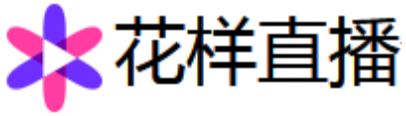
ES6开发体系实践

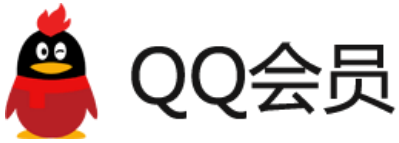
张成文(ouven)

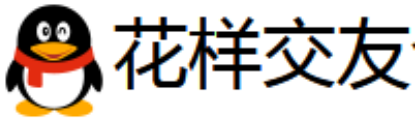
腾讯 社交网络事业群 web前端工程师

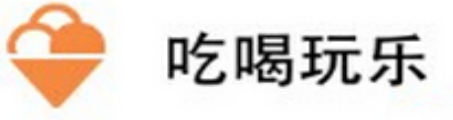
Github: http://github.com/ouvens
Blog: http://ouvens.github.io/
Weixin: ouvenzhang

简介









研究方向：

前端响应式页面设计、工程构建组件化、mv*设计实现、web前端优化、ES6开发体系、前端开发知识体系等

专题： 未来的前端

向未来兼容——ES2015+/TypeScript 开发 Node.js 项目

使用 ES2015+ 特性开发项目可以大大提高开发效率，尤其是使用 async/await 特性来解决异步的问题，借助 Babel 编译，现在就可以使用未来的一些特性。接着介绍如何解决断点调试编译后的代码，以及如何将错误信息定位到源代码下。最后介绍使用 TypeScript/Flow 类型推断和检测来辅助大型项目的开发。

TypeScript 和 ECMAScript 6 实践

过去两年，SitePen 的团队在 ES6/ES2015 和 TypeScript 的使用方面积累了很多经验。本次演讲希望尽可能在较短的时间内讲清楚这些经验，让大家快速了解更有用的语言特性，并了解它们在构建应用和 Dojo 2 方面的实际作用。

TypeScript = ES6 + 强类型 + Annotation

内容概要

一、ES6简介

2015.6.17，ES第6版本正式发布，被命名为ecmascript 2015，现在来说也不是什么新东西了。

官方ES6文档：

ES6 http://www.ecma-international.org/ecma-262/6.0/

推荐书籍


ES6-in-Depth
深入浅出ES6
Jason Orendorff Benjamin Peterson Nick Fitzgerald 等著
刘振涛 午川 译
ES6
InfoQ


ECMAScript 6 Primer
ES 6标准入门
（第2版）
阮一峰 著

ES6的设计三个理念

特性借鉴

特性补充

特性增强

特性借鉴

字符串模板	• 字符串模板借鉴了mustache等模板字符串功能
集合	• 集合借鉴自Python语言的集合collection对象
箭头函数	• 箭头函数借鉴自coffeescript的箭头函数
promise	• 和deferred类似的功能Promise
for of	• 借鉴了c++、java、python等语言的for-of语句

