Modules and Namespaces

№ урока: 4 **Курс:** TypeScript Fundamentals

Средства обучения: Visual Studio, Visual Studio Code, NotePad++

Обзор, цель и назначение урока

Цель урока – ознакомление студентов с пространством имен. Изучить применение модулей. Ознакомить учащихся с базовыми параметрами настройки **Webpack**.

Изучив материал данного занятия, учащийся сможет:

- Использовать работу с модулями;
- Применять функционал пространства имен;
- Использовать различные сборщики модулей;
- Применять в своих приложениях сборщик **Webpack**.

Содержание урока

1. Использование пространств имен;
2. Работа с модулями;
3. Настройка **Webpack**.

Резюме

- Для организации больших программ предназначены пространства имен. Пространства имен содержат группу классов, интерфейсов, функций, других пространств имен, которые могут использоваться в некотором общем контексте.
- Для определения пространств имен используется ключевое слово **namespace**. Чтобы типы и объекты, определенные в пространстве имен, были видны извне, они определяются с ключевым словом **export**. Пространства имен могут содержать и интерфейсы, и объекты, и функции.
- С помощью **директивы /// <reference path="personnel.ts" />** подключается файл **personnel.ts**.
- **TypeScript** поддерживает работу с модулями. Модули являются концепцией, привнесенной стандартом **ES2015**, однако в современных браузерах нативная поддержка модулей еще не реализована. Модули в некотором смысле похожи на пространства имен: они могут заключать различные классы, интерфейсы, функции, объекты. Модули выделяются в отдельные файлы, что позволяет сделать код приложения более ясным и чистым, и в то же время позволяет использовать модули в других приложениях. При этом модули подключаются в приложение не посредством тега <**script**>, а с помощью загрузчика модулей.
- Все модули имеют определенный формат и относятся к определенной системе. Всего есть 5 различных систем модулей: **AMD** (**Asynchronys Module Defenition**), **CommonJS, UMD** (**Universal Module Defenition**), **System**, **ES 2015**.
- А для загрузки модулей применяются специальные загрузчики:
- **RequireJS**: **RequireJS** использует синтаксис, известный как асинхронное определение модуля или **asynchronous module definition**(**AMD**);
- **Browserify**: использует синтаксис **CommonJS**;
- **SystemJS**: универсальный загрузчик, может применяться для модулей любого типа.
- Чтобы классы, интерфейсы, функции были видны извне, они определяются ключевым словом **export**. Чтобы задействовать модуль в приложении, его надо импортировать с помощью оператора **import**.
- **Webpack** представляет популярный инструмент для сборки модулей в один файл. Для настройки используется файл **webpack.config.js**, который будет содержать конфигурацию **Webpack** с основными параметрами:

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **entry**: определяет входные файлы для создания сборок;
- **output**: определяет конфигурацию выходных файлов;
- **resolve**: определяет, как будут обрабатываться файлы, если они не имеют расширений;
- **module**.**rules**: определяет загрузчики, которые загружают модули;
- **plugins**: определяет применяемые плагины.

Закрепление материала

- Что такое **triple slash reference**?
- Как импортировать модуль по умолчанию?
- Как экспортировать все элементы файла?
- Что такое **Webpack**?
- Как установить **Webpack** локально в проект?

Дополнительное задание

Установите **Webpack** локально себе в проект. Настройте все параметры для компиляции готового файла **main.js** в верхнюю директорию **final**. Установите для **Webpack** загрузочный сервер с портом **8800**.

Самостоятельная деятельность учащегося

Задание 1
Выучить основные понятия, рассмотренные на уроке.

Задание 2
Используя импорты **Typescript,** создайте следующее приложение из 2 урока (с абстрактными классами). В первом файле будут определены интерфейсы для описания, во втором – классы (родительский и производные). 3 файл – экземпляры классов, построенные на шаблонах интерфейсов. Используйте любой загрузчик модулей.

Рекомендуемые ресурсы

https://www.typescriptlang.org/

https://www.typescriptlang.org/play/index.html

https://github.com/Microsoft/TypeScript

https://webpack.js.org/

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com