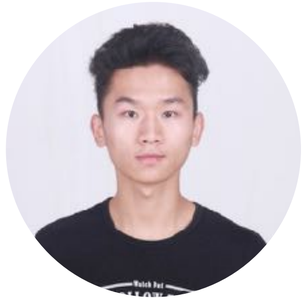
用TypeScript开发大型React应用

吴浩麟

美团点评 | 成都研发部



吴浩麟

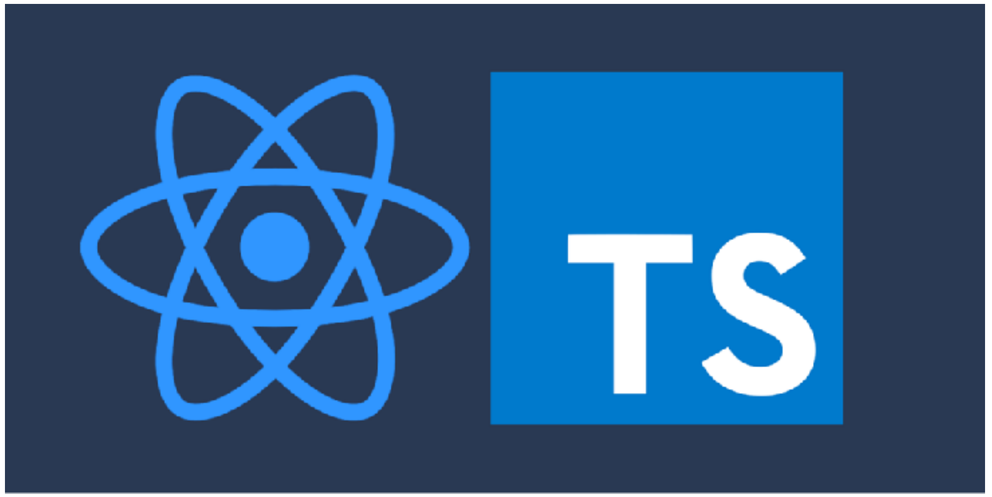
Web前端开发

2015年加入腾讯深圳，参与腾讯课堂项目。

2017年加入美团成都，参与美团餐饮生态的建设。

《深入浅出Webpack》作者

提纲

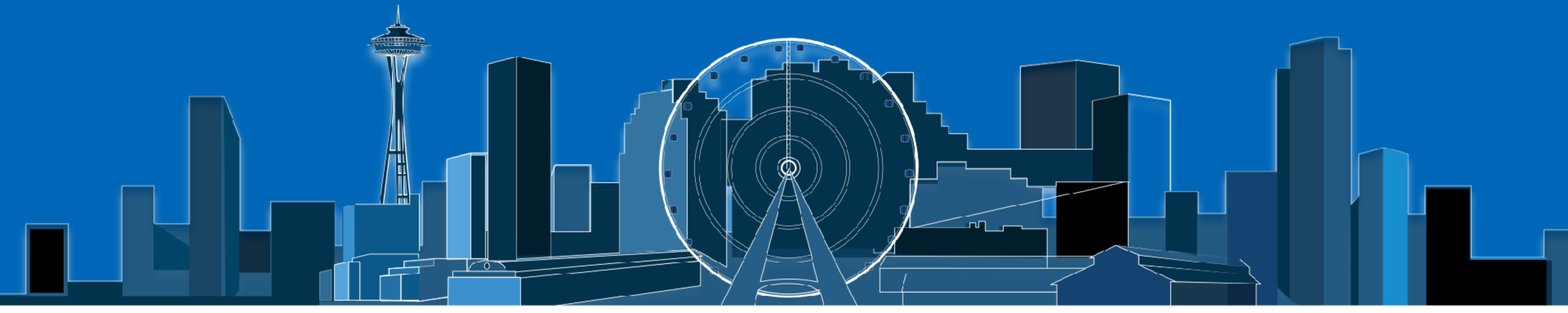


1.为何大型项目适合用TypeScript

2.TypeScript结合React实践

Microsoft

TypeScript



```
// 让指定id的input聚焦
function forcus(id) {
  document.getElementById(id).forcus();
}
forcus('name');
```

```
▶Uncaught TypeError: document.getElementById(...).forcus is not a function
    at forcus (<anonymous>:2:31)
    at <anonymous>:1:1
```

forcus拼写错误，应该是focus

```
▶Uncaught TypeError: Cannot read property 'forcus' of null
    at forcus (<anonymous>:2:30)
    at <anonymous>:1:1
```