特性补充

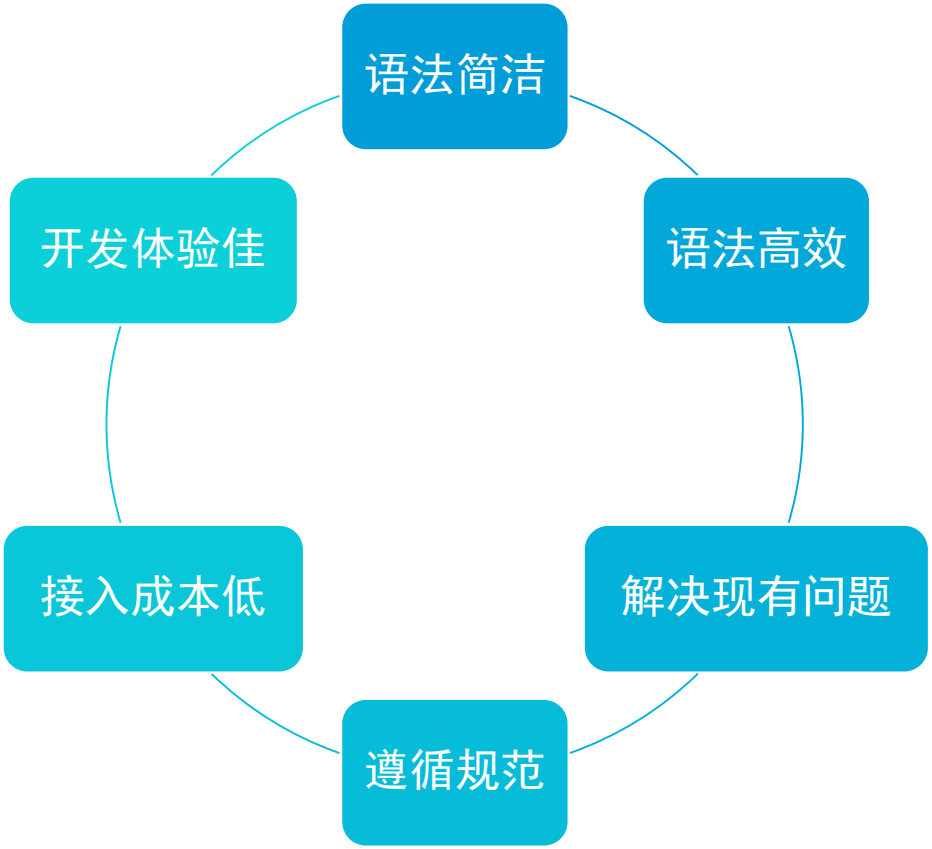
模块
- 补充了js缺少的模块化规范，import/export

class
- 补充了js没有class被嘲讽的不足

特性增强

- 迭代器
  - 更高效的迭代器iterator
- rest
  - 函数的不定参数
- default
  - 函数的默认参数
- 解构
  - 增强的解构赋值
- proxy
  - 代理对象
- Generator
  - 增强的高效异步方案，未来的async/wait

SWOT-优势



规范举例—简洁

```
// bad
"use strict";
var fn = function fn(v) {
    return console.log(v);
};

// good
var fn = (v => console.log(v));
```

规范举例—规范

```
// normal
const AirbnbStyleGuide = require('./AirbnbStyleGuide');
module.exports = AirbnbStyleGuide.es6;


// best
import { mod } from './AirbnbStyleGuide';
export default mod;
```

规范举例—开发体验

```
// normal
function Queue(contents = []) {
  this._queue = [...contents];
}
Queue.prototype.pop = function() {
  const value = this._queue[0];
  this._queue.splice(0, 1);
  return value;
}

// good
class Queue {
  constructor(contents = []) {
    this._queue = [...contents];
  }
  pop() {
    const value = this._queue[0];
    this._queue.splice(0, 1);
    return value;
  }
}
```

SWOT-劣势

兼容性差，浏览器和node支持不完全；

SWOT-机遇

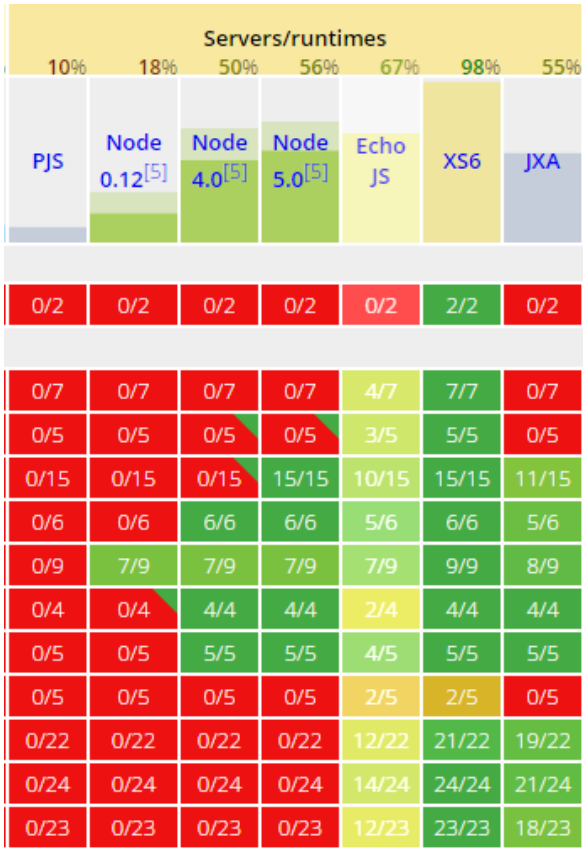
- ✓ ES6 transform 可以转为ES5;
- ✓ Nodejs在不断添加ES6的支持;
- ✓ 新的浏览器在不断添加ES6支持；

