JAVASCRIPT FUNDAMENTALS (JS-202)

LEVEL: INTERMEDIATE

OVERVIEW

5 days for seasoned programmers | 7 days for freshers
Considering ~7.25 hours/day (excluding time for lunch break and 2 tea breaks)

JavaScript (JS) is the language for scripting web pages – to enable user interactions on a web page, communicate with the backend etc.

Of late, JS has gained considerable popularity. A language, once solely in the domain of front-end web development, it has now expanded horizons to include applications in the server-side, mobile, desktops, IoT etc.

The latest versions of JavaScript like ES2015 (ES6), ES2016 (ES7) have introduced a plethora of great new features that have found adoption in modern frontend and backend frameworks. This bootcamp helps people with zero knowledge of JavaScript master basic and intermediate level features.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

The Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world. The Node.js built-in modules along with the plethora of third-party modules and frameworks make it a good choice to quickly create backend web applications.

The Node JS part of the course outlines use of MongoDB / MySQL / PostgreSQL with Mongoose/ Sequelize package.

The choice of database needs to be made before the training starts.

PREREQUISITES

- Working knowledge of HTML and CSS
- Sound knowledge of programming

APPLICATION BUILT DURING TRAINING

Add the end of this bootcamp, participants will build a product catalog application. They shall be provided a backend server. The application will involve communicating with the backend and listing products, adding, editing and removing products, posting product reviews etc.

© Prashanth Puranik
+91-9448441478
puranik@praharaconsulting.com
puranik@digdeeper.in

LIST OF SOFTWARE TO BE INSTALLED BEFORE TRAINING BEGINS

1. Git CLI on participant systems and GitHub account should be created for every participant (to be created individually by participant). The **GitHub account should be a personal one** and not one associated with the company's GitHub account (I will not be able to add a company account as collaborator on my repositories, and hence shall not be able to share code).

   Git CLI download: https://git-scm.com/downloads

   GitHub link for account creation: https://github.com/join?source=header-home

2. Node.js needs to be installed on all systems – Mac OSX, Linux and Windows is supported. The 10.x.x (LTS version) may be installed. This will also install npm. However, the proxy server details may need to be configured to enable npm access the npm registry (this registry is required to download Node modules required to build Node applications) – **the proxy configuration is a necessary step and has to be completed before training starts**.

   Node.js https://nodejs.org/en/download/

3. Download and install Visual Studio Code (VSCode)

   https://code.visualstudio.com/download

   It is available for Windows, Mac OSX and popular Linux distributions.

4. Depending on choice of database for the training, **only one of MongoDB or MySQL needs to be installed**.

   a) **MongoDB** needs to be installed on all systems. Instructions to download and install on various platforms can be found in the links below.

   https://www.mongodb.com/try/download/community

   https://docs.mongodb.com/manual/administration/install-community/

   Additionally, a \data\db (on root drive where MongoDB is installed on Windows, say C:\ or D:\) or /data/db folder (on Linux / Mac OSX) is required to be created. Participants should have write permissions on this folder.

   Next download and install **Robo 3T** - https://robomongo.org/download

   b) **MySQL** – Download and install from https://dev.mysql.com/downloads/mysql/

   Next download and install **MySQL Workbench** from
   https://dev.mysql.com/downloads/workbench/

5. For browser – latest version of one of Chrome or Firefox, **preferably Chrome**. Internet Explorer is not acceptable.

   Chrome: https://www.google.com/chrome/browser/desktop/index.html

   Firefox: https://www.mozilla.org/en-US/firefox/new/

6. **Additionally, it would be great if participants have as little restrictions (as permissible) on internet access during the session**

© Prashanth Puranik
+91-9448441478
puranik@praharaconsulting.com
puranik@digdeeper.in

CHAPTERS AND TOPICS

Overview

Brief history of ECMAScript (JS language specification)

Features introduced in various versions

Where and how JavaScript is used

Is JavaScript interpreted or compiled?

Before getting started

Inclusion and execution in an HTML context – inline JS, script tag (in document and external script)

What happens in case of errors?

Strict Mode Execution

Comments – single and multi-line

Case-sensitivity, automatic semi-colon insertion

Identifiers, Variables and Data Types

Rules for Identifiers (variable, function names etc.)

Variable declaration

Primitive Data Types – number, boolean, string

Two special primitive types/values - null and undefined

Arrays, multi-dimensional arrays

Array-like objects

Operators, expressions and control flow

Operators (arithmetic, relational, logical)

How === differs from ==

Copy by value for primitives vs copy by reference for non-primitives

Expressions and operator precedence

Miscellaneous operators – conditional (?..:), typeof (including null check)

Control flow – branching and looping (if..else, for, while, switch..case)

Introduction to Functions

© Prashanth Puranik puranik@praharaconsulting.com
+91-9448441478 puranik@digdeeper.in

Function declaration syntax

Function invocation

Anonymous functions and function expressions

Working with function references

Immediately Invoked Function Expression (IIFE)

The arguments object and handling variable number of arguments

Function context (the *this* keyword)

call( ), apply( ) and when to use which one?