当对应id的input元素不存在时，访问null的属性

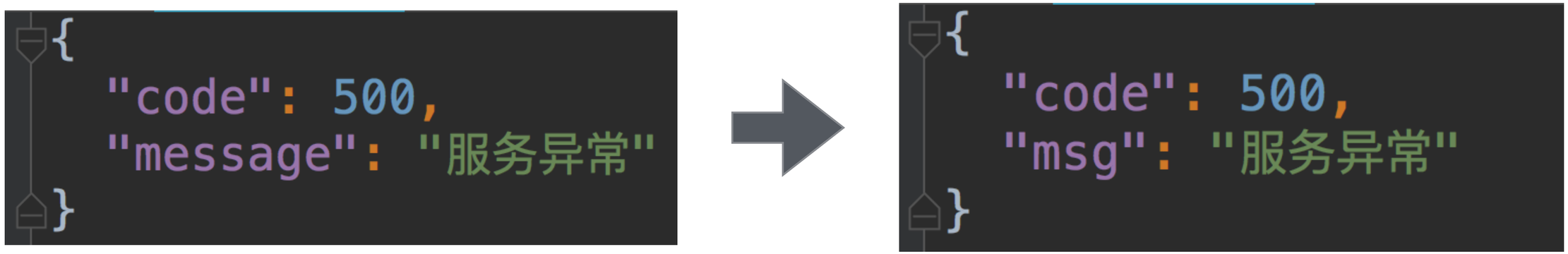
```ts
// 让指定id的input聚焦
function forcus(id: string) {
  document.getElementById(id).forcus();
}
forcus( id: 'name');
```

Error:(3, 33) TS2551: Property 'forcus' does not exist on type 'HTMLElement'. Did you mean 'focus'?

Error:(3, 5) TS2531: Object is possibly 'null'.

TS通过静态类型检查，能把错误从运行时提前到编译时，避免了一些线上BUG的发生

从大数据看来TypeScript起到了减少运行时错误的作用



后端非要把返回数据的message字段改成msg

JavaScript

全局搜索message替换为msg
但这可能会误伤部分也用message但不是读取后端数据逻辑的代码，
需要人工判断替换是否正确。

TypeScript

直接修改表示后端数据结构的接口
TS会自动把需要跟着修改的地方通过编译报错的方式找出来，
精确度100%。

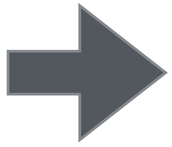
前后端联调

1.Swagger转换成TypeScript

2.Thrift转换TypeScript

3.KoaRouter转换成TypeScript

```
struct Status {
  1:required i32 code;
  2:optional string msg;
}
```



```
interface Status {
  code: number;
  msg?: string;
}
```