es6 transfrom支持

	Compilers/polyfills				
26%	58%	74%	42%	60%	16%
Current browser	Traceur	Babel + core-js[1]	Closure	Type-Script + core-js	es6-shim
0/2	0/2	0/2	0/2	0/2	0/2
0/7	4/7	4/7	4/7	5/7	0/7
0/5	4/5	3/5	2/5	4/5	0/5
0/15	15/15	13/15	12/15	4/15	0/15
0/6	6/6	6/6	4/6	6/6	0/6
6/9	9/9	9/9	6/9	3/9	0/9
0/4	2/4	4/4	4/4	4/4	2/4
0/5	4/5	4/5	3/5	3/5	0/5
0/5	3/5	3/5	0/5	0/5	0/5
0/22	20/22	21/22	18/22	15/22	0/22

74%支持，
常用特性90%支持

ES6 in nodejs

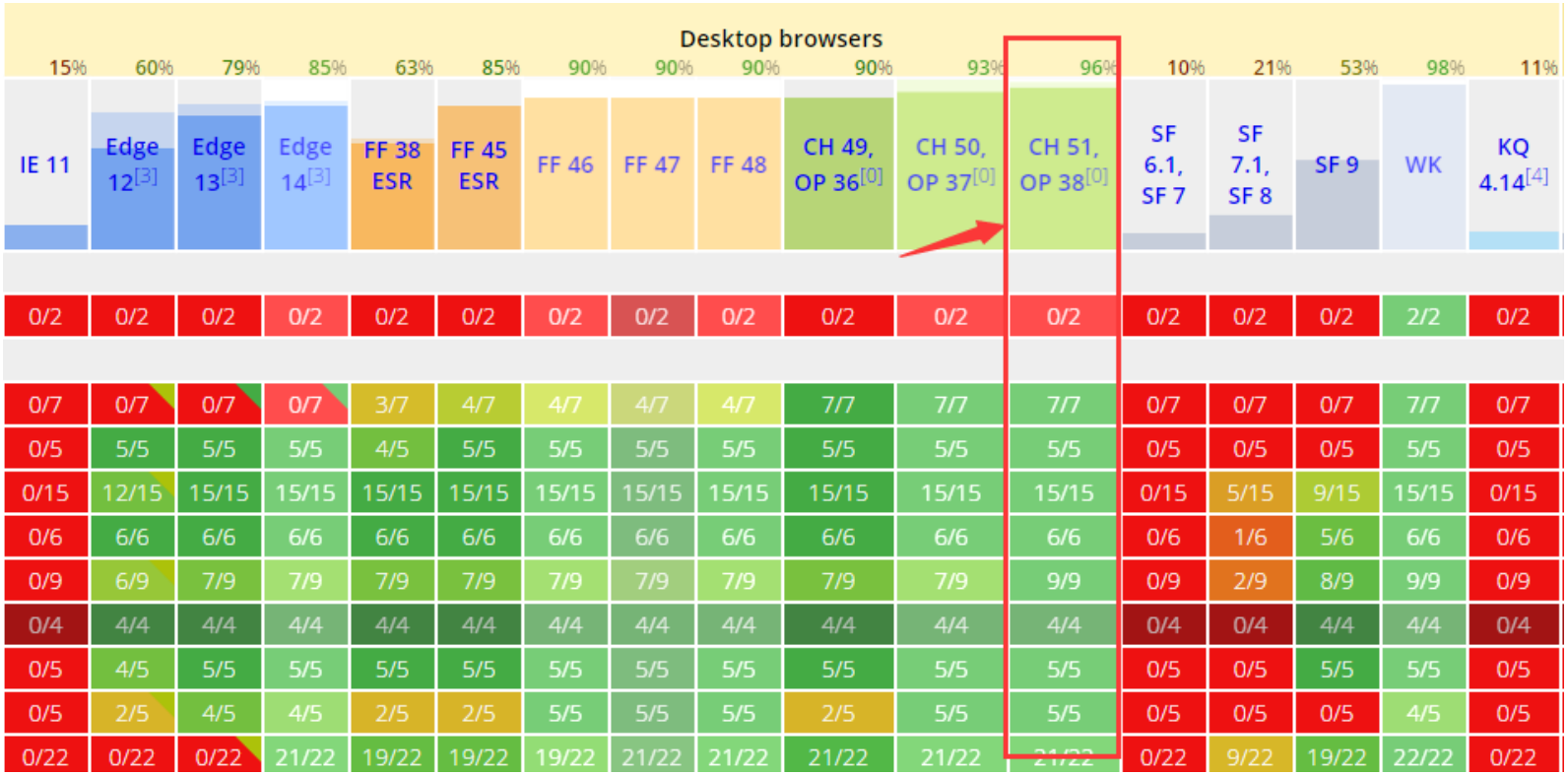


Servers/runtimes						
10%	18%	50%	56%	67%	98%	55%
PJS	Node 0.12[5]	Node 4.0[5]	Node 5.0[5]	Echo JS	XS6	JXA
0/2	0/2	0/2	0/2	0/2	2/2	0/2
0/7	0/7	0/7	0/7	4/7	7/7	0/7
0/5	0/5	0/5	0/5	3/5	5/5	0/5
0/15	0/15	0/15	15/15	10/15	15/15	11/15
0/6	0/6	6/6	6/6	5/6	6/6	5/6
0/9	7/9	7/9	7/9	7/9	9/9	8/9
0/4	0/4	4/4	4/4	2/4	4/4	4/4
0/5	0/5	5/5	5/5	4/5	5/5	5/5
0/5	0/5	0/5	0/5	2/5	2/5	0/5
0/22	0/22	0/22	0/22	12/22	21/22	19/22