Inner functions

Callbacks - passing functions as arguments

Returning Functions

Higher-order Functions and the Functional Programming Paradigm

Scope of variables – global and function scope

Scope chain

Closures and how they provide "memory" to a function

Introduction to Objects

Object Declaration using Literal Syntax

Valid property names

Accessing Properties and Methods – the dot and indexing, i.e. [ ] operator

Adding and Deleting Properties – at compile time and runtime

Built-in Classes and Singletons

Date

Primitive Type Wrappers - Number, Boolean, String

Basic array methods

Array iteration methods - Functional Programming revisited

JSON format, JSON.parse( ), JSON.stringify( )

Features of ES2015+

Block-level scoping and the use of let, const

Strings and Templating

Default arguments for functions

Object and Array Destructuring

Arrow Function syntax and semantics (how function context is bound) – choosing the

right function syntax

Rest operator – in a function, during array and object destructuring

Spread operator – with arrays and objects

Object literal syntax enhancements

Classes – definition of class, properties and methods, and creation of objects

Class Inheritance

Modules – import, export, default exports

Promises – using then( ) and catch( ), chaining serial async tasks

Promises vs the callback pattern

Using async..await for handling serial async tasks

puranik@praharaconsulting.com
+91-9448441478
puranik@digdeeper.in

CHAPTERS AND TOPICS

Getting Started

About Node.js (Node)

Downloading and installing Node

Creating a simple web server - Introduction Node and the node CLI tool

Packages and npm

Node packages, Semver, and npm registry

Introduction to package.json

Using npm to create a Node project

The npmjs.com registry

Using npm to search, install, update and uninstall packages

Using third-party modules

Global installations

Configuring npm

Setting up Development Workflows

Setting up and using npm scripts

Using nodemon

Modules and Using the Module System

Overview of Built-in modules

Using built-in modules

Creating your own modules and using them – exports, module.exports, and require

The global object

The module resolution process

Caching of module exports

How Node Works

Single-threaded execution of JavaScript code in Node

Node modules, Native C++ modules, libuv, Async I/O system calls and thread pools

© Prashanth Puranik puranik@praharaconsulting.com
+91-9448441478 puranik@digdeeper.in

Blocking vs Non-blocking I/O – Pros and cons

The event loop, setTimeout, setImmediate and process.nextTick()

Handling Asynchronous Code

Error-first callbacks

Drawbacks of the callback pattern

Brief introduction to promises

Events and Streams

Events and EventEmitter

Creating and using your own EventEmitter objects and classes

Readable and Writable Streams

Using pipes to join streams

Working with the Local System

Variables available to a script - __filename and __dirname

Working with the Node process - the process object

Working with environment variables – process.env, NODE_ENV and other variables

Working with file and folder paths – the path module

Working with files - the fs module

Handling binary data - the buffer module

Querying platform properties - the OS Module

Using Node to Build Web Applications

Making HTTP requests (ClientRequest)

Building a web server

Creating a server that serves over https

The server (Server), request (IncomingMessage) and response (ServerResponse) objects

The url and querystring modules

Using Express to Build Web Applications

What is Express.js (Express) and why use it?

Building web application using Express

Setting up project structure using Express Generator

The Application object

Routing using Application and Router objects

Responding with text, files and JSON

Redirects in Express

Defining controllers

Organizing files in an application

What is middleware in Express, and how does it work?

Setting up and using middleware

Setting up error-handling middleware

Handling cross-cutting concerns using Express middleware

Setting up a static file server for serving web assets

Setting up body parser middleware for parsing incoming JSON and form data

Using query params and path params in Express

Templating using EJS

Including partials in EJS

Integration with a Database (MySQL/PostgreSQL using mysql driver/node-postgres driver / sequelize)

Using MySQL/PostgreSQL using mysql / node-postgres driver

MySQL / PostgreSQL usage with sequelize package

Setting up relationships between tables and validating data

CRUD operations

Setting up an API that serves and stores data (handing HTTP verbs – GET, PUT, POST, DELETE)

(OR)

Integration with a Database (MongoDB using Mongoose)

Brief introduction to Mongo DB

Using MongoDB using mongodb driver

Using Mongoose for communicating with a MongoDB database

Mongoose Schemas and validations

Various model and model instance methods

Setting up an API that serves and stores data (handing HTTP verbs – GET, PUT, POST, DELETE)

Using Popular Third-party Modules

JWT-based authentication using jsonwebtoken

Logging using Chalk, Morgan and Winston

Configuration management for different environments with dotenv module