**JAVASCRIPT FUNDAMENTALS (JS-102)**

**LEVEL:** BASIC TOINTERMEDIATE

**OVERVIEW**

21 hours for QA engineers with 0 - 3 years of experience

(21 hours = 3 full days)

JavaScript (JS) is the language for scripting web pages – to enable user interactions on a web page, communicate with the backend etc. Along with HyperText Markup Language (HTML) and Cascading StyleSheets (CSS), it forms the basis for creating any web page.

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. Apart from these, transpilers like Babel, and languages like TypeScript, have further boosted application of the latest and greatest features of the JS language.

This bootcamp helps people with zero knowledge of JavaScript master basic and intermediate level features. This in turn lays a strong foundation to get started with front-end frameworks like React and Angular, as also runtimes like Node.js, that are highly sought-after skills for web developers today.

**PREREQUISITES**

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

**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>

Once an account is created by everyone, **the list of GitHub user names needs to be shared with me** - I will add them to the GitHub repository before start of training.

1. Node.js needs to be installed on all systems – Mac OSX, Linux and Windows is supported. The 20.x.x (LTS version) may be installed. This will also install npm.

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

1. Download and install Visual Studio Code (VSCode)

<https://code.visualstudio.com/download>

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

1. 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/>

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

**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?

Case-sensitivity, automatic semi-colon insertion

**Debugging**

Disadvantages of using console API

Debugging JS using the Chrome Developer Tools – Elements, Console, Source, Network and Application tabs

Using the Chrome debugger

**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

**Operators, expressions and control flow**

Operators (arithmetic, relational, logical)

How === differs from ==

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

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

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

**Introduction to Functions**

Function declaration syntax

Function invocation

Anonymous functions and function expressions

Working with function references

Function context (the *this* keyword)

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

**Introduction to Objects**

Object Declaration using Literal Syntax

Valid property names

Accessing Properties and Methods

Adding and Deleting Properties – at compile time and runtime

**Built-in Classes and Singletons**

Primitive Type Wrappers - Number, Boolean, String

Basic array methods

Array iteration methods - Functional Programming revisited

**Features of ES2015+**

Block-level scoping and the use of let, const

Strings and Templating

Object and Array Destructuring

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

Rest operator –its use during array and object destructuring

Spread operator – with arrays and objects

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

The Fetch API for making Ajax calls – handling success and error states, methods for parsing the response

**Introduction to Objects the Browser Creates**

The window object (setTimeout and setInterval)

How window acts as the global object

**DOM and the DOM API**

The Document Object Model (DOM), DOM Nodes, and the DOM tree

Accessing DOM Nodes – getElementById( ), querySelector( ), querySelectorAll( ) etc.

The innerText and innerHTML properties

Manipulating the DOM Tree – appendChild( )

Working with inline styles (style property)

Working with classes (classList and its API)

**Event Handling**

Popular Events

addEventListener()

Event Object - target, preventDefault( ), and event-specific properties

**Introduction to Unit Testing**

What is unit testing?

Advantages

Development and unit testing using TDD approach

Getting started with Jest and Jasmine

Matchers

Mocking APIs

Asynchronous tests

Typical unit tests

Setting up for code coverage

Overview of companion libraries used with Angular and React applications

**Introduction to End-to-End (E2E) Testing**

What is E2E testing?

Advantages

Unit testing vs E2E testing

Getting started with Selenium and Cypress

Overview of E2E testing React apps using Cypress