社区中已经出现很多自动生成TS的工具，解决前后端类型联调时类型松散的痛点

JavaScript VS TypeScript

JavaScript	TypeScript
语法灵活	静态类型约束但语法比JS啰嗦
可直接运行	编译成JS才能运行

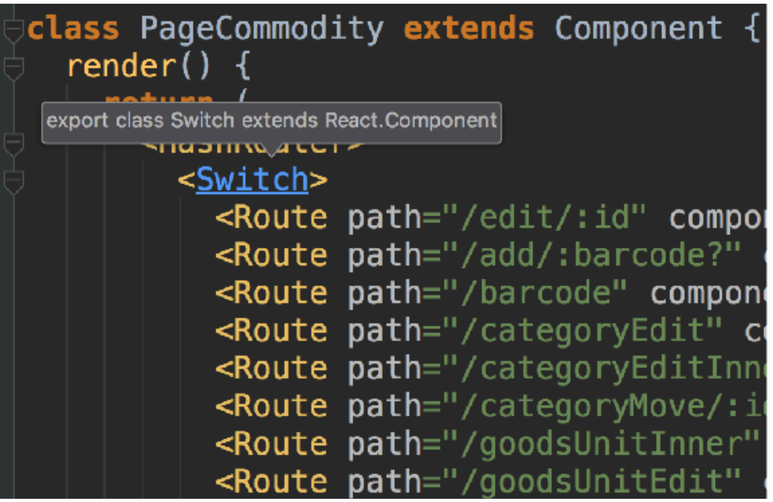
JS和TS各有千秋侧重点不同

JS诞生之初没考虑到如今的Web应用会如此庞大

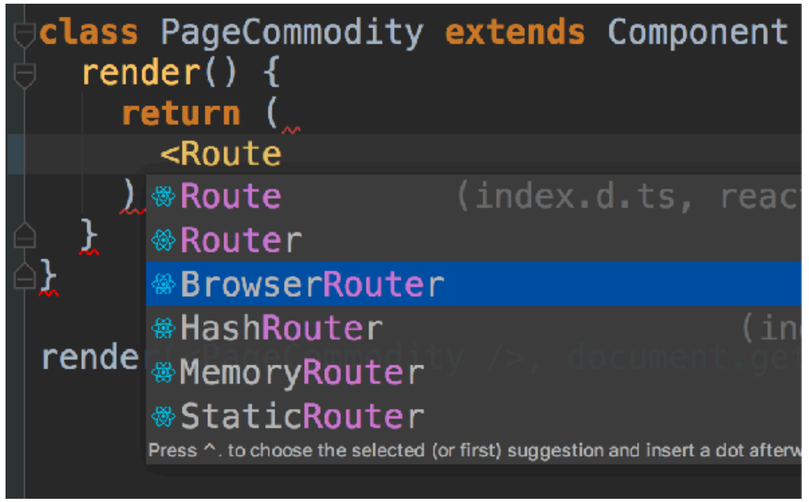
TS通过工程化手段补足JS在应对大型应用时的不足

TypeScript开发体验

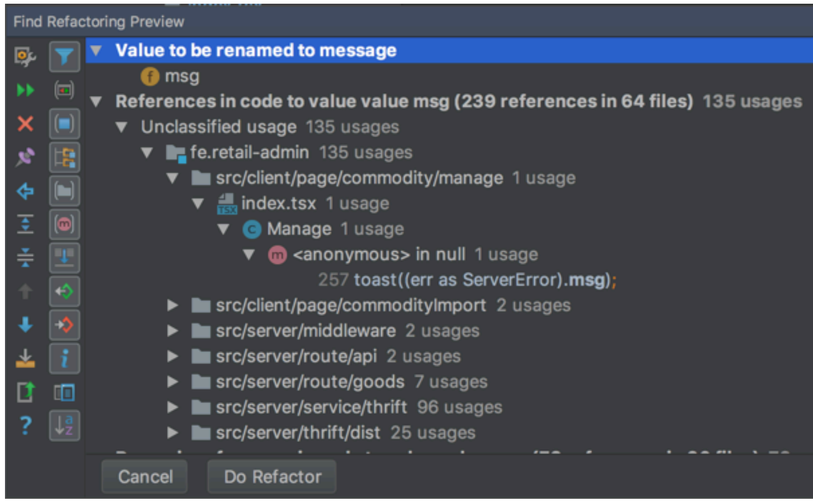
TS语法虽比JS啰嗦，但借助IDE写码时却比JS轻松



文件跳转



智能补全



智能重构





TypeScript VS Flow

	TypeScript	Flow
GitHub 提问数	2400/11200	1500/2200
GitHub PR数	100/5000	60/1200
Npm下载量	720万	290万
三方库定义	3700	340
StackOverflow问题数	38000	900

