实战：手机日历组件 第二讲

ES6 & 测试

@MEATHILL

经常出没于：

- 博客： http://blog.meathill.com
- 微博： https://weibo.com/meathill/

关于作者

课程大纲

1. ES2015（ES6）简介
2. 使用 ES6 开发 JavaScript
3. 使用 Mocha.js + Should.js 搭建测试环境

教学目标

1. 了解 ES2015

1. `let` 和 `const`
2. 变量解构
3. 箭头函数
4. `Class`
5. `Module`
6. 模板字符串

2. 了解测试

1. 为什么要写测试?
2. 什么是单元测试? 怎么写?
3. 怎么开始写测试?

3. 测试驱动开发

[实战]

ES

= ECMAScript

= 由 ECMA 国际（前身为欧洲计算机制造商协会）通过 ECMA-262标准化的脚本程序设计语言。

- JavaScript 是 ECMAScript 的一种方言
- 浏览器中的 JavaScript 增加了 DOM 和 BOM
- 目前 ES5 基本完成普及，主流浏览器部分或全部支持 ES6

- ES5 = ES6 之前的规范
- ES6 = ES2015
- ES7 = ES2016 + ES2017
- ES8 = ES2018（讨论中）
- ES9 = ES2019（征集想法中）

ES2015 / ES6

妾本出身贫寒家，未想尊威仪天下

1. 为校验表单而生
2. 只花了10天就设计出原型
3. 连名字都是山寨的……

江山代有才人出，各领风骚数百年

1. JScript
2. Flash，ActiveX
3. SilverLight，JavaFX

天下逐鹿，终归秦属

1. ES3 大体上统一了 JavaScript
2. ES4 夭折
3. HTML5 获得浏览器之争的最终胜利
4. ES5 诞生
5. ES6 诞生

JAVASCRIPT 的历史

ES2015 增加了很多优秀的新特性，可以让我们开发出更强壮，更好维护的代码。

以后的开发，都应当以ES2015 为基础。

ES2015 新特性

`let` & `const`

1. 声明变量 `let a = 1;`
2. 声明常量 `const B = 2;`
3. 块级作用域 code
4. 没有变量提升 code

只要两边模式对照一致，可以有很多种拆法：

```
let [a, ...b] = [1, 2, 3, 4];
// a = 1
// b = [2, 3, 4]

let [a, , b] = [1, 2, 3];
// a = 1
// b = 3
```

解构对象

```
let {a, b, c} = {a: 1, b: 2, c: 3};
// a = 1
// b = 2
// c = 3

// 解构对象只关注键名，不关注顺序
let {b, c, a} = {a: 1, b:2, c: 3};
// a = 1
// b = 2
// c = 3
```

常见用法：函数返回多个值

```
function sample() {
  return {
    a: 1,
    b: 2,
    c: 3
  };
}
let {a, b, c} = sample();
```

常见用法：作为函数参数

```
function sample({id, name, age, sex, height, photo}) {
  // do something
}
sample({
  id: 1,
  name: 'meathill',
  age: 33,
  sex: MALE,
  height: 181,
  photo: './photo.jpg'
});
```

变量解构

Destructuring

按照一定模式，从数组和对象中提取值。

```
// 解构数组
let [a, b, c] = [1, 2, 3];
// a = 1;
// b = 2;
// c = 3;
```

箭头函数

```
let f1 = p1 => {
  // do something
};

let f2 = p1 => p1 * 100;
// 等价于
let f2 = p1 => {
  return p1 * 100;
}

let f3 = (p1, p2) => {
  // do something
};
```

几个常见的点：

1. 这里的类仍然是原型继承
2. `class` 关键词不存在变量提升
3. 子类构造函数里，必须 `super()` 之后才有 `this`

CLASS

```
class A {
  constructor() {
    // 构造函数

    // 声明实例属性
    this.var = '';
  }

  method1() {

  }

  method2() {
```

默认值

```
// profile.js
export default 'Meathill';
export let age = 33;
export let weight = 100;

// user.js
import anyname from './profile.js';
```

MODULE

终于有模块机制了！

```
// profile.js
export let name = 'Meathill';
export let age = 33;
export let weight = 100;

// user.js
import * as user from './profile'
// user.name = 'Meathill';
// user.age = 33;
// user.weight = 100;
// 亦可使用解构方法
import {name, age, weight} from './profile'
```

模板字符串

```
let name = 'Meathill';
let age = 33;
let favor = 'Gakki';

alert(`Hi, I'm ${name}, I'm ${age}, I like ${favor}`);
// Hi, I'm Meathill, I'm 33, I like Gakki
```

1. 支持运算符，如 `Hi, ${user.name}` `${a * 3}`
2. 支持换行
3. 尽量不要在里面使用复杂的运算

测试与单元测试

单元测试

什么是单元测试?

在计算机编程中，单元测试（英语：Unit Testing）又称为模块测试, 是针对程序模块（软件设计的最小单位）来进行正确性检验的测试工作。程序单元是应用的最小可测试部件。

Wiki

怎么写单元测试?

以 Node.js 为例

```
const assert = require('assert');

assert.equal(1, '1'); // 1 == '1' true
```

核心概念：

1. 断言
2. 边界条件

我对测试的态度

1. 开发人员必须了解测试
    1. 测试用例
    2. 边界条件
    3. 回归测试
2. 开发人员应该自己写测试
    1. 稳定质量
    2. 方便重构

我以前没写过，现在要怎么开始?

1. 如果要重构，先写测试
2. 比较重要的操作，先写测试
3. 如果要修复 Bug，先写测试

总之，测试不嫌少，能写赶紧写。

测试驱动开发

1. Test Driven Development
2. 先想怎么用，而不是先想怎么写
3. 适用于前期规划项目

工具

- Mocha.js
- Should.js

CODING

Q&A

完整项目代码仓库：

- Github meathill-freelance/date-picker
- 线上文档

参考阅读：

- Using a package.json
- Stylus
- ES6
- Wiki ECMAScript
- 阮一峰 测试框架 Mocha 实例教程