GUVI | HCL

Skill Up. Level Up

Ebook

Start Your Development Journey with JavaScript:

A Beginner's Guide

GUVI | HCL
Skill Up. Level Up

JavaScript

JavaScript is the cornerstone of modern web development, powering dynamic and interactive experiences across billions of websites. This eBook is designed to guide you through the journey of mastering JavaScript, whether you are a complete beginner or an experienced developer looking to deepen your knowledge.

We will start with the basics, such as setting up your environment and understanding the core concepts, and progress to advanced features and real-world applications.

GUVI | HCL
Skill Up. Level Up

Table of Contents

1. Introduction to JavaScript

What is JavaScript?

JavaScript is a versatile, high-level programming language that powers the interactive features of websites. Alongside HTML and CSS, JavaScript forms the backbone of modern web development. Features include:

a. Lightweight and Interpreted
b. Platform-Independent
c. Dynamic Typing
d. Prototype-Based Object-Oriented
e. Event-Driven Programming
f. Functional Programming Support

Why Learn JavaScript?

- Build dynamic web applications
- Create interactive user interfaces
- Learn the foundation for advanced frameworks like React, Vue, and Angular

Where JavaScript is Used?

JavaScript is everywhere: in web browsers, server-side applications (Node.js), and even mobile app development. Let’s talk about some of its applications

- **Web Development:** Creating interactive and dynamic websites using frameworks like React, Angular, and Vue.

- **Mobile Applications:** Developing cross-platform mobile apps with frameworks such as React Native and Ionic.
- **Server-Side Development:** Building back-end applications with Node.js.
- **Game Development:** Crafting browser-based and mobile games using libraries like Phaser.
- **Desktop Applications:** Developing desktop apps with frameworks such as Electron.
- **Automation and Scripting:** Automating repetitive tasks and writing browser scripts using tools like Puppeteer.

2. Setting Up Your Environment

Tools You Need

a. Text Editor: VS Code, Sublime Text, or Atom.
b. Web Browser: Chrome, Firefox, or Edge (with developer tools).
c. Code Playground: Platforms like CodePen, JSFiddle, or Replit.

Writing Your First Script

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Test</title>
</head>
<body>
  <script>
    console.log('Hello, World!');
  </script>
</body>
</html>
```

Open this file in a browser and check the console for the output.

About Node.JS

Node.js is a powerful JavaScript runtime that enables developers to execute JavaScript code outside of a browser. If you're new to Node.js, this guide will walk you through the process of checking if Node.js is installed, downloading and installing it, and running your first JavaScript file.

Step 1: Checking if Node.js is Installed

Before installing Node.js, check if it's already installed on your system:

1. Open the **Command Prompt** (CMD):
   - **Shortcut:** Press Win + R, type cmd, and press Enter.
   - **Path:** Click on the **Start Menu,** type Command Prompt in the search bar, and click on the result.

Type one of the following commands and press Enter:
node -v
or
node --version

2. a. If Node.js is installed, this command will display the installed version.
   b. If not, you'll see an error message indicating that the command is not recognized.

Step 2: Downloading Node.js

1. Visit the official Node.js website.
2. Download the LTS (Long-Term Support) version for stability and reliability.
3. Run the installer and follow the on-screen instructions to complete the installation.

Step 3: Exploring the Browser Console

While the installation is in progress, you can explore JavaScript basics using the browser's developer tools:

a. Open your web browser (e.g., Chrome).
b. **Shortcut:** Press F12 or Ctrl + Shift + I to open the Developer Tools.

i. **Path:** Click on the browser's menu (three dots in the top-right corner), go to **More Tools,** and select **Developer Tools.**

c. Navigate to the **Console** tab.

Type and execute the following JavaScript commands one by one:

```
let name = "Pugazh";
let company = "GUVI";
console.log(`name = ${name}, company = ${company}`);
```

d. This will output: name = Pugazh, company = GUVI.

Exploring the Sources Tab and Snippets

a. Go to the **Sources** tab in the Developer Tools.

b. In the left-hand panel, locate the **Snippets** section.

- If you don't see it, click on the three-dot menu in the Sources tab and ensure that **Snippets** is enabled.

c. Right-click in the Snippets section and select **New Snippet.**

Write the following JavaScript code in the snippet editor:

```
let name = "Pugazh";
let company = "GUVI";
console.log(`name = ${name}, company = ${company}`);
```

d. Save the snippet with a meaningful name.

e. Run the snippet by right-clicking on it and selecting **Run.**

- The output will appear in the Console tab, and the entire code will execute at once.

Step 4: Verifying Node.js Installation

Once Node.js is installed:

a. Open the Command Prompt again.
Type one of the following commands and press Enter:

node -v
or
node --version

b. This should now display the installed version of Node.js.
To open the Node.js console, type:

node

c. This will start an interactive Node.js environment where you can execute JavaScript commands one by one, just like in the browser console.

Step 5: Running a JavaScript File with Node.js

1. Open **Notepad** or any text editor.
2. Write the following JavaScript code:
   Type and execute the following JavaScript commands one by one:
   let name = "Pugazh";
3. let company = "GUVI";
4. console.log(`name = ${name}, company = ${company}`);
5. Save the file with a .js extension, for example, sample.js.
6. In the Command Prompt, navigate to the directory where you saved the file:
   - **Shortcut:** Use cd path\to\your\file.
   - **Path:** Manually navigate to the folder, right-click in the address bar, and copy the path. Paste it in the Command Prompt after cd.

7. Execute the file using the following command:
   node sample.js
8. This will give output: name = Pugazh, company = GUVI.

3. JavaScript Basics

Variables and Data Types

- **Variables:** var, let, const
- **Data Types:** Strings, numbers, booleans, arrays, objects, null, and undefined

Example

```
let name = 'Alice';
const age = 25;
console.log(`Name: ${name}, Age: ${age}`);
```

Operators

Operators are symbols or keywords used to perform operations on variables and values. They form the backbone of expressions and enable the manipulation of data. Let's discuss some of the types of operators in JavaScript:

1. Arithmetic Operators

Used for mathematical operations:

- + (Addition): 5 + 3 → 8
- - (Subtraction): 5 - 3 → 2
- * (Multiplication): 5 * 3 → 15
- / (Division): 5 / 3 → 1.67
- % (Modulus): 5 % 3 → 2
- ** (Exponentiation): 5 ** 3 → 125

2. Assignment Operators

Used to assign values:

- = (Assign): x = 5
- += (Add and assign): x += 3 → x = x + 3
- -= (Subtract and assign): x -= 3
- *= (Multiply and assign): x *= 3
- /= (Divide and assign): x /= 3

3. Comparison Operators

Compare values and return true or false:

- == (Equal to): 5 == "5" → true
- === (Strict equal to): 5 === "5" → false
- != (Not equal to): 5 != "5" → false
- !== (Strict not equal to): 5 !== "5" → true
- > (Greater than): 5 > 3
- < (Less than): 3 < 5
- >= (Greater than or equal to): 5 >= 3
- <= (Less than or equal to): 3 <= 5

4. Logical Operators

Used for logical operations:

- && (AND): true && false → false
- || (OR): true || false → true
- ! (NOT): !true → false

5. Bitwise Operators

Operate at the binary level:

- & (AND)
- | (OR)

- ^ (XOR)
- ~ (NOT)
- << (Left shift)
- >> (Right shift)

6. String Operators

Used for string operations:

- + (Concatenation): "Hello" + " " + "World" → "Hello World"

7. Ternary Operator

A shorthand for conditional statements:

- condition ? value_if_true : value_if_false: 5 > 3 ? "Yes" : "No" → "Yes"

8. Type Operators

- typeof: Returns the type of a variable: typeof "Hello" → "string"
- instanceof: Checks if an object is an instance of a specific class: obj instanceof Object

4. Control Flow and Loops

Conditional Statements

Used to execute code based on specific conditions.

- if, else if, else

```
if (age > 18) {
  console.log('Adult');
} else {
  console.log('Minor');
}
```

Loops

Used to repeat a block of code as long as a condition is met.

- for, while, and do...while

Let’s take an example of for loop

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

5. Functions, Scope, and Arrays

Functions

A function is a block of code designed to perform a specific task. It is executed when it is invoked (called).

Syntax

```
function functionName(parameters) {
 // Function body
return value; // (optional)
}
```

Function Declaration

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet('Alice'));
```

1. Traditional Functions

Traditional functions are declared using the function keyword. They can have a name and are the most commonly used form of functions.