0/24	0/24	0/24	0/24	14/24	24/24	21/24
0/23	0/23	0/23	0/23	12/23	23/23	18/23

最新版56%支持

http://ouvens.github.io/frontend-javascript/2015/12/06/es6-in-nodejs.html

ES6 in browser



最新版chrome 51支持96%的特性

SWOT-挑战

	语法简洁高效	插件多，接入容易	遵循js语法规范	特性完善	兼容性
coffeescript	✓	X	X	X	X
jsx	X	✓	X	✓	X
typescript	✓	✓	✓	✓	X
es5	X	✓	✓	X	✓
es6	✓	✓	✓	✓	X

是时候使用ES6了

二、ES6特性与开发规范

EcmaScript6 全规范（含node）

- 一、类型规范
- 二、字符串模板
- 三、数组类型
- 四、解构类型
- 五、函数
- 六、arrow箭头函数
- 七、对象
- 八、类
- 九、模块
- 十、Iterators 和 Generators
- 十一、属性访问
- 十二、map + set + weakmap + weakset 数据结构
- 十三、promise、symbols、proxies
- 十四、统一码
- 十五、进制数支持
- 十六、不建议使用reflect对象和tail calls尾调用

小结

ES6的优秀特性需要与开发规范相结合

https://github.com/ouvens/es6-code-style-guide

三、ES6兼容理论

1、Node端兼容性

2、浏览器端兼容性

ES6 in nodejs

ES6特性	Nodejs兼容性
let,const,块	strict模式支持
class类	strict模式支持
Map，Set 和 WeakMap，WeakSet	支持
generators	支持
二进制转换	支持
对象字面量扩展	支持
promise	支持
String对象新API	支持
symbols	支持
字符串模板	支持