TypeScript热度高于Flow且还在上升

社区参与度TypeScript远高于Flow

选择流行度高的TS能让你的团队更快的找到更多的优秀人才

谁在用TypeScript

Slack

团队协作桌面应用

VSCode

代码编辑器

IONIC框架

移动App开发框架

白鹭引擎

H5游戏开发引擎

越来越多大型项目选择TS，可以用JS写的应用都可用TS

React+TS实战

接入Webpack构建

webpack.config.js

```
module.exports = {
  resolve: {
    // TypeScript源码文件的后缀是ts、tsx
    extensions: ['ts', '.tsx', '.js']
  },
  module: {
    rules: [
      {
        // 对ts、tsx结尾的文件使用ts-loader转换
        test: /\.tsx?$/,
        use: ['ts-loader'],
      },
    ]
  },
};
```

把ts、tsx文件交给ts-loader转换成js

@types

编译报错	error TS2307: Cannot find module 'react'.
报错原因	react模块由js编写，tsc不知道该模块暴露出的接口类型
解决方法	安装react模块对应的类型描述模块 **@types/react**

经过社区的努力@types已有4.6K个模块，涵盖所有常用模块

@types还用于反哺JS项目，让IDE实现对JS代码的智能补全

公用私有保护

```
class Counter extends Component {

  _dontCallMe() {...}

  callMe() {...}

  render() {...}
}
```

在JS中想隐藏组件实例上的方法，按照社区的规范通过_开头，以向使用者表示改方法不可调用，但实际上使用者还是可以调用。

```
class Counter extends Component<ICounterProps> {

  private dontCallMe() {...}

  public callMe() {...}

  render() {...}
}
```

在TS中通过private修饰符来表示改方法不可调用，如果使用者调用会编译报错

React组件属性检查

```ts
interface ICounterProps {
  onChange: (em: string) => void;
}

export class Counter extends Component<ICounterProps> {
  render() {...}
}
```

TS通过接口进行编译时检查

写码时即抛出错误

编译出的代码无类型检查，体积更小，性能更快

能描述清onChange的参数类型，描述更明确

```ts
import * as PropTypes from 'prop-types';

class Counter extends Component {
  static propTypes = {
    onChange: PropTypes.func
  }

  render() {...}
}
```

