САМАРСКИЙ УНИВЕРСИТЕТ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Лекция № 4

Модули и пространства имён в TypeScript

2023

РУБРИКА: «ВОПРОСЫ ПО ЛЕКЦИИ № 3»

Пространство имён

Что такое Пространство имён?

это конкретно обозначенная область исходного кода в программе, указывающая, что содержимое его является сопряженным по отношению друг к другу.
Пространства имен содержат группу классов, интерфейсов, функций, других пространств имен, которые могут использоваться в некотором общем контексте

Зачем это нужно?

Для разграничения больших частей ПО, их именования и указания на целостность кода программы в данном контексте.
Примером может послужить главы книги, далее идут их части (но далеко необязательно) и так далее (вплоть до букв).
В общем контексте можно это назвать это содержимым оглавления ПО.

ОГЛАВЛЕНИЕ

Пространство имён :: Пример

Для определения пространств имен
используется ключевое слово namespace

```
namespace Personnel {
    export class Employee {

        constructor(public name: string){
        }
    }
}
```

```
namespace Personnel {
    export class Employee {

        constructor(public name: string){
        }
    }
}

let alice = new Personnel.Employee("Alice");
console.log(alice.name);    // Alice
```

Пространство имён :: Пример дополненный

```
namespace Personnel {

    export interface IUser{
        displayInfo(): void;
    }

    export class Employee {
        constructor(public name: string){
        }
    }

    export function work(emp: Employee) : void{
        console.log(emp.name, "is working");
    }

    export let defaultUser= { name: "Kate" }

    export let value = "Hello";
}

let tom = new Personnel.Employee("Tom")
Personnel.work(tom);                    // Tom is working

console.log(Personnel.defaultUser.name);    // Kate
console.log(Personnel.value);    // Hello
```

Пространство имён :: Отдельный файл

personnel.ts

```
namespace Personnel {
    export class Employee {

        constructor(public name: string){
        }
    }
    export class Manager {

        constructor(public name: string){
        }
    }
}
```

app.ts

```
/// <reference path="personnel.ts" />

let tom = new Personnel.Employee("Tom")
console.log(tom.name);

let sam = new Personnel.Manager("Sam");
console.log(sam.name);
```

С помощью директивы /// <reference path="personnel.ts" /> подключается файл personnel.ts.

Пространство имён :: Вложенные пространства имён

```
namespace Data{
    export namespace Personnel {
        export class Employee {

            constructor(public name: string){
            }
        }
    }
    export namespace Clients {
        export class VipClient {

            constructor(public name: string){
            }
        }
    }
}

let tom = new Data.Personnel.Employee("Tom")
console.log(tom.name);

let sam = new Data.Clients.VipClient("Sam");
console.log(sam.name);
```

Вложенные пространства имен определяются со словом export. Соответственно при обращении к типам надо использовать все пространства имен

Пространство имён :: Псевдонимы

```
namespace Data{
   export namespace Personnel {
      export class Employee {

         constructor(public name: string){
         }
      }
   }
}

import employee = Data.Personnel.Employee;
let tom = new Employee("Tom")
console.log(tom.name);
```

Строка, помеченная желтым заявляет компилятору, что хочет выполнить именование класса Employee в иерархии пространства имён Data.Personnel

Модули

Что такое Модуль?

это конкретно обозначенная область исходного кода в программе, указывающая, что содержимое его является сопряженным по отношению друг к другу.
Пространства имен содержат группу классов, интерфейсов, функций, других пространств имен, которые могут использоваться в некотором общем контексте

Зачем это нужно?

Для разграничения больших частей ПО, их именования и указания на целостность кода программы в данном контексте.
Примером может послужить главы книги, далее идут их части (но далеко необязательно) и так далее (вплоть до букв).
В общем контексте можно это назвать это содержимым оглавления ПО.

ОГЛАВЛЕНИЕ

Что тогда не модуль?

Все, что не помечено в рамках кода ключевыми словами «export» или «await»

Можно ли сделать модуль по умолчанию?

Да, после ключевого слово «export» нужно всего лишь поставить ключевое слово «default»

Импортирование модулей

Для того, чтобы применить модуль в рамках скрипта, необходимо выполнить его импорт в файл, в котором будет происходить использование элемента модуля. Для этого необходимо использовать ключевое слово «import»

- AMD (Asynchronys Module Definition)
- CommonJS (используется по умолчанию, если параметр --target равен "ES3" или "ES5")
- UMD (Universal Module Definition)
- System
- ES 2015
- ES 2020
- ESNext

Какой из типов мы будем использовать в рамках выполнения практических работ?

- ES 2015
- ES 2020

- AMD (Asynchronys Module Definition)
- CommonJS (используется по умолчанию, если параметр --target равен "ES3" или "ES5")
- UMD (Universal Module Definition)
- System
- ES 2015
- ES 2020
- ESNext

Какой из типов мы будем использовать в рамках выполнения практических работ?

- ES 2015
- ES 2020

Определение модуля?

```
export default function hello(name: string) {
   console.log("Hello, “ + name);
}
```

Импорт модуля?

Да, после ключевого слово «export» нужно всего лишь поставить ключевое слово «default»

```
import hello from "./hello.js";

hello();
```

Компиляция модулей

Для компиляции модулей можно использовать следующие команды, вводимые в командную строку

```
tsc --module commonjs main.ts
tsc --module amd main.ts
tsc --module umd main.ts
tsc --module system main.ts
tsc --module esnext main.ts
```

Применение компилированных файлов

Используем как самый обычный импорт JS-файлов в документ

```
<!doctype HTML>
<html>
  <head>
    ...
  </head>
  <body>
    ...
    <script type="module" src="main.js"></script>
  </body>
</html>
```

РУБРИКА: «ВОПРОСЫ СЛУШАТЕЛЕЙ»

КОНЕЦ ЛЕКЦИИ № 4
СПАСИБО ЗА ВНИМАНИЕ