内有效，下述代码打印时，会提示 a is not defined

```
<script>
   {
      let a = 1;
      var b = 2;
   }
   console.log(a)
   console.log(b)
</script>
```

let声明的变量不能重复定义，var可以，下属代码运行提示 Identifier 'a' has already been declared

```
<script>
   {
      let a = 1;
      let a = 1;
      var b = 2;
      var b = 3;
   }
```

```
    console.log(a)
    console.log(b)
</script>
```

var 存在变量提升，let不存在变量提升，下述代码，b打印结果为undefined，a打印提示 caught ReferenceError: Cannot access 'a' before initialization

```
<script>
    console.log(b)
    var b = 2;
    console.log(a)
    let a = 1;
</script>
```

const

const声明的变量不能修改，const声明的变量必须初始化，下属代码修改const值，控制台报错提示 pError: Assignment to constant variable.

结构表达式

解构赋值是对赋值运算符的扩展。

他是一种针对数组或者对象进行模式匹配，然后对其中的变量进行赋值。

```
<script>
    let arr = [1,2,3]
    let [a,b,c] = arr;
    console.log(a,b,c)
</script>
```

函数参数

参数默认值

下述代码中，a和b默认都为0，如果不给函数传递参数，拿就会按默认值运算

```
function addNum(a=0,b=0){
    console.log(a+b)
}
```

参数个数不固定

```
    function paramsFunc(...params){
        console.log(params)
    }
    paramsFunc(1,2)
    paramsFunc(1,2,3)
```

下属代码输出结果

```
▶ (2) [1, 2]
▶ (3) [1, 2, 3]
```

箭头函数

箭头函数是一种简洁得函数写法

单个参数

```
var f = v => v;
//等价于
var f = function(a){
    return a;
}
```

多个参数

```
var f = (a,b) => a+b;
```

方法有多行

方法有多行得话，将箭头后的部分用大括号括起来，然后再里面写方法体

```
var f = (a,b) =>{
    let c = a+b;
    return  c;
};
```

方法返回值

方法不加大括号就默认返回箭头后得表达式运算值，如果方法不需要返回值则需要在大括号中声明

```
var f = (a,b) =>{
    let c = a+b;
    console.log(c)
};
```

返回一个对象

需要返回对象时，可以用大括号将方法体包起来，然后再里面定义定向返回，如果不适用方法体，可用()封装对象

```
var f = (a,b) => ({'a':a,'b':b})
    console.log(f(1,2))
```

返回结果

```
▶ {a: 1, b: 2}
```

使用时特殊情况总结

- 没有 this、super、arguments 和 new.target 绑定；
- 箭头函数体中的 this 对象，是定义函数时的对象，而不是使用函数时的对象；

- 不可以作为构造函数，也就是不能使用 new 命令，否则会报错

```
// 此时的this指向定义Pershon中sayHello函数的对象，此时控制台输出undefined
var Person = {
    'age':18,
    'sayHello':()=>{
        console.log(this.age)
    }
}
```

若对象中定义方法，需要访问this使用以下方法

```
var Person = {
    'age':18,
    'sayHello':function (){
        setTimeout(()=>{
            console.log(this.age)
        })
    }
}
```

或者

```
var Person = {
    'age':18,
    'sayHello':function (){
      console.log(this.age)
    }
}
```