最新版56%支持

http://ouvens.github.io/frontend-javascript/2015/12/06/es6-in-nodejs.html

性能小测试

```
'use strict'
let i = 0;
let t1 = +new Date(),
    t2;

while(i++ < 1000000){
    const a = 1;
    const b = '1';
    const c = true;
    const d = {};
    const e = [];
}

t2 = +new Date() - t1;
console.log(t2);
```

使用let，const声明变量的速度比var快了约65%左右

ES5运行时间	ES6运行时间
52-53ms	33-34ms

```
'use strict'
let i = 0;
let t1 = +new Date(),
    t2;

while(i++ < 100000){
    class A{
        constructor() {
           this.name = 'ouven';
        }
        getName(){
            return this.name;
        }
    }

    const a = new A();
    a.getName();
}

t2 = +new Date() - t1;
console.log(t2);
```

可见使用Nodejs的Class比ES6的function构造方法慢约25%

ES5运行时间	ES6运行时间
1179-1211ms	1411-1442ms

```
'use strict'
let i = 0;
let t1 = +new Date(),
    t2;

let vars = {
    name: 'ouven',
    address: 'tencent'
};

while(i++ < 1000000){

    let str = `string text ${vars.name} string ${vars.address}`;

}

t2 = +new Date() - t1;
console.log(t2);
```

ES6的字符串模板整体上性能就相对ES5的字符串拼接慢了很多。相对于数组join的方式就更慢了。

ES5运行时间	ES6运行时间
8ms	59-61ms

但是一点也不影响

ES6 in babel

	Compilers/polyfills				
26%	58%	74%	42%	60%	16%
Current browser	Traceur	Babel + core-js[1]	Closure	Type-Script + core-js	es6-shim
0/2	0/2	0/2	0/2	0/2	0/2
0/7	4/7	4/7	4/7	5/7	0/7
0/5	4/5	3/5	2/5	4/5	0/5
0/15	15/15	13/15	12/15	4/15	0/15
0/6	6/6	6/6	4/6	6/6	0/6
6/9	9/9	9/9	6/9	3/9	0/9
0/4	2/4	4/4	4/4	4/4	2/4
0/5	4/5	4/5	3/5	3/5	0/5
0/5	3/5	3/5	0/5	0/5	0/5
0/22	20/22	21/22	18/22	15/22	0/22

ES6 in babel

ES6特性	兼容性
箭头函数	支持
类的声明和继承	支持
增强的对象字面量	支持
字符串模板	支持
解构	支持
参数默认值，不定参数，拓展参数	支持
let与const	支持
for of	支持
iterator, generator	支持
Map，Set 和 WeakMap，WeakSet	不支持
Promises、Math，Number，String，Object 的新API	不支持
模块 export & import	支持
symbol	不支持

74%支持，
常用特性90%支持

http://ouvens.github.io/frontend-javascript/2015/10/16/es6-under-babel.html

小结

执行环境	兼容性
node	56%（node 5.0）
原生浏览器	26%
Babel transform node	80%以上
Babel transform browser	74%