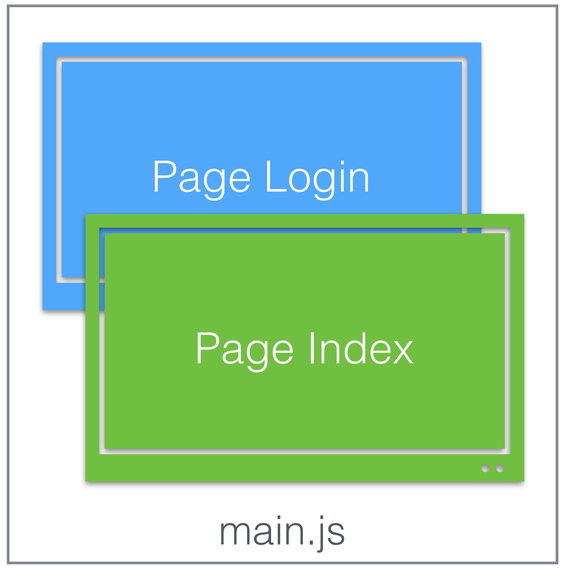
JS通过PropTypes库运行时检查

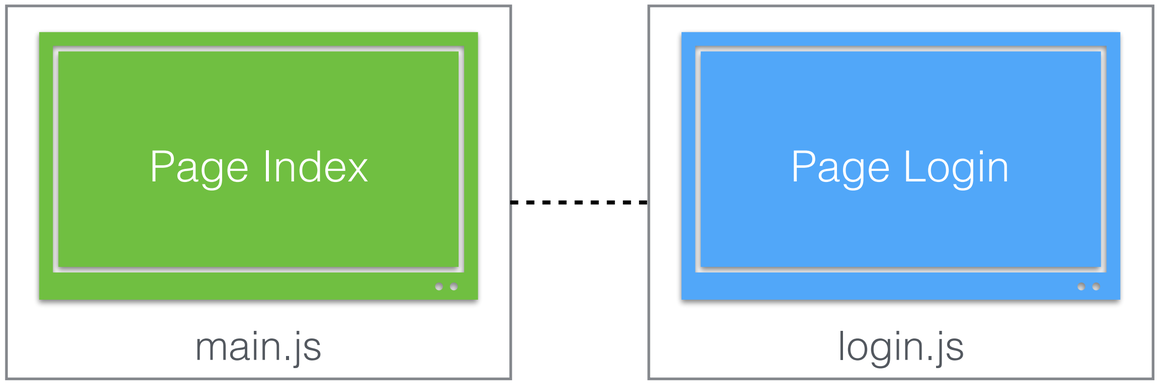
运行时检查类型，影响性能

导入了prop-types，体积更大

只能描述清楚onChange是func类型

ReactRouter异步按需加载





没有异步分割时所有页面的代码被打包到一个文件中；首页访问Index时，Index和Login的代码都被加载；首次加载资源大导致首屏加载慢印象用户体验

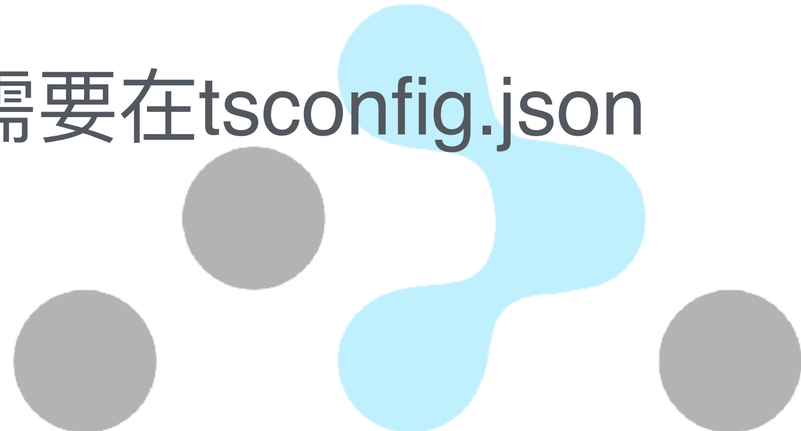
采用异步分割后2个页面被分别打包成2个文件，当首页访问Index页时只加载main.js;当切换到Login页时再去加载login.js,加载完后展示Login页；这提升了首页加载速度

ReactRouter异步按需加载

```
<BrowserRouter>
  <Switch>
    <Route path='login' component={getComponentAsync( load: () => import('./page/Login'))}/>
    <Route component={PageIndex}/>
  </Switch>
</BrowserRouter>
```

通过Webpack的异步分割语句**import()**,实现抽离Login页到独立的文件。

为了让tsc编译TS到JS时保留**import()**语句，需要在tsconfig.json中配置 **"module": "esnext"**



按需加载UI组件库

在TS项目中使用Antd组件库时，这样导入Button组件

```
import {Button} from 'antd';
```

会发现Antd提供的所有组件都被打包进去了，原因是Antd入口文件导出了所有组件

```
{
  test: /\.tsx?$/,
  use: ['ts-loader', 'ui-components-loader'],
},
```

修改Webpack Loader配置插入**ui-components-loader**，自动完成以下转换

```
import {Button} from 'antd';
```

```
import Button from 'antd/lib/Button';
import 'antd/lib/Button/style/index.css';
```

