

Node.js 11 HTTPS/Express 4.16/PUG Using TypeScript

Web UI app, REST API app, Node HTTPS, Express, Generator, Routing, PUG, Project

Because of its ease of development, similarity with web client-side programming, maturity along with its expanding framework and tooling story, a Node-based approach to server-side development is very compelling. Since TypeScript is usually used client-side, it make sense to also use it server-side.

This course explores using TypeScript to develop server-side web UI and REST API applications using the latest technologies based on Node.js 11. It covers a mix of technologies that together allow developers to quickly build robust server-wide web solutions.

The Node.js HTTP(S)&HTTP/2 modules are how Node apps talk to the HTTP protocol family. The Node-based Express Application Framework is how rich web UI and web API apps can be built. It exposes powerful routing and middleware capabilities. The PUG template engine (the latest version of what used to be called Jade), transforms an HTML-like template syntax plus supplied data values into HTML which is sent to the web browser for rendering. In some ways PUG competes with Angular on the client – we contrast what they offer and investigate how to use them together in the same solution (often a sensible approach).

Contents of One-Day Training Course		
<p>Target Audience Web developers seeking to build rich web UI and REST API apps using the latest Node runtime, Express application framework and PUG template engine.</p> <p>Prerequisites Attendance at our <i>Node.js 11 Runtime Programming Using TypeScript</i> course or equivalent experience.</p> <p>All demos and lab exercises will be in TypeScript.</p>	<p>Building Web UI apps Review of server-side and client-side UI development options Role of templating and access to data Creating compelling user experiences in a Node world</p> <p>REST API Apps Review of REST API concepts Designing a REST API using Node Evolving a REST API Security issues</p> <p>Node.js 11 HTTP[S] Modules HTTP message flows using Node Use of createServer to create server server.listen and server.on syntax HTTP message content – headers, etc. Error handling considerations (error.code and Exxx values)</p> <p>Introduction to Express (4.16) Application framework based on Node Main components: express(), Application, Request, Response and Router Intro to what Express calls middleware</p> <p>Express Generator Recommended file layout for Node/Express apps Express-Generator constructs boilerplate code and layout quickly Reviewing generated content</p> <p>Express Routing Converting incoming URLs and query syntax to action invocations Defining and configuring a router</p>	<p>Implementing an API Testing an API</p> <p>Express Middleware Additional custom steps added to processing pipeline Stack of middleware functions Helpful auxiliary packages such as path, morgan and bodyParser</p> <p>PUG Template engine based on Node & Express Intro to PUG Template syntax Intro to PUG framework - runs on server and transforms syntax + data into HTML</p> <p>PUG Template Syntax Concise representation of markup Consumes data to generate contents Loops and control flow Useful shorthand for common needs</p> <p>PUG API Usage pug.compile[[File Client FileClient ClientWithDependenciesTracked] pug.render and pug.renderFile Options interface</p> <p>Angular Client-side UI Server-side UI vs. Client side Why server-wide UI is important (e.g. efficient access to server-side data) even in a world with Angular Integrating Angular on the client with server-side coding</p> <p>Project Developing an integrated project that brings together all the topics covered in the course</p>