Example:

```
function add(a, b) {
```

```
return a + b;
}
console.log(add(5, 10)); // Output: 15
```

2. Anonymous Functions

Anonymous functions are functions without a name. They are often used as arguments to other functions or assigned to variables.

Example:

```
const multiply = function(a, b) {
return a * b;
};
console.log(multiply(5, 10)); // Output: 50
```

3. Arrow Functions

Arrow functions are a concise way to define functions using the => syntax. They are particularly useful for shorter functions and are widely used in modern JavaScript.

Example:

```
const subtract = (a, b) => a - b; // Implicit return
console.log(subtract(10, 5)); // Output: 5
```

Arrays

Arrays store ordered collections of data. JavaScript arrays are resizable and can contain a mix of different data types. Some of the useful methods include:

- push()
- pop()
- map()
- filter()

Example:

```
const mixedArray = [42, "Hello", true, { name: "Alice" }, [1, 2, 3]];
console.log(mixedArray);
// Output: [42, "Hello", true, { name: "Alice" }, [1, 2, 3]]
```

6. Working with the DOM

What is the DOM?

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of objects, allowing developers to interact with and manipulate the content, structure, and style of a webpage dynamically using JavaScript.

Selecting Elements

- document.querySelector()
- document.getElementById()

```
const heading = document.querySelector('h1');
heading.textContent = 'Welcome to JavaScript!';
```

1. document.getElementById("id"); (Selects an element by its id)
2. document.getElementsByClassName("className"); (Selects all elements with a specific class name. Returns an **HTMLCollection**)
3. document.getElementsByTagName("tagName"); (Selects all elements with a specific tag name. Returns an **HTMLCollection**)
4. document.querySelector("CSS selector"); (Selects the **first element** that matches a CSS selector. This is very versatile)
5. document.querySelectorAll("CSS selector"); (Selects **all elements** that match a CSS selector. Returns a **NodeList**)

Event Listeners

- An **event listener** is a function in JavaScript that listens for a specific event (e.g., a click, key press, hover) on an element and executes a

callback function when the event occurs.

Adding Event Listeners:
Use the addEventListener() method to attach an event listener to an element.

Syntax: element.addEventListener(eventType, callbackFunction);

- Adding interactivity:

```
button.addEventListener('click', () => {
alert('Button Clicked!');
});
```

Advantages of addEventListener

- Allows multiple listeners for the same event.
- Supports event capturing (useCapture parameter).
- Flexible and modern approach to handling events.

7. Advanced JavaScript Features

Closures

A closure is a function that "remembers" the variables from its outer scope, even after that scope has finished executing. Closures enable powerful programming patterns by preserving access to the variables within the scope where they were created.

Why Use Closures?

- **Data Privacy:** Encapsulate variables within a function, protecting them from external access.
- **Stateful Functions:** Maintain and update state between function calls.
- **Event Handlers:** Store references to variables for later use.

Promises and Async/Await

A **Promise** is an object representing the eventual completion (or failure) of an asynchronous operation and its resulting value. It simplifies asynchronous code, making it more readable and easier to manage.

- **States of a Promise**

  a. **Pending:** The initial state, neither fulfilled nor rejected.
  b. **Fulfilled:** The operation completed successfully.
  c. **Rejected:** The operation failed.

