✓ 100 XP

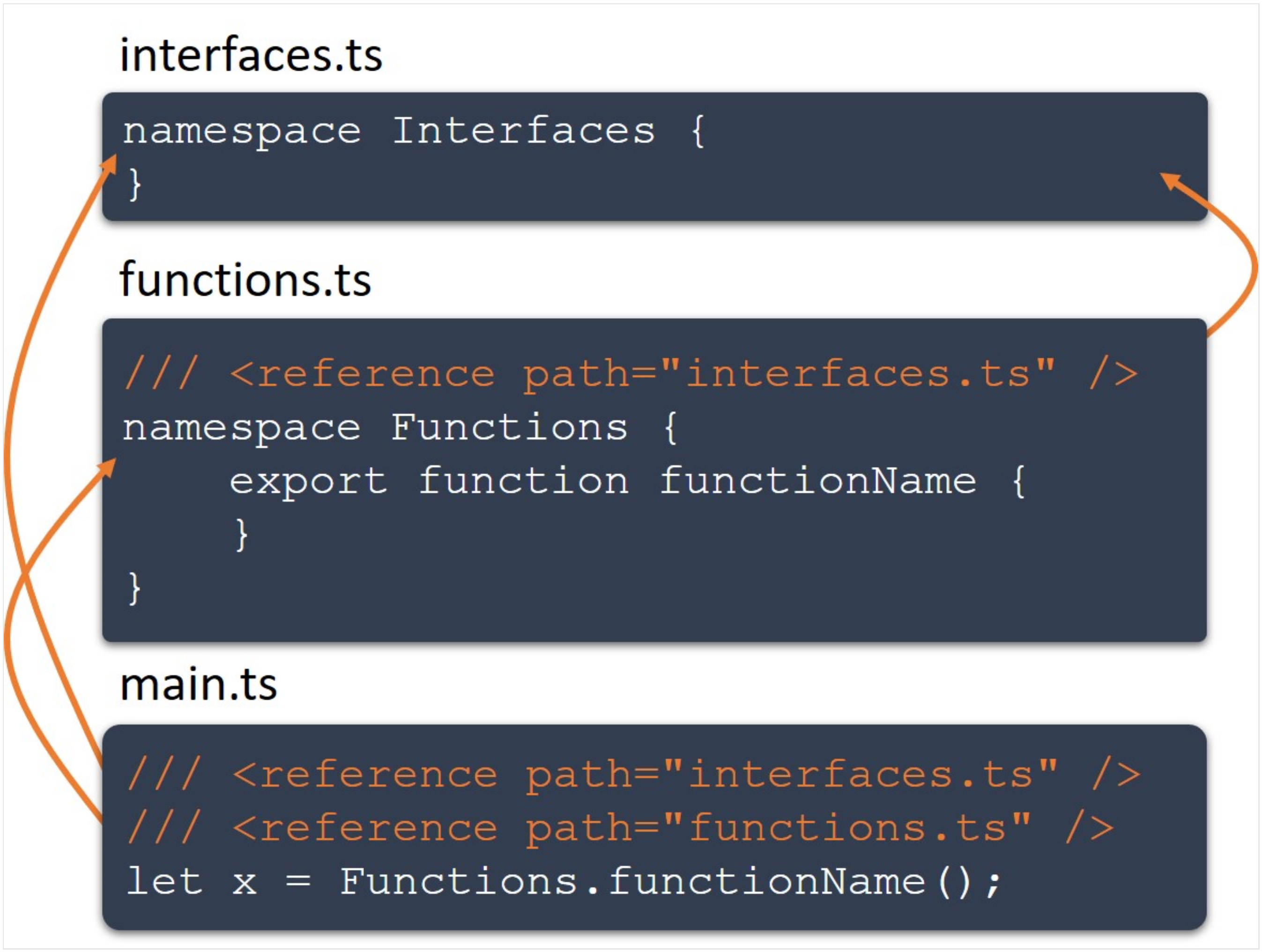
Organize code by using multi-file namespaces

5 minutes

You can extend namespaces by sharing them across multiple TypeScript files. When you have namespaces in multiple files that relate to each other, you must add `reference` tags to tell the TypeScript compiler about the relationships between the files. For example, assume that you have three Typescript files:

- **interfaces.ts**, which declares a namespace that contains some interface definitions.
- **functions.ts**, which declares a namespace with functions that implement the interfaces in **interfaces.ts**.
- **main.ts**, which calls the functions in **functions.ts** and represents the main code of your application.

To inform TypeScript of the relationship between **interfaces.ts** and **functions.ts**, you add a `reference` to **interfaces.ts** using the triple slash (`///`) syntax to the top of **functions.ts**. And then in **main.ts**, which has a relationship with both **interfaces.ts** and **functions.ts**, you add a `reference` to both files.



```
namespace Interfaces {
}
```

```
/// <reference path="interfaces.ts" />
namespace Functions {
    export function functionName {
    }
}
```

```
/// <reference path="interfaces.ts" />
/// <reference path="functions.ts" />
let x = Functions.functionName();
```

When there is a reference to more than one file, start with the highest-level namespace and then work your way down. TypeScript will use this order when compiling the files.

Compiling multi-file namespaces

There are two ways to compile multiple file namespaces: per-file compilation and single file compilation.

By default, when you run the TypeScript compiler on **main.ts**, it will examine the `reference` statements in the file and produce one JavaScript file for each input file. If you choose this option, use `<script>` tags on the webpage to load each emitted file in the appropriate order.

You can also instruct the compiler to produce a single JavaScript output file by using the `--outFile` option. In the example above, the command `tsc --outFile main.js main.ts` instructs the compiler to produce a single JavaScript file called **main.js**.

Next unit: Design considerations

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