



FPT POLYTECHNIC



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

LẬP TRÌNH TYPESCRIPT

- ◎ 1.1 Arrow function
- ◎ 1.2 Function return
- ◎ 1.3 Function as types
- ◎ 1.4 Function with parameters
 - ❖ 1.4.1 Default parameter
 - ❖ 1.4.2 Optional parameter
 - ❖ 1.4.3 Spread operators
 - ❖ 1.4.4 Rest parameter
- ◎ 1.5 Function & void
- ◎ 1.6 Never & void



- ② 2.1 Watch mode
- ② 2.2 Compiling the entire project multiple files
- ② 2.3 tsconfig.json
 - ❖ 2.3.1 Including & excluding files
 - ❖ 2.3.2 Target & lib
 - ❖ 2.3.3 More configuration & source map
 - ❖ 2.3.4 rootDir and outDir





PHẦN 1

NEXT-GENERATION JAVASCRIPT & TYPESCRIPT

prefix Tên hàm

```
function f_name ()  
{  
    /*statements*/  
}
```

```
let f_name = () =>  
{  
    /*statements*/  
}
```

Arrow notation

ARROW FUNCTION RETURN

```
function f_name ()  
{  
    return;  
}
```



Lệnh return

```
let f_name = () =>  
{  
    return;  
}
```



Arrow notation

ARROW FUNCTION AS TYPE

```
function Sum () : number
{
    return;
}
```

Kiểu của giá trị
trả về

```
let Sum = () : number =>
{
    return;
}
```

Arrow notation

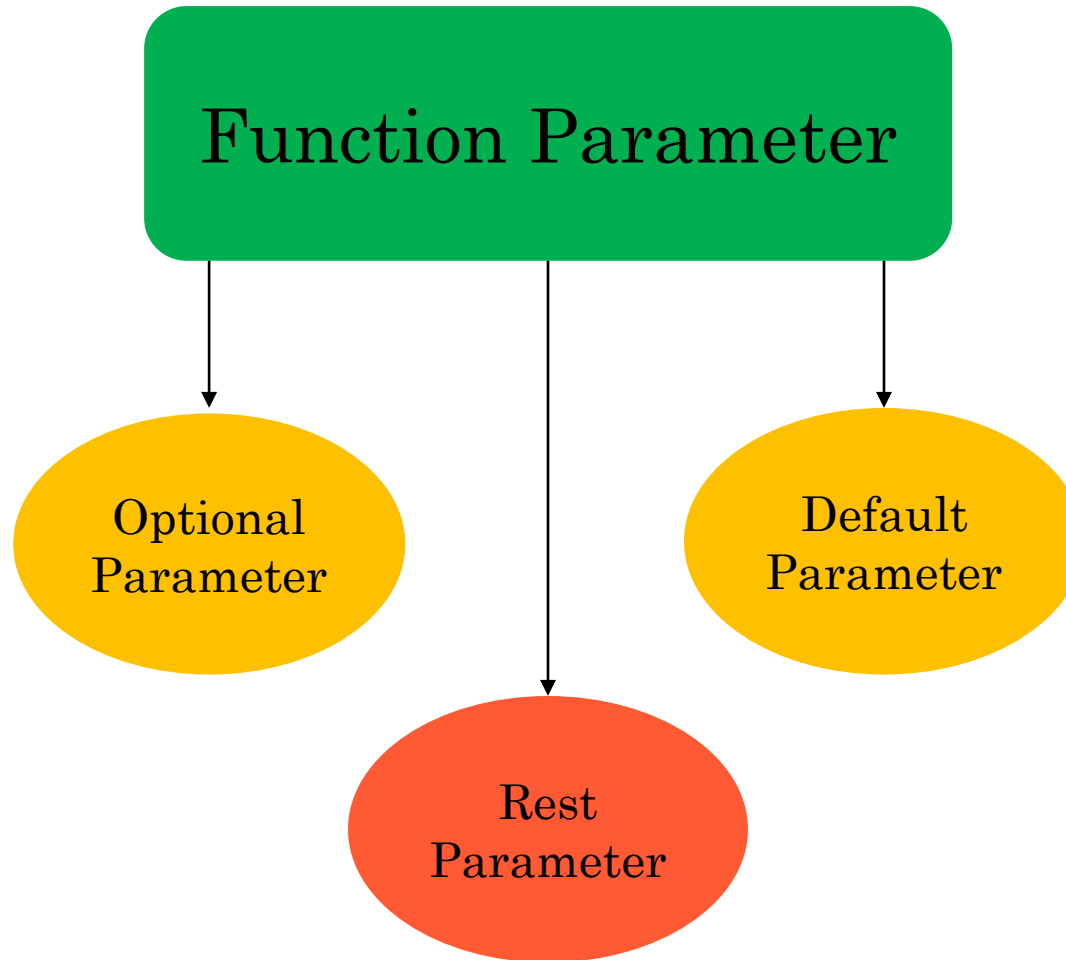
TS app.ts



SimpleProject > TS app.ts > ...

```
1  function Sum(): string {
2      return "Result: 5";
3  }
4  let showSum; //type?
5  showSum = Sum;
6  console.log(showSum());
7
8  let greeting = (): number => {
9      return 10;
10 }
11 console.log("Result: " + greeting);
12
13
14
```


FUNCTION WITH PARAMETERS



```
function f-name (para1[:type], para2[:type], ...)  
{  
    return;  
}
```

```
let f-name = (para1[:type], para2[:type], ...) =>  
{  
    return;  
}
```



Arrow notation

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  const sum = (x:number, y:number) => x+y;
2  const printOutput = (output: string|number) => console.log(output);
3
4  printOutput(sum(5,6));
```

5

6

7

8

9

10

11

12

