ES6

- let\const关键字
- =>函数
- 解构赋值
- 模板字符串
- promise（AJAX中学习）

ES6_let/const

□ let

■Let用来声明变量

ES6新增了let命令，用来声明变量。它的用法类似于var，但是所声明的变量，只在let命令所在的代码块内有效。

如：

```
{
        var num1 = 20;
        let num2 = 30;//num2的作用域范围是所在的花括号内
}
alert("num1="+num1);
alert("num2="+num2);//num2不能使用，因为出了num2 的作用域范围
```

■ Let声明的变量不会提前（前置）

■ 块级作用域

ES5只有全局作用域和函数作用域，没有块级作用域。

*var*申明的变量在代码块外面能被识别，但是let命令却不能被识别，这样就实现了js的块级作用域，我们在使用条件语句 循环语句等就会不担心变量污染的问题了。一般来说，一对花括号属于一个块级作用域。

■以后能用let不用var。

ES6_let/const

- const

Const是只读变量（也叫常量），const修饰的是直接指向(修饰)的内存。

```
const num =20;
alert(“num=”+num);
//num = 100;//此句话执行失败
//alert(“num=”+num);

const arr = [12,34,45,56];
alert(“arr=”+arr);

arr[0]=90;//此句话可以执行
alert(“arr=”+arr);

arr = [100,200];//此句话不可以执行
alert("arr="+arr);
```

ES6 =>函数

- Arrow Function（箭头函数）。这是ES6标准新增的一种的函数
  箭头函数就是定义时使用的是箭头。
  - 定义一（无函数名）：
    参数=>函数体
    如：
    x => x*5；
    等价于：
    function (x) { return x*5; }
    箭头函数相当于匿名函数，省略了return和花括号
  - 调用：
    let t = (x=>5*x)(8); //这个有点像自运行函数
    console.log(t);//输出40。

ES6 =>函数

- 定义二（有函数名）：

  let 函数名 = 参数=>函数体

  如：

  ```
  let f = x=>x*x；//等价于 let f = function (x) {    return x*x; }
  console.log(f(8));
  ```

ES6 =>函数

- 定义三（函数体有多条语句）：
  此时不能省略函数体的花括号和return。
  如：

```
x => {
        if (x > 0) {
            return x * x;
        }
        else {
            return -x * x;
        }
    }
```

ES6 =>函数

- 定义四（函数有多个参数）：
  把参数要用括号括起来。
  如：

```
(x, y) => x>y?x:y;

定义并自运行：
var t = ((x, y) => x>y?x:y)(12,5);
console.log(t);

//定义
var f = (x, y) => x>y?x:y;
//调用
console.log(f(12,50));
```

ES6 =>函数

- 定义五（无参）：
  把参数要用括号括起来。
  () => 24*3600
- 定义六（可变参数）：

```
var f = (a,...end) => {  // 相当于把省略号的所有值，作为数组end处理
            var sum = a;
            for(var i=0; i<end.length; i++){
                    sum += end[i];
            }
            return sum;
    }
    console.log(f(1,2,3,4,5,6));
```

ES6 =>函数

- 定义七（返回值是对象）：
  var f=(id,name)=>({id:id,name:name}); //红颜色的圆括号不能省略，因为，对象的花括号和函数体的花括号冲突了。
  console.log(f("007","张无忌").id);

ES6_解构赋值

解构数组

解构赋值可将数组的元素或对象的属性赋予给另一个变量。该变量的定义语法与数组字面量或对象字面量很相似

- 批量给变量赋值一：

如下数组：

```
let  arr=[12,23,34];
```

如果需要把每个数组元素对应一个变量，以前的做法，分别定义三个变量，一一赋值。

```
let  v1 = arr[0];
let  v2 = arr[1];
let  v3 = arr[2];
```

而使用解构赋值，就可以一次性搞定：

```
let [v1,v2,v3]=arr;
```

- 批量给变量赋值二：

```
let [v1,v2,v3]=[12,23,34];
```

ES6_解构赋值

- 解构数组
  - 批量给变量赋值三：
    左侧的变量列表可以用逗号的形式跳过右侧对应的值
    let [,,v3]=[12,23,34];　　//这相当于只定义了一个变量v3，值为34。
  - 批量给变量赋值四（数组嵌套）：
    左右两侧的格式应保持一致
    let [v1,[v2,v3],[v4,v5]]=[12,[23,34],[45,56]];//数组嵌套
  - 批量给变量赋值五：

```
//将右侧多余的值以数组的形式赋值给左侧变量的语法——“rest“模式
let [v1,...v2]=[12,23,34,45,56];//从23开始朝后的所有数字作为数组赋给v2。
console.log("v1="+v1);
console.log("v2="+v2);
```

ES6_解构赋值

解构对象

```
var p = {
        "id":"007",
        "name":"张三疯",
        "sex":"男"
};
```

- 定义变量，把属性的值进行对应赋值

```
var {id:idV,name:nameV} = p;//定义变量idV和nameV，分别用属性id和name的值进行赋值。
console.log("idV="+idV);
console.log("nameV="+nameV);
```

- 如果属性名和变量名一样时，可以简写：

```
var {id,name} = p;
console.log("id="+id);
console.log("name="+name);
```

ES6_解构赋值

解构对象

- 对象嵌套

```
var {id,name,address:{city,area}} = {id:"007",name:"李四",address:{city:"beijing",area:"昌平"}};

console.log("id="+id);
console.log("name="+name);
console.log("city="+city);
console.log("area="+area);
```

ES6_模板字符串

- **语法：**
  用`（反引号）标识，用${}将变量括起来
  实例：
  `He is <b> ${person.name} </b> and we wish to know his ${person.age} .that is all`