TypeScript

一.特点

微软开发

Js的超集

ES6

Angular2框架由typeScript开发

二.内容

1.ts优势:

2.安装环境

3.ts概念语法和特性

1Ts优势:

1)支持ES6(2015年) , 客户端脚本语言的规范

2)强大的IDE支持 :类型检查\语法提示\重构(修改变量或方法的名字)

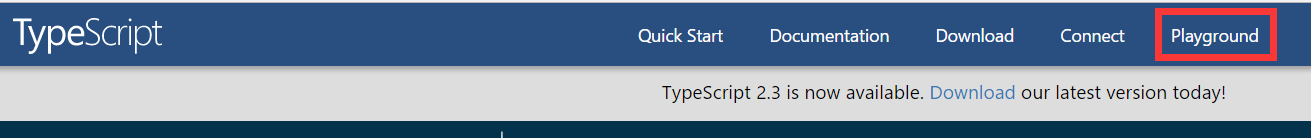
3)angular2 的开发语言

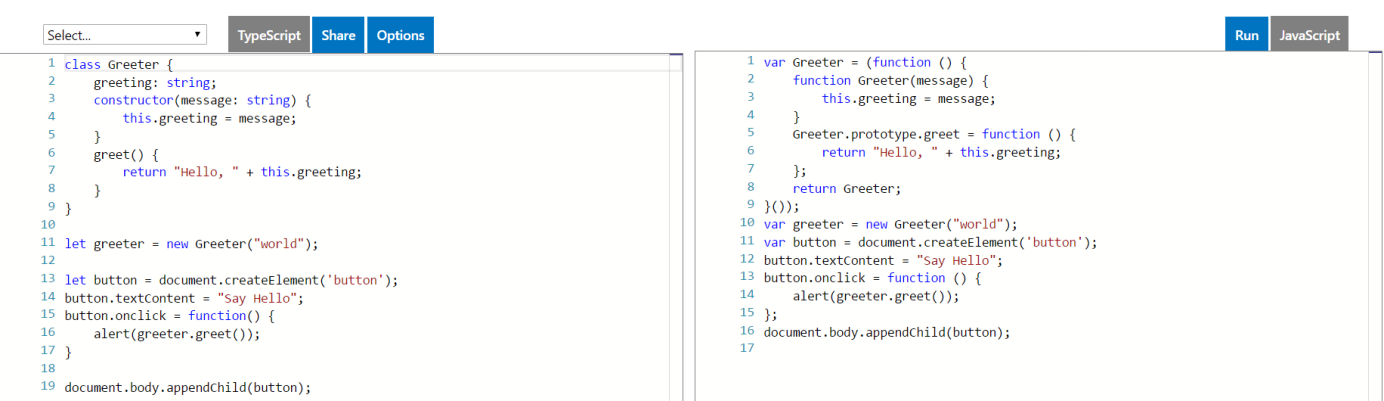
一.ts开发环境

Compiler 编译器 ts🡪js 原因 ES6 部分浏览器不支持

1.在线compiler编译

地址: <http://www.typescriptlang.org/>

选择playground



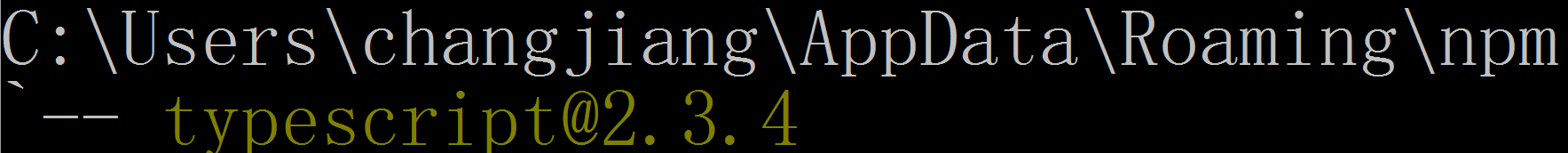
2.本地环境搭建

使用node环境下的npm安装

安装ts, 只涉及win平台 ,

安装步骤: npm install -g typescript (全局安装ts)

出现如下情况,ts安装完毕



3.简单测试

1)sublime新建文档

2)输入如下代码:

//ts创建类代码

export class hello{

}

保存在桌面,命名为hello.ts

3)通过命令行,编译该文件 tsc hello.ts

此时,会生成hello.js(编译后产生的代码,将会在浏览器执行的)

4.使用IDE工具

WebStorm

新建项目

新建ts文件, 输入文件名, 保存

开始书写ts代码

点击右上角OK编译,自动生成js文件

二.字符串新特性

1.多行字符串

//多行字符串使用连接

//js中

var content1 = "aaa" + "bbb" + "ccc";

//ts中 双撇号(数字1左边的键)

var content2 = `aaa

bbb

ccc`

编译后,js中显示

**var** content2 = "aaa\nbbb\nccc";

2.字符串模板

var myname = "changjiang";

var getname = function () {

return "changjiang";

}

console.log(`hello ${myname}`);

console.log(`hello ${getname()}`);

console.log(`<div>

<span>${myname}</span>

<span>${getname()}</span>

<div>xxx</div>

</div>`);