从而实现只打包进用到的组件，减少输出文件大小

还支持Material-UI、Elemental-UI等等UI库

数据管理库最佳选择mobx

```
function todos(state = [], action) {
  switch (action.type) {
    case 'ADD_TODO':
      return [...state, {
        id: action.id,
        text: action.text,
      }];
    case 'TOGGLE_TODO':...
    default:
      return state
  }
}
```

redux在运行时通过**action.type**去区分当前action类型。

无法用TS描述清楚action的类型

```
interface Todo {
  id: number;
  text: string;
}

class TodoList {
  @observable todos: Todo[] = [];

  @action
  addTodo(todo: Todo) {
    return this.todos.push(todo);
  }
}
```

mobx通过在原生class语法上加入注解来编写，可以简单清晰的描述任何类型；

mobx本身就是用TS编写。

空格还是Tab是写码界一个永无休止的争论

除此之外还有：

- 单词间空几个空格?
- 什么时候该换行?
- 单引号还是双引号?

。。。

用TSLint统一代码风格

TSLint是一个可配置的TS代码检查工具，内置很多检查规则，根据配置的规则找出不符合规则的代码

Tab还是空格？全部采用recommended

停止无意义的争论

tslint-react针对JSX语法进行检查，让JSX语句更美观

单元测试

单测含义	分别对组成系统的模块单元编写用例，保证模块正常
单测作用	保证代码质量，重构时保证功能正常。 驱动开发人员划分系统为低耦合的模块，降低维护成本
Why Jest	内置功能齐全、Snapshot让React组件测试更快更方便、并行测试速度更快

接入Jest

package.json

```
{
  "transform": {
    "^.+\\.tsx?$": "<rootDir>/node_modules/ts-jest/preprocessor.js"
  },
  "testRegex": "(/__tests__/.*\\.test)\\.tsx?$",
  "moduleFileExtensions": [
    "tsx",
    "ts",
    "js",
    "json"
  ]
}
```

把ts文件交给ts-jest处理后可直接运行单测

自动为React组件生成文档

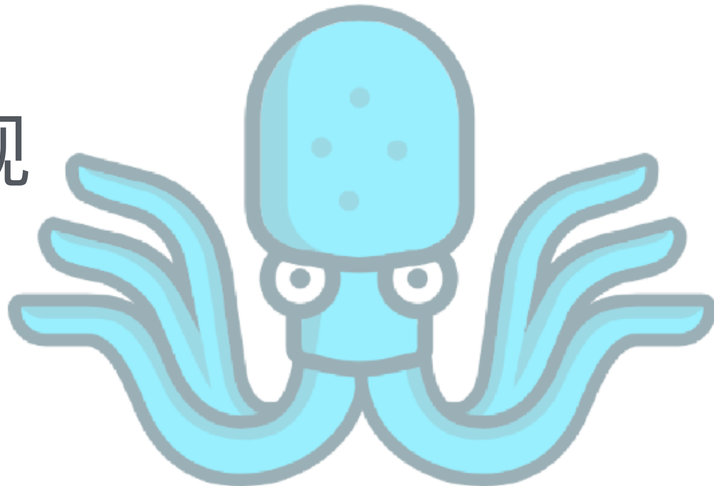
```tsx
interface ICounterProps {
  /**
   * 当值发生变化时回掉
   * @param {string} value 新值
   */
  onChange: (value: number) => void;
}

/**
 * 步进器-用作增加或者减少当前数值。
 */
export class Counter extends Component<ICounterProps> {
  render() {...}
}
```