13

14

15

16

17

18

□ Cú pháp

```
let f-name = (para1[:type],  
para2[:type]=default-value,...) =>  
{  
    return;  
}
```

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  let sum = (x:number=5, y:number) => x+y;
2  const printOutput = (output: string|number) => console.log("Result:" + output);
3
4
5  printOutput(sum(3));
6  printOutput(sum(undefined,5));
7  printOutput(sum(3,5));
8
9
10
11
12
```

❑ Sử dụng **?** sau parameter name

```
let f-name = (para1[:type],  
para2?:[:type],...) =>  
{  
    return;  
}
```

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  let sum = (x:number=5, y?:number) => { return x + <number>y; }
2  const printOutput = (output: string|number) => console.log("Result:" + output);
3
4
5  printOutput(sum(3));
6  printOutput(sum(undefined,5));
7  printOutput(sum(3,5));
8
9
10
11
12
13
```

- ❑ Merging object (trộn objects)
- ❑ Merging arrays, Copying arrays (trộn mảng hoặc sao chép mảng)
- ❑ Ký hiệu



...

MERGING OBJECT WITH SPREAD OPERATOR

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  //MERGING OBJECT
2  let person : {
3      name: string,
4      age: number
5  } = {
6      name: 'Typescript',
7      age: 11
8  }
9  const salary : {
10     grade: string,
11     basic: string
12 } = {
13     grade: 'A',
14     basic: '$2900'
15 }
16
17 const summary = {...person, ...salary};
18 console.log(summary);
19
```

MERGING ARRAY WITH SPREAD OPERATOR

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  //ARRAY
2  const hobbies = ['Sports', 'Cooking'];
3  const activehobbies = ['Hiking'];
4  activehobbies.push(hobbies);
5  activehobbies.push(hobbies[0], hobbies[1]);
6  activehobbies.push(...hobbies);
7  console.log(activehobbies);
8
9
10
11
12
```

Chỉ có 1 rest
parameter trong
function

Có kiểu là
array

Phải là
parameter cuối
cùng trong danh
sách parameter

TS app.ts X

SimpleProject > TS app.ts > ...