- ✓ Node端目前支持一半以上的es6特性
- ✓ 原生浏览器支持较差，约26%
- ✓ 毕竟ES5的特性浏览器支持也不是100%

四、ES6实践方案

1、Node端实践方案

2、浏览器端实践方案

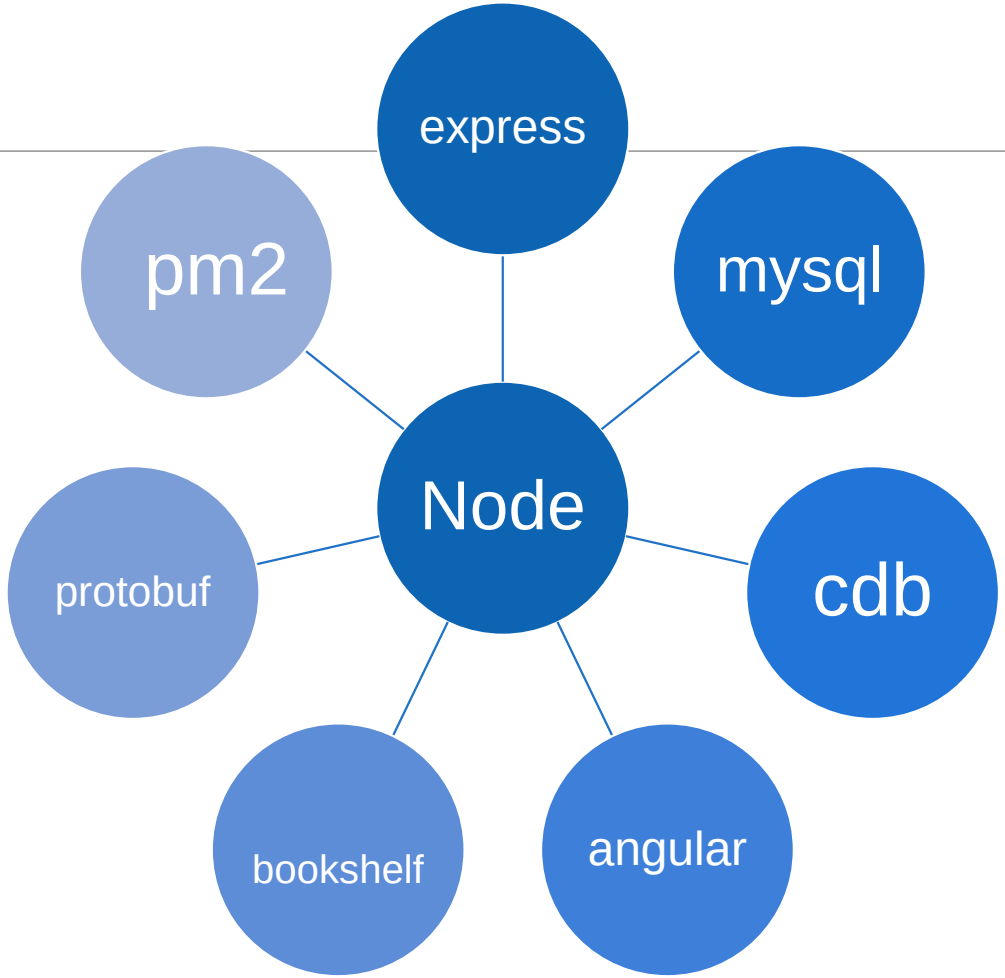
4.1 Nodejs端开发实现

花样直播

秩序监管系统基本技术模块：

M(ysql)**E**(xpress)**A**(ngular)**N**(ode)