- **Example:**

  Here’s an example of using a Promise with a fetch call to make a request to a public API and get some data:

```
// Public API: JSONPlaceholder (Free fake API for testing and prototyping)
const apiUrl = "https://jsonplaceholder.typicode.com/posts/1";

// Fetching data using Promises
fetch(apiUrl)
  .then((response) => {
    // Check if the response status is OK
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    return response.json(); // Parse the JSON from the response
  })
  .then((data) => {
    // Handle the fetched data
    console.log("Fetched Data:", data);
  })
  .catch((error) => {
    // Handle any errors
    console.error("Error fetching data:", error);
  });
```

Output:

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum..."
}
```

Aysnc/Await

Async/Await is a modern syntax for handling asynchronous operations in JavaScript. It provides a cleaner and more readable way to work with Promises.

How it Works

- The async keyword declares a function that always returns a Promise.
- The await keyword pauses the execution of the async function until the Promise resolves.

Modules and Imports

Modules: Modules are reusable chunks of code that can be imported and exported between files. They help organize code, promote reusability, and improve maintainability.

Organize code using ES6 modules:

import { myFunction } from './myModule.js';

Do check out Advanced JavaScript course which has got the major concepts covered like promise, Async/Await, TypeScript, CORS, FetchAPI, methods, etc.

8. Conclusion and Resources

Summary

In this eBook, we covered:

- The basics of JavaScript
- Working with variables, functions, arrays, and loops
- Advanced JavaScript Concepts
- Manipulating the DOM

Next Steps

- Practice by building small projects (e.g., a to-do list or calculator).
- Explore frameworks like React or Vue.
- Use platforms like StackOverflow, GitHub to explore more.
- You can also use our platform Webkata to practice.

Further Reading and References

Explore resources such as:

- "Eloquent JavaScript" by Marijn Haverbeke
- MDN Web Docs
- JavaScript.info

Now that you know the basics, you must definitely get into the depth and explore concepts like error handling, classes, objects, etc. which is not possible to cover in one eBook, so you should definitely go for a course which talks about it and helps you with real-world examples.

Before that, let’s explore together about GUVI.

GUVI | HCL
Skill Up. Level Up

About GUVI

GUVI (Grab Ur Vernacular Imprint), an IIT-Madras Incubated Company is First Vernacular Ed-Tech Learning Platform. Introduced by Ex-PayPal Employees Mr. Arun Prakash (CEO) and Mr. SP Balamurugan, and late Sridevi Arun Prakash (co-founders) and driven by leading Industry Experts, GUVI empowers students to master on-demand technologies, tools, and programming skills in the comfort of their native languages like Hindi, Tamil, Telugu, English etc.

Personalized Solutions

End-to-end personalized solutions in online learning, upskilling, and recruitment.

Empowering Learners

Empowering learners with tech skills in native languages.

Gamified Learning

Gamified, bite-sized videos for an interactive learning experience.

Accreditations & Partnerships

Want to know more about GUVI? Visit our website today!

About Zen Class

Zen Class is GUVI's specially curated learning platform that incorporates all the Advanced Tech Career Courses like Full Stack Web Development, Data Science, Automation Testing, Big Data & Cloud Analytics, UI/UX Designing, and more. Zen Class provides the best industry-led curriculum programs with assured placement guidance

Zen Class mentors are experts from leading companies like **Google, Microsoft, Flipkart, Zoho & Freshworks.** Partnered with 600+ tech companies, your chances of getting a high-paying tech job at top companies increases.

Why choose Zen Class?

- Assured Placement Guidance
- IIT-M Pravartak Certification for Advanced Programming
- Ease of Learning in native languages such as Hindi & Tamil
- A rich portfolio of real-world Projects
- Self-paced Online Classes
- 600+ Hiring Partners
- Excellent Mentor Support
- Easy EMI options

Do check our Zen Class - Full Stack Development Program which talks about the fundamentals of JavaScript, MongoDB, ExpressJS, NodeJS, ReactJS, AWS, and the core tools and technologies you must know. You'll also learn working on real-world projects, and get placement guidance.

GUVI | HCL
Skill Up. Level Up

Thank You

www.guvi.in