```

1  let addInputValues = function( ...values: number[], output:string, ): string {
2      let result = 0;
3      for (let val of values) {
4          result += val;
5      }
6      return output+":"+result;
7  };|
8
9  let printOutput = (output: string) => console.log(output);
10
11 printOutput(addInputValues("Hello! We have");); //OK - You can choose not to pass anything as well
12 printOutput(addInputValues("Hello! We have", 1, 1)); //OK
13 printOutput(addInputValues("Hello! We have", 1, 2, 3)); //OK
14 printOutput(addInputValues("Hello! We have", 1, 2, 3, 4, 5, 6)); //OK
15
16

```

TS app.ts ×

SimpleProject > TS app.ts > [🔍] sum

```
1 let sum = (x:number=5, y?:number) => { return x + <number>y; }
2 let speech = (output: any): void => {
3   console.log("Result:" + output);
4 }
5 speech(sum(5,12));
6 console.log(speech(sum(8,5)));
7
8
9
```

`void` được sử dụng khi không có dữ liệu

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1  let something: void = undefined;
2  let nothing: never = null; // Error: Type 'null' is not assignable to type 'never'
3  function throwError(errorMsg: string): never {
4      |   throw new Error(errorMsg);
5  }
6
7
8
9
```

FUNCTION & CALLBACK

TS app.ts ×

SimpleProject > TS app.ts > ...

```
1 function AddandHandle(x: number, y: number, cb:(num:number)=>void){  
2     const result = x + y;  
3     cb(result);  
4 }  
5 AddandHandle(10, 20, (result) => { console.log(result); })  
6  
7  
8  
9
```


Callback
function

demo



PHẦN 2

THE TYPESCRIPT COMPILER



The screenshot shows a VS Code editor with a file named `app.ts`. The code contains two lines: `let str: string = "Welcome to this course";` and `console.log(str);`. Below the code, two blue boxes highlight the commands `tsc -w` and `tsc --watch`. At the bottom, the terminal window shows the output of running these commands: `[1:19:14 PM] Starting compilation in watch mode...` and `[1:19:16 PM] Found 0 errors. Watching for file changes.`

```
TS app.ts  ×
TS app.ts > ...
1  let str: string = "Welcome to this course";
2
3  console.log(str);
```

`tsc -w`

`tsc --watch`

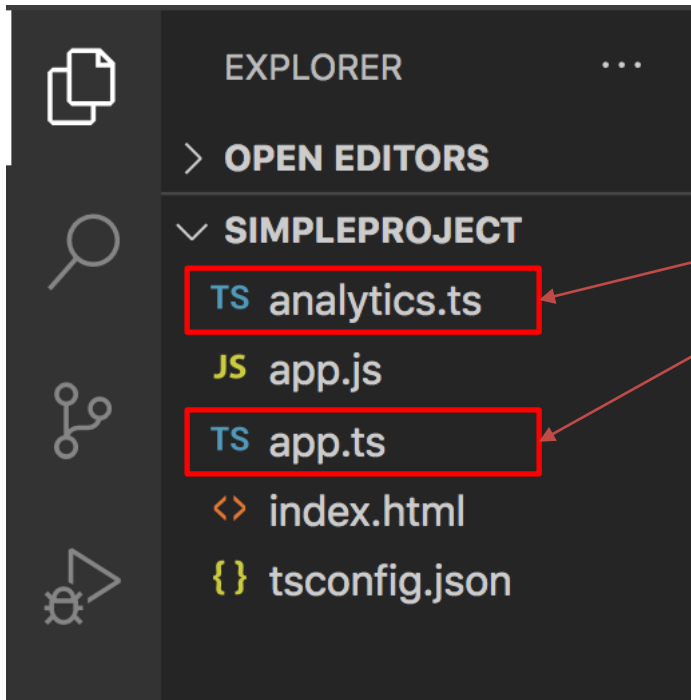
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

[1:19:14 PM] Starting compilation in watch mode...

[1:19:16 PM] Found 0 errors. Watching for file changes.