使用了**CDB**(云数据库)来代替mysql，主要服务模块如下：

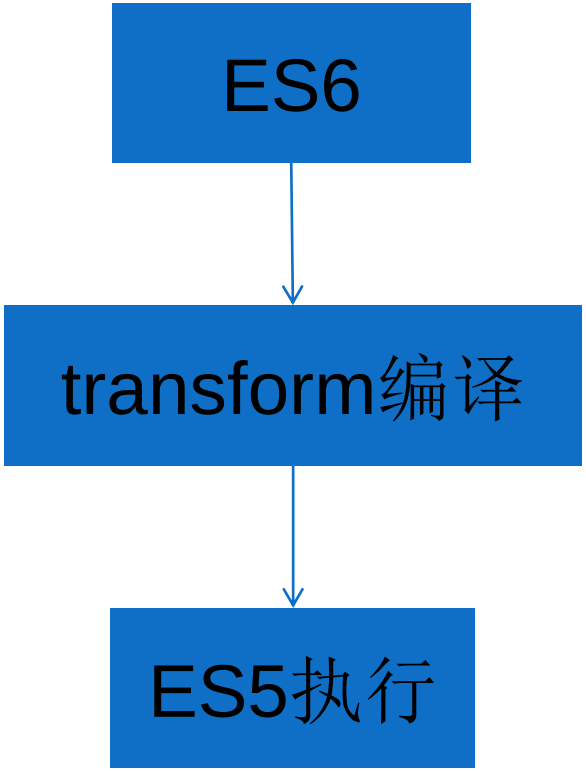

express
pm2
mysql
Node
protobuf
cdb
bookshelf
angular

- 后台开发框架：Express
- MySql数据库驱动器：node_mysql
- 数据库ORM框架：Bookshelf
- protobuff解析模块：protobufjs
- buffer处理模块：node-struct
- 会话管理模块：express-seesion
- 单元测试工具：mocha
- 进程管理器：PM2（主要用于：守护进程、热重启以及输出服务器本地日志，解决负载突发情况自动重启等功能）
- 异步流程控制模块：async（这里没有用promise）

ES6执行方案

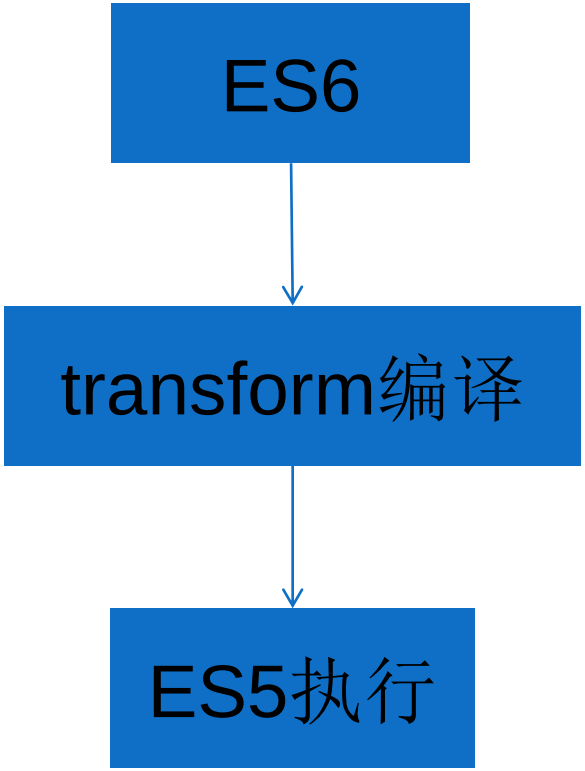
使用babel转换node脚本

Node端可以使用 node –harmony 查看node已经支持的特性，也可以开启对新特性的支持。



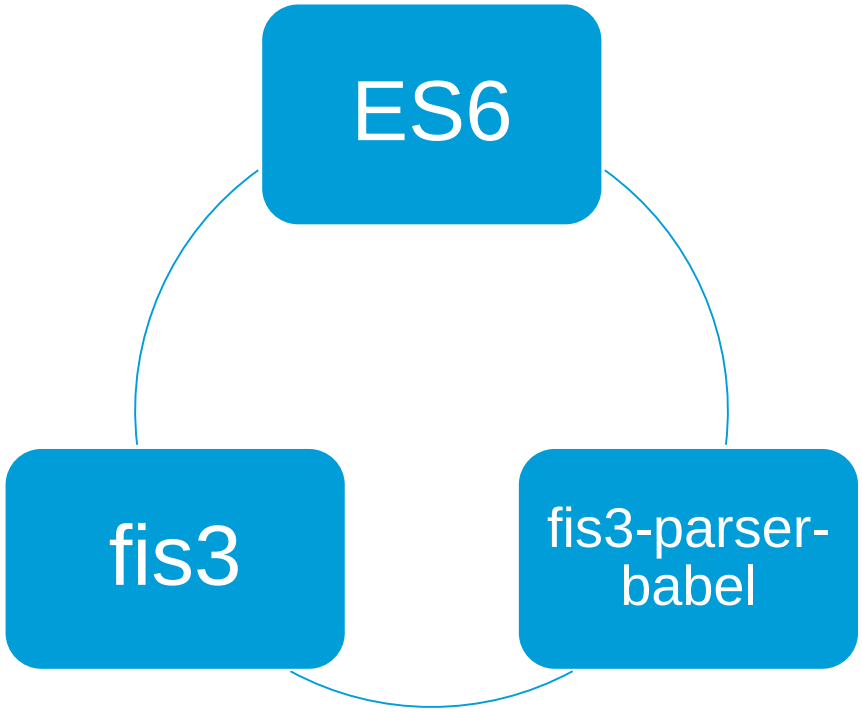
```
historyRecords.fetchBy(query.orderfield , {
    page,
    limit,
    order,
    where
}, {}).then((data) => {
    let historyrecords = data.collection.toJSON();
    let result = [];
    for (let record of historyrecords) {
        record.pic_ts = record.pic_ts * 1000;
        result.push(record);
    }
    res.json({
        retcode: 0,
        pagination: data.pagination,
        result
    });
}).catch((error) => {
    console.log("error:", error);
    res.status(500).send(error.message);
});
```