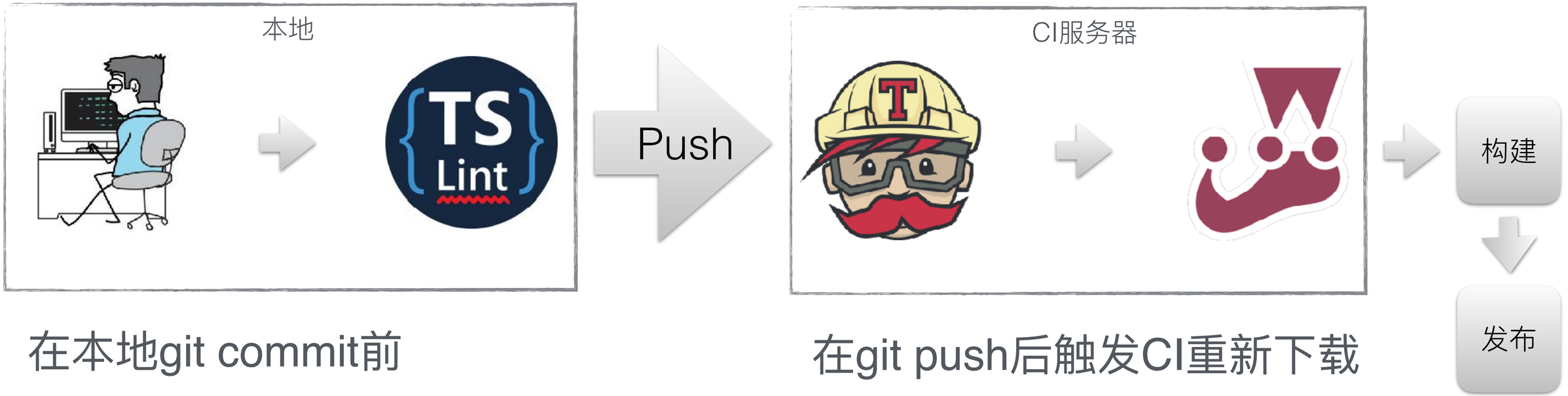
从源码注释中提取信息自动生成组件文档，不用另外编写文档

Styleguide结合react-docgen-typescript实现

```tsx
module.exports = {
  components: 'src/components/[A-Z]*/index.tsx',
  propsParser: require('react-docgen-typescript').parse,
};
```



持续集成



在本地git commit前

运行TSLint保证代码风格

安装husky模块后自动注册

git hook实现

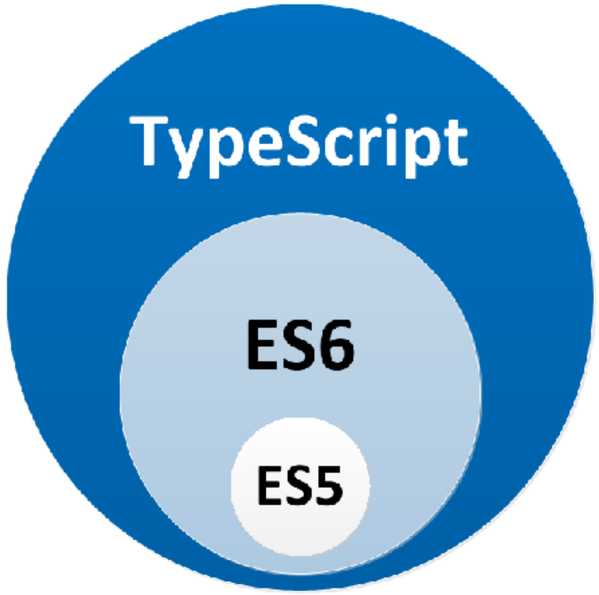
在git push后触发CI重新下载

最新代码运行Jest单测

CI平台监听Webhooks来知

道有新的代码被Push

从JavaScript到TypeScript



TS是JS的超级
把JS文件后缀改成TS即可

开启allowJS

开启allowJS后可以让TS和JS并存，渐进式的把JS用TS重写

```
static propTypes = {
  onChange: PropTypes.func
}
```

```
interface IButtonProps {
  onChange: (val: number) => void;
}
```

JS的PropTypes自动
转化为TS的interface

招人

美团成都研发部迅速扩张中

大量招聘各种研发岗位

发送简历邮箱 **gwuhaolin@icloud.com** 内推

Q&A

Thanks

既往不恋，纵情向前