KHÔNG THOÁT chế độ Watch trong khi phát triển chương trình.
Chỉ thoát chế độ Watch sau đó thông qua lệnh **CTRL+C**



files typescript

- ❑ Tạo file tsconfig.json

```
tsc -init
```

- ❑ Biên dịch các tập tin typescript

```
tsc app.ts analytics.ts
```

Hoặc

```
tsc
```

❑ Include file js vào file html

```
<script src="app.js" defer></script>  
  
<script src="analytics.js"  
defer></script>
```

- ◎ 2.1 Watch mode
- ◎ 2.2 Compiling the entire project multiple files
- ◎ 2.3 tsconfig.json
 - ❖ 2.3.1 Including & excluding files
 - ❖ 2.3.2 Target & lib
 - ❖ 2.3.3 More configuration & source map
 - ❖ 2.3.4 rootDir and outDir



❑ 2.3.1 Include và exclude

Mảng các tập tin hoặc các mẫu (thư mục chứa tập tin, phần mở rộng của tập tin) được phép đưa vào/bỏ ra trong chương trình

```
{  
  "include": ["src/**/*", "tests/**/*"]  
}
```

```
{  
  "exclude": ["scripts/**/*"]  
}
```

□ Ví dụ

```

.
├── scripts
│   ├── lint.ts
│   ├── update_deps.ts
│   └── utils.ts
├── src
│   ├── client
│   │   ├── index.ts
│   │   └── utils.ts
│   └── server
│       └── index.ts
├── tests
│   ├── app.test.ts
│   ├── utils.ts
│   └── tests.d.ts
├── package.json
├── tsconfig.json
└── yarn.lock
  
```

❑ 2.3.2 Target và lib

- ❑ **Target:** phiên bản của ngôn ngữ được sử dụng cho output sau khi compile
- ❑ **Lib:** cho phép chỉ định những objects và features(tính năng) mặc định mà typescript ghi chú

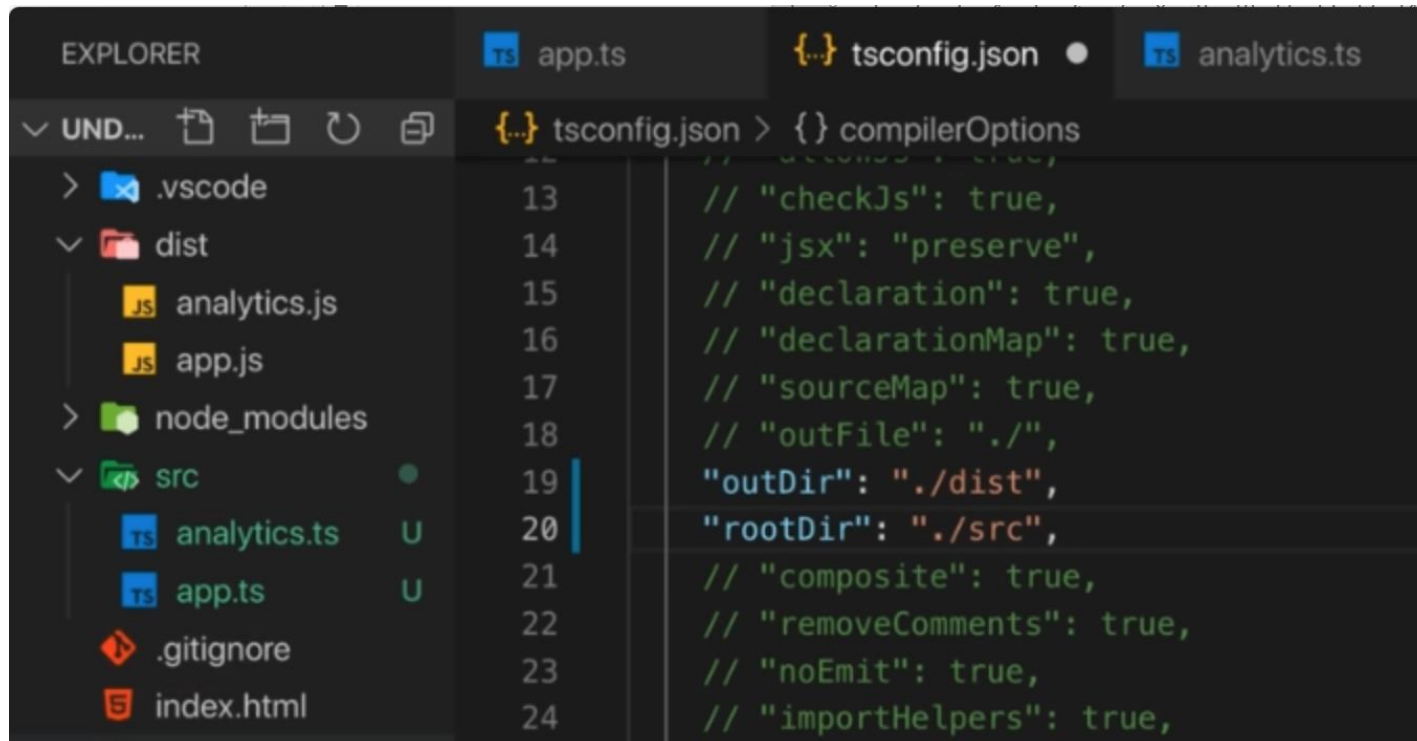
```
"target": "es5",  
"module": "commonjs",  
"lib": [  
  "dom",  
  "es6",  
  "DOM.Iterable",  
  "scripthost"],
```