4.2 浏览器端实践方案



现有模式组合

- Gulp + webpack + gulp-babel + es6
- fis3 + fis3-paser-babel + es6
- react + webpack + es6
- typescript + vscode + es6 + ts transform



可以结合fis3的构建优势开发

fis3 + fis3-paser-babel + es6

添加自己封装的fis3 babel插件

```
.match('partials/**.js', {
    parser: fis.plugin('babel'),
    release: '$0',
    rExt: '.js'
})
```

挑战：

浏览器端直接调试es6断点不是很方便，vscode可以断点调试。

```
class Area {
    constructor(items = {}) {
        this.Items = items;
        this.selectOptions = ["省份", "城市", "地区"];
    }
    add(id, iArray) {
        this.Items[id] = iArray;
    }
    Exists(id) {
        if (typeof(this.Items[id]) == "undefined") return false;
        return true;
    }
}

const area = new Area();
```

不得不注意的坑

1，兼容性约束。

使用特性时要注意是否支持

2，SyntaxError: Block-scoped declarations (let, const, function, class) not yet supported outside strict mode

必须在严格模式下才能启用let const class关键字

3，node下面ES6函数增强不能使用默认值

4，特性使用不和场景

```
const obj = {
    method1: (a, b) => {
        console.log(a, b);
    },
    method2: (c) => {
        console.log(c);
    }
};
```

```
const obj = {
    method1 (a, b){
        console.log(a, b);
    },
    method2 (c){
        console.log(c);
    }
};
```

5，注意差异性

```
try{
    let a = 1;
}catch(e){
    console.log(e.msg)
}

// Uncaught ReferenceError
console.log(a);
```

实践心得：

1、前端测速来看，和ES5相比并没有较大区别

2、node端使用情况来看，暂无运行时问题

2、ES6语法本身的优势，代码相对ES5更严谨简洁

3、构建尽量支持到ES6的透明化transform

4、ES6是基于规范的，未来会被支持

5、ES6即将不仅仅是语法糖，例如koa2、angular2的下一代框架等

五、ES6的发展

异步流程控制模块：async（这里没有用promise）

Promise同时处理多个异步请求时，需要循环定义多个promise对象，显得不优雅，所以这里用了封装使用方便的async。

```
async.eachSeries(phones, function(phone, callback) {
    phone = phone.trim();
    if (phonePattern.test(phone)) {
        sendMsg(phone, text, function(err, res) {
            callback(err, res);
        });
    } else {
        callback(null, phone);
    }
}, function (err, results) {
    console.log('eachSeries results: ', results);
    if(!err) {
        res.json(resConfig.success);
    } else {
        res.json(resConfig.fail);
    }
});
```

代替方案可以是generator或es7的async

推动前端技术革新
紧跟时代步伐

谢谢

Q&A

感谢linkzhu在node上es6的实践反馈和分享

相关参考

http://ouvens.github.io/frontend-javascript/2015/12/06/es6-in-nodejs.html

http://ouvens.github.io/frontend-javascript/2015/10/16/es6-under-babel.html

ES6 http://www.ecma-international.org/ecma-262/6.0/

https://github.com/ouvens/es6-code-style-guide/blob/master/ES6-in-depth.pdf

http://kangax.github.io/compat-table/es6/

https://iojs.org/en/es6.html