

--	--

-
-
- Class
-



```
$ npm install --global typescript
```

TypeScript

.ts tsc

TS

--	--	--	--	--	--	--	--

JavaScript

```
# example.js
```

```
$ tsc example.ts
```



TypeScript 9

- Boolean
- Number
- String
- Array
- Tuple
- Enum
- Any
- Void
- Function



```
// 常量的类型
const lie : boolean = false;
const pi : number = 3.14159;
const tree_of_knowledge : string = "Yggdrasil";

// 常量的类型
const divine_lovers : string[] = ["Zeus", "Aphrodite"];
const digits : Array<number> = [143219876, 112347890];
```

Tuple `[...]` `[(Item Type), . . . , (Item Type)]`

```
let date_triplet : [number, number, number];
date_triplet = [31, 6, 2016];

let athena : [string, number];
athena = ['Athena', 9386];
var name : string = athena[0];
const age : number = athena[1];
```

Enums `[...]` Enum `[...]`

```
enum Color { Red, Green, Blue };
const red : Color = Color.Red;
console.log(Color[0]); // 'Red'

// TypeScript
enum RomanceLanguages { Spanish = 1, French, Italian, Romanian, Portuguese };
console.log(RomanceLanguages[4]); // 'Romanian'
console.log(RomanceLanguages[0]); // undefined
```

Any `[...]`

```
let mystery : any = 4; // number
mystery = "four"; // string -- no error

const not_only_strings : any[] = [];
not_only_strings.push("This Works!")
not_only_strings.push(42); // This does too.
```

Void `[...]` undefined `[...]` null `[...]`

```
let the_void : void = undefined;
the_void = null;

the_void = "nothing"; // Error
```

Function

```
function capitalizeName (name : string) : string {
    return name.toUpperCase();
}

let multiply : (first : number, second : number) => number;
multiply = function (first, second) {
    return first * second;
}

let multiplyFirst : ( first : number) => ((second : number) => number);
multiplyFirst = function (first) {
    return function (num) {
        return first * num;
    }
}
```

Class

TypeScript

```
class ListComponent {
    constructor () {
        this.things = [];
    }
}
```

TypeScript `this`

```
class ListComponent {
    things : Array<ListItem>; // Or -- things : ListItem[];
    constructor () {
        this.things = [];
    }
}
```

```
}  
}
```

□□□□□□□□

public □ private □□

```
class ListComponent {  
  public things : Array<ListItem>; // Or -- things : ListItem[];  
  constructor () {  
    this.things = [];  
  }  
}
```

```
class Person {  
  constructor (  
    public first_name : string,  
    public last_name : last_name  
  ) { }  
}
```

// □□□

```
class Person {  
  public first_name : string;  
  public last_name : string;  
  constructor (  
    first_name : string,  
    last_name : last_name  
  ) { }  
}
```

private □□□□

```
class ListComponent {  
  private _things : Array<ListItem>;  
  constructor () {  
    this._things = [];  
  }  
  get things () : Array<ListItem> { return this._things; }  
}
```

TypeScript□□□□ ES6□□□□□□□□

```

class CompletedListItem extends ListItem {
    completed : boolean;
    constructor (name : string) {
        super(name);
        this.completed = true;
    }
}

```

Interface `interface` Class `class`

```

interface User {
    name : string;
    email : string;
}

class RegisteredUser implements User {
    constructor (
        public name : string,
        public email : string
    ) { }
}

```

`interface`

```

interface User {
    name : string;
    email : string;
    avatar? : Object;
}

```

```

// 接口定义
interface User {
    // PROPERTIES
    name : string;
    email : string;
    avatar? : Object;

    // API
    print () : void;
}

```

```
}
```



- Peleke Sengstacke, [From JavaScript to TypeScript, Pt. IIA: Using Classes, Interfaces, & Mixins](#)

☐☐ #1

☐ ☐☐ ☐☐ 2017-04-09 10:10:41 UTC

☐ ☐☐ ☐☐ 2017-04-09 10:10:41 UTC