❑ 2.3.3 More configuration & source map

- ❑ **AllowJs:** cho phép chấp nhận tập tin js như là đầu vào của tập tin ts
- ❑ **CheckJs:** hoạt động song song với AllowJs, khi CheckJs được bật thì lỗi trong tập tin js sẽ được báo nếu có.
- ❑ **Declaration:** tạo tập tin có phần mở rộng là .d.ts cho các tập tin .ts hoặc .js
- ❑ **Sourcemap:**
 - ❖ Tạo ra dạng tập tin .js.map bên cạnh tập tin output .js
 - ❖ Hiển thị tập tin nguồn .ts của tập tin output .js và cho phép debug lỗi trên tập tin nguồn đó.
 - ❖ Trên tập tin .js có chú thích vị trí của tập tin source map

❑ 2.3.4 rootDir & outDir

- ❑ **rootDir**: đường dẫn của các tập tin đầu vào không khai báo
- ❑ **outDir**: đường dẫn chứa các tập tin output



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with folders like .vscode, dist, node_modules, and src. The src folder is expanded, showing files analytics.ts, app.ts, .gitignore, and index.html. The main editor area shows the tsconfig.json file, which is part of the compilerOptions section. The configuration includes settings for checkJs, jsx, declaration, declarationMap, sourceMap, outFile, outDir, rootDir, composite, removeComments, noEmit, and importHelpers. The outDir is set to './dist' and the rootDir is set to './src'.

```
tsconfig.json > { } compilerOptions
13 // "checkJs": true,
14 // "jsx": "preserve",
15 // "declaration": true,
16 // "declarationMap": true,
17 // "sourceMap": true,
18 // "outFile": "./",
19 "outDir": "./dist",
20 "rootDir": "./src",
21 // "composite": true,
22 // "removeComments": true,
23 // "noEmit": true,
24 // "importHelpers": true,
```

demo

- ☑ 1.1 Arrow function
- ☑ 1.2 Function return
- ☑ 1.3 Function as types
- ☑ 1.4 Function with parameters
 - ❖ 1.4.1 Default parameter
 - ❖ 1.4.2 Optional parameter
 - ❖ 1.4.3 Spread operators
 - ❖ 1.4.4 Rest parameter
- ☑ 1.5 Function & void
- ☑ 1.6 Never & void



- ☑ 2.1 Watch mode
- ☑ 2.2 Compiling the entire project multiple files
- ☑ 2.3 tsconfig.json
 - ❖ 2.3.1 Including & excluding files
 - ❖ 2.3.2 Target & lib
 - ❖ 2.3.3 More configuration & source map
 - ❖ 2.3.4 rootDir and outDir



thank
you!