#### ES2019提案预览

##### 之Angular2特性

摘要

Angular2特性——pipeline

@link <https://github.com/tc39/proposal-pipeline-operator>

Angular2特性——Observable

@link <https://github.com/tc39/proposal-observable>

Angular2特性——export

@link <https://github.com/tc39/proposal-export-default-from>

Angular2特性——访问控制符

@link <https://github.com/tc39/proposal-private-declarations>

Angular2特性——Declarations in Conditionals

@link <https://github.com/tc39/proposal-Declarations-in-Conditionals>

正文

什么是流水线（Pipeline）？

Pipeline是Linux系统Shell语言的一个特性，能够将若干处理函数用vertical bar相连，并顺序执行。Angular2在模板引擎中加入pipeline特性，比如国际化中的translate指令。示例如下：

function doubleSay (str) {

return str + ", " + str;

}

function capitalize (str) {

return str[0].toUpperCase() + str.substring(1);

}

function exclaim (str) {

return str + '!';

}

let result = exclaim(capitalize(doubleSay("hello")));

result //=> "Hello, hello!"

let result = "hello"

|> doubleSay

|> capitalize

|> exclaim;

result //=> "Hello, hello!"

什么是Observable？

Angular2扩展模块@angular/rxjs中提供了用于异步调用的Observable对象和用于同步调用的Promise对象。该提案建议将Observable引入JavaScript，示例如下：

let subscription = commandKeys(inputElement).subscribe({

next(val) { console.log("Received key command: " + val) },

error(err) { console.log("Received an error: " + err) },

complete() { console.log("Stream complete") },

});

什么是export？

该提案建议export关键字支持导出ModuleNameSpace，示例如下：

export someIdentifier from "someModule";

export someIdentifier, { namedIdentifier } from "someModule";

访问控制符

TypeScript支持private、protected、public等访问控制符。该提案建议将访问控制符特性引入JS，示例如下：

// https://github.com/Polymer/lit-html/blob/1a51eb54/src/lib/parts.ts

private #createPart;

class AttributeCommitter {

//...

[#createPart]() {

return new AttributePart(this);

}

}

class PropertyCommitter extends AttributeCommitter {

[#createPart]() {

return new PropertyPart(this);

}

}

Declarations in Conditionals

TypeScript允许Declarations in Conditionals特性，示例如下：

for(let k in valueMap){

let param = {

label: valueMap[k],

value: k

}

arr.push(param);

}

JavaScript不允许该特性：

class Foo {

get data() {

let result = [];

/\* ... do some expensive work ... \*/

return result;

}

}

let foo = new Foo;

if (foo.data) {

for (let item of foo.data) {

/\* A \*/

}

} else {

/\* B \*/

}

JS添加该特性后，上述代码可简写成：

class Foo {

get data() {

let result = [];

/\* ... do some expensive work ... \*/

return result;

}

}

let foo = new Foo;

if (let data = foo.data) {

for (let item of data) {

/\* A \*/

}

} else {

/\* B \*/

}

什么是代理？

JavaScript作为一门编程语法,对属性(或者说是字段)没有提供原生的 setter & getter 函数.这些功能在其他的,类似于 Java,.Net,Objective-C里都有提供.

所以,proxy 更像是 JavaScript 提供的一个统一的属性的getter & setter 的监测接口.

作用：

1. 代理可以作为一个“漏斗”，定义一组公共的getter、setter方法，动态地附加到需要的对象上
2. 代理可以用于封装原生的HTML API.

□