编译结果:

**var** myname = "changjiang";  
**var** getname = **function** () {  
 **return** "changjiang";  
}  
console.log(`hello ${myname}`);  
console.log(`hello ${getname()}`);  
console.log(`<div>  
<span>${myname}</span>  
<span>${getname()}</span>  
<div>xxx</div>  
</div>`);

3.自动拆分字符串

字符串模板作为函数参数,自动拆分变量

function test(template, name, age) {

console.log(template);

console.log(name);

console.log(age);

}

var myname = "chang jiang";

var getAge = function () {

return 18;

}

test`hello my name is ${myname},i'm ${getAge()}`

编译结果:

**function** test(template, name, age) {  
 console.log(template);  
 console.log(name);  
 console.log(age);  
}  
**var** myname = "chang jiang";  
**var** getAge = **function** () {  
 **return** 18;  
};  
(\_a = ["hello my name is ", ",i'm ", ""], \_a.raw = ["hello my name is ", ",i'm ", ""], test(\_a, myname, getAge()));  
**var** \_a;

三.参数

1.参数类型:在参数名称后面使用冒号来指定参数类型

var myname: string = "chang jiang";

myname = 15;//错误,报错,在js中不报错.

var abc = "xixi";

abc = 13;//错误,报错, 在第一次赋值时已经默认类型为字符串

var def: any = "haha";

def = 200; //此时不报错

var age: number = 20;//数值类型

var man: boolean = true;//布尔类型

//函数返回值定义类型

function test(): void{

return "abc";//因为上面制定了返回类型为void空, 此时返回字符串,报错;可以将上面改为string

}

//函数参数定义类型

function test1(name: string) {

}

test1("abc");

test1(12);//报错,与参数不匹配

//自定义类型

class Person{

name: string;

age: number;

}

var zhangsan: Person = new Person();

zhangsan.name = "张三";

zhangsan.age = 18;

2.默认参数:在参数声明后面用等号来指定参数的默认值

var myname: string = "chang jiang";

function test(a: string, b: string, c: string) {

console.log(a);

console.log(b);

console.log(c);

}

test("xxx", "yyy", "zzz");

//指定参数默认值

//带默认值的参数一定放在最后面

function test1(a: string, b: string, c: string ="abc" ) {

console.log(a);

console.log(b);

console.log(c);

}

test1("xxx", "yyy");

3.可选参数:在方法的参数声明后面用问号来表名此参数为可选参数

//可选参数

//b参数为可选状态

//必填写参数在可选参数前面

function test1(a: string, b?: string, c: string ="abc" ) {

console.log(a);

console.log(b);

console.log(b.length)//报错,可选参数没传时时,使用报错

console.log(c);

}

test1("xxx");

四.函数新特性

1.Rest and Spread操作符

用来声明任意数量的方法参数

//传递任意数量的参数

function test(...args) {

args.forEach(function (arg) {

console.log(arg);

});

}

test(1, 2, 3);

test(101, 102, 103, 104);

//传递固定数量参数

function test(a,b,c) {

console.log(a);

console.log(b);

console.log(c);

}

var args1 = [1, 2];

test(...args1);//此调用方法在ts中报错, 在ES6环境下没问题,编译出的js也没问题,缺少的参数显示undefined

var args2 = [7, 8, 9, 10, 11];

test(...args2);//只接受前面三个参数

2.generator函数

控制函数的执行过程, 手工暂停和恢复代码执行

function\*doSomething(){

console.log("start");

yield;

console.log("finish");

}

var func1 = doSomething();

func1.next();

func1.next();

function\* getStockPrice(stock){

while(true){

yield Math.random()\*100;

}

}

var priceGenerator = getStockPrice("IBM");

var limitPrice = 15;

var price = 100;

while(price>limitPrice){

price = priceGenerator.next().value;

console.log(`the generator return ${price}`);

}

3.destructuring析构表达式

通过表达式将对象或数组拆解成任意数量的变量

function getStock(){

return {

code: "IBM",

price: 100

}

}

//ES5里面

var stock = getStock();

var code = stock.code;

var price = stock.price;

//ES6

//此时的变量必须与方法中的一样

var { code, price } = getStock();

console.log(code);

console.log(price);

var { codex, price } = getStock();//报错

var { code: codex, price } = getStock();

多层对象中

function getStock(){

return {

code: "IBM",

price: {

price1: 200,

price2:300

},

aaa: "xixi",

bbb: "haha"

}

}

//ES6

//此时打印是对象

var { code, price } = getStock();//直接打印是对象

var { code, price:{price2} } = getStock();//打印出该值

console.log(code);

console.log(price2);

//及时内容多了, 也不影响析构表达式

数组中

var array = [1, 2, 3, 4];

var [number1, number2] = array;

console.log(number1);

console.log(number2);

var [, , number3, number4] = array;

console.log(number3);

console.log(number4);

var [number1, , ...others] = array;

console.log(others);

var array1 = [5, 6, 7, 8];

function doSomething([number5, , ...others1]) {

console.log(number5);

console.log(others1);

}

doSomething(array1);