JavaScript语言

> 讲师：邹义良

开发环境

编辑器: Visual Studio Code 或 WebStorm

官方下载 Node.js https://nodejs.org TLS版本

安装`readline-sync`模块

```
node -v
npm install readline-sync
```

练习从控制台获取键盘输入，新建一个`example.js`文件，代码如下

```
var rs = require('readline-sync')
var username = rs.question('input your name: ')
console.log("Hi, " + username)
```

运行

```
node example.js
```

变量

JavaScript 是大小写敏感的，分号可有可无

```
//三种声明方式
var age = 18
let num = 100
const PI = 3.14
```

变量的作用域

数据类型

ECMAScript 标准定义了8种数据类型：

七种`基本`数据类型 + `对象`类型

- boolean(布尔) 有2个值分别是：true 和 false
- null 一个表明 null 值的特殊关键字

- undefined 也一个特殊的关键字，通常表示一个没有被赋值的变量
- number (数字) 整数或浮点数，例如： 1949 或者 3.14
- bigInt (任意精度的整数) 可以安全地存储和操作大整数，甚至可以超过数字的安全整数限制
- string（字符串） 字符串是一串表示文本值的字符序列，例如："hello"
- symbol (符号类型) 一种实例是唯一且不可改变的数据类型，ES6新增
- object (对象) 对象可以被看作是一组属性的集合

除 Object 以外的所有类型都是不可变的（值本身无法被改变）。例如，与 C 语言不同，JavaScript 中字符串是不可变的（译注：如，JavaScript 中对字符串的操作一定返回了一个新字符串，原始字符串并没有被改变）。我们称这些类型的值为“原始值”。

动态类型，变量的类型可以动态改变

字符串单双引号都一样

```
let name = "jack"
let msg = "hi " + name
console.log(msg)
let msg2 = `hi ${name}`
console.log(msg2)
```

用对象来表示数据

```
let user = {name: "jack", age: 18}
let user2 = user
user2.name = "mary"
console.log(user.name)  //会影响到user，因为对象是引用的
```

类型转换

```
//字符串转数字
var str = "3.14"
console.log(str + 5)          //3.145
console.log(parseInt(str))    //3
console.log(parseFloat(str)) //3.14
```

控制结构

```
let num = "3"

switch (num) {
    case 2:
        console.log("星期二")
        break
    case 3:                     //注意数据类型是否一样
        console.log("星期三")
        break
    default:
        console.log("未知")
}
```

函数

```
function add(a, b) {
    return a + b
}

function sub(a, b) {
    return a - b
}

let m = 2
let n = 4

function calc(a, b, cb) { //cb: callback
    return cb(a, b)
}

let result = calc(m, n, add)
console.log(result)
console.log(calc(m, n, sub))
```

箭头函数

```
let user = {
    name: "jack"
}
user.say = function () {
    console.log(this.name)  //jack
}
user.say()

//箭头函数中，没有自己的this
user.hi = () => {
    console.log(this.name)  //undefined
}
user.hi()
```

容器

数组是特殊的对象

```
let arr = ["jack", "mary"]
console.log(arr[0])             // 通过下标操作数组中的元素
arr.push("lily")                // 添加一个成员
console.log(arr.length)         // 数组长度为3
let temp =  arr.slice(1, 2)     // 数组截取，返回新的数组，从下标1开始，到下标2(不包含2)
temp[0] = "new value"                   //不影响原数组
console.log(arr)
```

使用 `for of` 或 `for in` 遍历数组

```
let users = [
  {name: "jack", age: 18, sex: 1},
  {name: "mary", age: 20, sex: 2},
  {name: "lily", age: 17, sex: 2}
]

for(let user of users){
    console.log(user.name)
}

for(let i in users){
    console.log(users[i].name)
}
```

```
var users = [
    {name: "jack", age: 18, sex: 1},
```

```
    {name: "mary", age: 20, sex: 2},
    {name: "lily", age: 17, sex: 2},
]

//数组过滤
var girls = users.filter((user) => {
    return user.sex === 2
})

console.log(girls)
girls[0].name = "new value"
console.log(users)              // 会影响到原数组中的值
```

面象对象

```
function User(name, age) {
    this.name = name
    this.age = age

    this.say = function () {
        console.log(`姓名:${this.name}, 年龄:${this.age}`)
    }
}

//把函数当成`类`的构造方法来使用

let user = new User("jack", 18)
console.log(user.name)
user.say()

//原型链继承

function Boy() {
}

Boy.prototype = user

let boy = new Boy();
boy.say()
user.name = "new value"
boy.say()
boy.name="boy"
boy.say()
user.name = "new value xxx"
boy.say() //不再变了
```

对象中属性或方法的缩写

```
let name = "jack"
let obj = {
    // "name": name
    // name: name
    name,
    /*
    say: function () {
        console.log(this.name)
    }*/
    say() {
        console.log("我是:" + this.name)
    }
}

console.log(obj.name) //jack
obj.say()
```

箭头函数的应用

```
let users = [
    {name: "mary", sex: 1},
    {name: "tom", sex: 1},
    {name: "lily", sex: 2},
]

let me = {
    name: "jack",
    sex: 1,
    demo() {
        //挑选出性别与我相同的人

        // let _this = this
        // return users.filter(function (user) {
        //     return user.sex === _this.sex
        // })

        return users.filter((user) => {
            return user.sex === this.sex
        })
    }
}

console.log(me.demo())
```

复制出一个新对象

```
let foo = {
    name: "商品a",
    price: 12.8
}

let bar = Object.assign({}, foo)
console.log(bar)
bar.name = "商品b"
console.log(bar)
console.log(foo)

let bar2 = {...foo, description: "物美价廉"}
console.log(bar2)
```

数字与日期

```
Number 对象
Math 对象
Date 对象
```

参考资料

- JavaScript 指南 https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide