Unlock your potential and master web development

PROGRAMMING STEP BY STEP AND MORE AN INSTRUCTIVE GUIDE

UP-TO-DATE, VERSATILE, ADVANCED

BOOK 2 – Part Two

JAVASCRIPT & DOM

René F. Ruano Domínguez

DEDICATION

TO THE GENERATIONS OF THE FUTURE, MILLENNIALS, BABY BOOMERS AND BEYOND.

This How-To is dedicated to everyone: To you, the brilliant young minds shaping the world with your ingenuity and creativity. To the millennials, who navigate the digital age with ease and constantly seek new ways to innovate. To the baby boomers, who pioneered the technological revolution and now contribute their experience and wisdom. To the retirees who dream of staying up-to-date, filling their free time with useful hours, and exercising their minds.

PROLOGUE

PROGRAMMING STEP BY STEP AND MORE - Book 2 - Part Two: JAVASCRIPT & DOM is the natural continuation of a series that has earned a place in the programming community for its didactic and accessible approach. In this second installment, René F. Ruano Domínguez extends his meticulous teaching to the fascinating world of **JavaScript and the manipulation of the document object model** (**DOM**), two essential pillars for modern web development.

This book builds on the foundation of Part I, taking readers beyond static design to the creation of interactive and dynamic applications. Through clear explanations and practical examples, the author guides readers through understanding how **JavaScript** can transform a web page into a rich, adaptive experience, while the **DOM** allows them to effectively access and modify page elements.

The content is designed to develop both theoretical understanding and practical application of concepts, ranging from language fundamentals to advanced DOM manipulation techniques and event integration.

Programming Step by Step and More - Book 2 - Part Two is not just a learning guide; it's a bridge to mastery in web development. The author's dedication and passion for teaching are reflected on every page, presenting complex concepts in a simple and accessible way for everyone.

With the same enthusiasm with which the author shared his wisdom and experience in Book 1 - Part 1, this Part 2 promises to continue offering in-depth and effective learning. I am convinced that you will find in these pages a constant source of knowledge and motivation to continue growing on your journey as a programmer.

I hope you enjoy this journey toward mastering **JavaScript and the DOM** and that you find it a valuable resource for your projects and professional growth, especially in this era where Artificial Intelligence and interactive experiences are increasingly intertwined.

With respect and admiration,

Idalmy Baluja Conde

Index **Página**

GRATITUDE.

To my family, my colleagues, my friends.
Thank you, thank you so much for the unconditional support you've given me since I started this project. And to those who didn't believe in me, I'm grateful for motivating me to prove that it was worth it.

Without the encouragement, suggestions, trust they placed in me, and the personal challenge of creating a useful product, this project would not have been possible.

Book 2 – Part Two
JAVASCRIPT & DOM

René F. Ruano Domínguez

1 JAVASCRIPT, NODE.JS, ECMA AND THEIR CURRENT AND FUTURE ROLE IN ARTIFICIAL INTELLIGENCE (AI).

1.1. Introduction.

Welcome to this journey into learning JavaScript, an essential language in modern web development. Explore why JavaScript is so vital to web development and how it has evolved over time.

1.1.1. About JavaScript.

JavaScript is a high-level, interpreted, object-oriented programming language. It is widely used in web development to add interactivity and dynamism to web pages. Furthermore, with the rise of Node.js, JavaScript can also run server-side, allowing you to perform server-side tasks and build complete web applications.

In **Book 1 - Part One,** we dedicated a Chapter to describing the history of JavaScript, its progress and the scripting language standard developed according to the guidelines contained in the **ECMA-262** document, provided by Ecma International (European Computer Manufacturers Association) which gave rise to ECMA Script 6. Therefore, we recommend reviewing again the foundations on which this language has been created, as they are fundamental to understanding modern web development.

Although it's possible to start from scratch when learning JavaScript, it's recommended that you first master the programming languages HTML and CSS, as well as the development environment tools for writing and editing code. These tools are essential for learning any programming language like JavaScript. Having this knowledge from the start optimizes the learning process, allowing you to practice each script with greater precision and efficiency. At the same time, this foundation allows for a better understanding and more fluid application of the practice web pages that will be created.

Those who have completed Part One of the Tutorial, having mastered HTML and CSS through multiple practices, have a solid foundation to delve into learning JavaScript.

1.2. Current and future role of JavaScript in Artificial Intelligence (AI).

JavaScript has and will continue to play a fundamental role in artificial intelligence, especially in the field of web development and interactive applications. Its accessibility, coupled with the continued evolution of libraries and tools that facilitate AI implementation, will ensure that JavaScript remains relevant in this field for the foreseeable future. The combination of AI and JavaScript will not only expand application capabilities but also improve the user experience and offer new opportunities for innovation.

1.3. How to Get Started and Use This Book.

This Book 2 - Second Part of the "Programming Step by Step and More" Instructional Guide is the main guide of the integrated learning system, designed to facilitate the understanding of each concept and code presented.

In the book, students will learn the code and program logic they will write in their code editor, Visual Studio Code. Once written and saved in the editor, they will be executed automatically. They can then be published locally in the browser to verify the effects and results of their execution.
To complement the learning process, the system connects to an online website where the codes are stylized and adapted to web standards. These pages also expand on the knowledge and examples studied in the book, reinforcing what has been learned and practiced.

To access the online webpage, QR codes have been included in the right margin of the related topic. Scanning this code with your cell phone's camera and clicking on the URL displayed on the screen will open the page associated with that topic, allowing you to check and validate the online exercises.
This QR code is only a sample. In this case, it takes you to the first page of this learning system.

1.4. The Development Environment.

Those who have completed the First Part of the Book 1 - First Part Instructions know how to configure and work with the necessary tools of the development environment to program, execute and code.

These tools necessary for the development of learning are:

a) VSC and the main extensions.
b) Node.js.
c) Know how to work with the Browser Development Inspector.

It is vital to master these tools in order to execute the codes that will be practiced throughout the Instructional.

2 TOPICS COVERED BY THE BOOK.

This **Second Part** of the Tutorial contains the basic concepts of the JavaScript programming language and a comprehensive section covering the Basic and Intermediate Level of the **Document Object Model** (**DOM**).

2.1. The basics of the JavaScript language.

- Data types. Primitive (numbers, strings, Booleans, null, undefined). Complex (objects, arrays).
- Variables. Declaration and initialization. **var** , **let** , **const** . Scope (global and local).
- Operators. Arithmetic, Comparison, Logical, Assignment.
- Control Structures. Conditionals (if, else if, else, switch)

Loops (for, while, do-while).

- Functions. Declaring and calling functions. Parameters and arguments. Anonymous functions and arrow functions.
- Objects and Arrays. Data Structures. Object and Array Syntax. Arrays. Methods and Properties.

Once the basic preparation is complete, we will continue with the exercises and practices to master the DOM, covering:

2.2.Document Object Model (DOM). Basic and Intermediate Level .

- Hierarchical structure of the DOM. Learn to select, modify, and manipulate HTML elements using JavaScript. The DOM's role in the interaction between HTML and JavaScript.
- DOM Box Model. Understanding the CSS box model. The relationship between CSS styles and the DOM.
- DOM manipulation.
- Selecting DOM Elements.

Selection methods (getElementById, getElementsByClassName, getElementsByTagName).

- Modern Selection Methods (querySelector, querySelectorAll). Comparison and efficient use of selection methods.
- Content Modification. Changing text content (textContent, innerText). Modifying inner HTML (innerHTML).
- Create and add new elements (createElement, appendChild, insertBefore).
- Attribute Manipulation. Accessing and modifying attributes (getAttribute, setAttribute, removeAttribute).
- Managing CSS classes (classList.add, classList.remove, classList.toggle). Styles and Classes. Changing styles inline (style.property). Adding and removing CSS classes (className, classList).
- Form Management. Accessing and manipulating form elements.
- Form validation with JavaScript. Collecting form data (value, checked, etc.).
- Events. Introduction to DOM events (click, submit, mouseover, etc.). Assigning event handlers (addEventListener, removeEventListener). Event propagation (bubbling and capturing). Preventing default behavior (preventDefault). Stopping event propagation (stopPropagation).
- Traversing of the DOM. Traversing methods (children, parentElement, nextElementSibling, previousElementSibling).
- Node comparison (compareDocumentPosition, contains).
- Creating and Deleting DOM Elements. Creating new nodes (document.createElement, document.createTextNode).
- Insert nodes (appendChild, insertBefore). Remove nodes (removeChild, replaceChild).
- Document Fragments. Using DocumentFragments to improve the efficiency of DOM manipulations.
- Node Cloning. Cloning DOM elements (cloneNode).

- Working with NodeList and HTMLCollection. Differences between NodeList and HTMLCollection. Iterating over node collections.
- Performance and Optimization. Techniques for optimizing DOM manipulations. Minimizing reflow and repainting.
- Best Practices. Best practices for efficient and maintainable DOM manipulation.

2.3. Exercises and practices to be performed.

- More than 200 practical exercises that the student will complete during the Instructional Course reinforce and consolidate the learning of each of the language's topics, algorithms, and methods.

2.4. APPENDIX 1. Script of all the web pages created during the Instructional to carry out the practices and exercises.

2.5. APPENDIX 2. Catalog of DOM Functionalities: Methods and Properties.

2.6. APPENDIX 3. Summary of all QR codes for quick access to additional content and complementary resources available on the online page.

3 WORK TOOLS TO INSTALL.

It is a requirement to have the essential tools for learning any programming language installed on your computer.

In Book 1 - Part One, you can find detailed information on how to download, install, and learn how to use these tools. However, here's a summary.

3.1. Work tools to be installed on the computer.

1. The code editorVisual Studio Code,
2. JavaScript plugins.

I describe below where you can download these tools.

3.2. Visual Studio Code

The recommended code editor for this course is Visual Studio Code. You can download it from its official website. Scan the QR code provided on the right to directly access the download site.

3.3. Node.js

To work efficiently with JavaScript, you need to install Node.js. Visit the official Node.js website and download the appropriate installer for your operating system. Scan the QR code on the right.

On both websites, you'll find instructions for installing them on your computer and documentation for the most essential extensions for editing code. Extensions act as essential complements for various tasks, including highlighting errors when writing code, making it easier to correct them instantly.

It's also recommended to use the Google Chrome browser and learn how to work with its Development Inspector, primarily the Elements and Console tabs. If you haven't installed Chrome before, you can download it from its official website. Book 1 - Part One explains, and

includes exercises and practices on how to work with both tabs of the Development Inspector, primarily the browser's Console.

If you still have questions about the operation and installation of the above tools, you can consult the internet, where you will find extensive information, both written and published in videos. YouTube offers dozens of videos on this topic, and the websites of each of these applications contain information on the installation and setup of these tools. Having these tools on your computer is essential for developing the content of this tutorial.

3.4. Visual Studio Code extensions to install.

Visual Studio Code extensions are very useful helper applications that make it easier for users to write, run, and troubleshoot code. Some of the most commonly used extensions include:

-Auto Close Tag	-Live Preview	-PHP Debug
-Auto Rename Tag	-Live Sass Compiler	-PHP DocBlocker
-Bearded Theme	-Material Icon Theme	-PHP Getters & Setters
-Composer	-Microsoft Edge Tools for VS Code	-PHP IntelliSense
-Debugger for Firefox-Chrome	-Open in browser	-PHP Profiler
-IntelliPHP - AI Autocomplete for PHP	-PHP	-PHP Server
-Live Server	-PHP cs fixer	-Prettier - Code formatter

There are numerous extensions for Visual Studio Code that can significantly improve your web development experience. From the editor itself, you can explore the available extensions and decide which ones to install based on your needs.

Conclusions:

In this chapter, we've summarized the essential tools for setting up an efficient development environment on your computer and where to find the sources to download and install them.
The advantages of using these tools in learning are diverse. We'll mention a few:

Visual Studio Code: This is the recommended code editor for its versatility and large user community. It's essential for writing, editing, and debugging code effectively.
Node.js: This platform is essential for running JavaScript outside the browser, facilitating the development of backend applications and automation scripts.
Visual Studio Code extensions: Significantly improve the development experience by providing tools such as smart auto-completion, code formatting, integrated debugging, version control with Git, and more.

It is necessary, but not essential, to have practice and skills in the operation of the development environment, to know the rules for writing, executing and checking each algorithm that will be studied during the progress of the Instructional.

The initial setup may take some time, but the long-term benefits in terms of productivity and code quality make it worth the investment. By following these steps and using these tools effectively, you'll be well prepared for optimal learning.
If you've completed Book 1 - Part 1, you've acquired this knowledge and skills. You're well prepared.

CONCLUSIONS BOOK 2 - SECOND PART.

This **Book 2 - Part 2 of Programming Step by Step and More** has provided a solid foundation in the language's fundamental concepts, from data types and control structures to the use of functions, objects, and arrays. By learning these basic and intermediate-level aspects, the reader gains a comprehensive understanding of how JavaScript works as a programming language and its interaction with the Document Object Model (DOM), enabling the creation of robust and efficient web solutions.

The next key step was to explore in detail the Document Object Model (DOM), an essential tool for any web developer who wants to interact with HTML pages dynamically. Throughout the DOM chapter, we covered the methods and techniques needed to select, modify, and manipulate HTML elements, which is essential for creating interactive web interfaces. In addition, we presented modern methods and best practices for optimizing performance in DOM manipulations, allowing readers to work efficiently and professionally on complex projects.

Advanced DOM Manipulation: Understanding the hierarchical structure of the DOM and how to use JavaScript to access, modify, and delete nodes is critical for creating dynamic web applications. Event handling, forms, and performance optimization provide readers with a comprehensive set of tools for improving user experience and application efficiency.

Practical Exercises: The hundreds of exercises included in this book provide a valuable opportunity for readers to apply their knowledge in real-life situations. Through practical problem-solving, key concepts are reinforced, helping to consolidate learning and gain confidence in implementing solutions in future projects.

Future prospects:

With the fundamentals of JavaScript and a mastery of the DOM well established, the reader is ready to move on to more specialized areas. In **Book 3 - Part Three (JavaScript**, **Intermediate and Advanced Level**, **Node.js) of Programming Step by Step and More**, we will delve into the higher-level methods in the JavaScript language, as well

as its processes and algorithms. Topics such as Advanced Understanding of Object-Oriented Programming (OOP) in JavaScript, Prototypes, Mastering Classes and Subclasses in JavaScript, Data Handling with JSON, Asynchronous and Synchronous Requests, Asynchronous Programming with `async` and `await`, Comparison of AJAX and Modern Asynchronous Programming and extensive and detailed content on Node.js Fundamentals, Server Development, Database Operations, Advanced Topics and Practical Projects.

This will expand the knowledge and skills needed to handle larger and broader projects, such as developing complete web applications using modern libraries and frameworks like **React** ,**Vue** , and **Angular** , optimizing performance in high-traffic applications, and exploring advanced topics like asynchronous data handling with AJAX and Promises. Constant practice and ongoing learning in this dynamic field will be essential to continue progressing in this direction.

REFERENCES.

1. MDN Web Docs (Mozilla Developer Network). https://developer.mozilla.org/es/docs/Web/JavaScript
2. ECMAScript Specifications: https://tc39.es/ecma262/
3. The Modern JavaScript Tutorial https://javascript.info/
4. Node.js v22.6.0 documentation https://nodejs.org/docs/latest/api/
5. W3Schools JavaScript Tutorial https://www.w3schools.com/js/
6. MDN Web Docs: JavaScript Reference: https://developer.mozilla.org/es/docs/Web/JavaScript/Reference
7. "Eloquent JavaScript" de Marijn Haverbeke:
8. "JavaScript: The Good Parts" de Douglas Crockford:
9. "You Don't Know JS" (serie) de Kyle Simpson:
10. "JavaScript &JQuery: Interactive Front-End Web Development" de Jon Duckett
11. "DOM Enlightenment" de Cody Lindley
12. "Learning JavaScript Design Patterns" de Addy Osmani

APPENDIX. Catalog of DOM Functionalities: Methods and Properties.

1) Difference between Methods and Properties.

METHOD OR FUNCTION	PROPERTY
The method is a function that performs an action.	The property stores a value or state of an element.

2) DOM Methods and Properties

M or P	Method or Property	Functionality (Purpose)
M	document.getElementById('id')	Selects the DOM element (or selector) with id='title'.
M	getElementsByClassName('class')	Selects DOM elements with class='class'.
M	getElementsByTagName('p')	Selects DOM elements based on their HTML tag name.
P	id.textContent	Contains the value or content stored in the element with id='title'. It is concatenated with the ID referenced by document.getElementById('title'). Note: textContent escapes HTML tags and displays them as plain text. For example, if you try to bold a text using or , it will show the tags literally.
M	document.querySelector('.list > li')	Selects the first element that is a direct child of an element with class .list.
M	document.querySelectorAll('li')[3]	Selects all elements and accesses the 4th one (index 3).
M	document.createElement('li')	Creates a new element not yet inserted into the DOM.
M	id.innerHTML	Ensures that HTML tags within the string are processed and rendered as HTML, not plain text.
M	id.insertBefore	Inserts another element before the first selector with a predefined ID.

M or P	Method or Property	Functionality (Purpose)
M	class or id.removeChild	Removes a child node from a DOM element.
M	getComputedStyle()	Retrieves the computed (final) style of an HTML element.
M	querySelector()	Searches the entire DOM for the first element matching the given selector.
M	createElement	Creates a new HTML element but does not automatically insert it into the DOM.
M	createTextNode	Creates a new text node (a text fragment inside an HTML element).
M	addEventListener(eventType, callbackFunction)	Adds an event listener. Takes two arguments: (a) eventType to listen for (e.g., 'click', 'mouseover', 'keydown'); and (b) callbackFunction to execute when the event occurs.
M	removeEventListener	Removes an event previously assigned to an element.
M	preventDefault()	Cancels the default action of an event. Example: before submitting a form, you can validate user data and call preventDefault() to stop submission if there's an error.
M	appendChild	Adds a new node to the end of a parent node's list of children.
M	insertBefore	Inserts a node into the DOM before an existing reference node.
M	removeChild	Removes a child node from a parent element.
M	replaceChild	Replaces a child node within a parent element with another node.
M	getAttribute	Retrieves the value of an HTML element's attribute (e.g., href, src, alt).
M	setAttribute	Sets or updates an attribute on an HTML element (e.g., href, src, class).
P	removeAttribute	Removes an attribute from an HTML element (e.g., disabled,

M or P	Method or Property	Functionality (Purpose)
		class, target).
P	classList	Provides convenient access to manage CSS classes of an element.
P	classList.add	Adds a new class to an element.
P	classList.remove	Removes an existing class from an element.
P	classList.toggle	Toggles (adds or removes) a class on an element.
P	className	Allows access and manipulation of CSS classes associated with an HTML element.
P	cloneNode(true)	Creates a duplicate copy of a specified DOM element.
P	NodeList	A collection of DOM nodes matching specific conditions (e.g., all <p> elements).
P	HTMLCollection	An array-like object representing a list of HTML elements matching certain criteria.
M	stopPropagation()	Prevents further propagation of an event through the DOM hierarchy.
P	event.target	Identifies the specific DOM element that triggered the event.
M	preventDefault()	Prevents the default behavior associated with an event.
M	nextElementSibling	Accesses the next sibling element in the DOM tree.
M	previousElementSibling	Accesses the previous sibling element in the DOM tree.
M	compareDocumentPosition()	Determines the relative position of one node to another in the DOM hierarchy.
M	contains()	Checks whether one element or string contains another.
FC	DocumentFragment	A lightweight, in-memory document fragment used as a temporary container for groups of nodes before inserting them into the DOM.
M	window.innerWidth	Returns the inner width of the

M or P	Method or Property	Functionality (Purpose)
		browser window in pixels.
M	window.onresize	Triggered whenever the browser window is resized.
M	window.onload	Fires when the entire web page (including images, scripts, and styles) has finished loading.
M	window.onclick	Triggered when the user clicks anywhere in the browser window.
M	document.createDocumentFragment()	Creates a document fragment that can temporarily hold groups of nodes before inserting them into the DOM.
M	document.execCommand()	Executes editing commands like copy, paste, or text formatting in an element.
M	element.closest(selector)	Returns the closest ancestor (or the element itself) matching the specified selector.
M	element.matches(selector)	Checks if an element matches a given CSS selector.
M	element.scrollIntoView()	Scrolls the element into the visible area of the browser window.
M	window.scrollTo(x, y)	Scrolls the browser window to a specific position.
M	document.importNode()	Imports a node from another document into the current one.
M	element.replaceWith()	Replaces an element with another node or text.
P	node.firstChild / node.lastChild	Return the first and last child nodes of an element.
P	node.parentNode	Returns the parent node of the current element.
P	element.innerText	Similar to textContent, but respects visual formatting (e.g., line breaks).
P	element.outerHTML	Returns the complete HTML of the element, including itself.
P	node.nodeType	Returns the type of the node (e.g., 1 for elements, 3 for text nodes).
P	node.nodeValue	Returns or sets the value of a node.

M or P	Method or Property	Functionality (Purpose)
P	element.style	Allows direct access and modification of an element's inline styles.
P	element.dataset	Accesses custom data attributes (data-*) of an element.
P	document.readyState	Indicates the document's loading state (loading, interactive, complete).
P	window.localStorage / window.sessionStorage	Provide persistent or session-based data storage in the browser.

(1) When using the methods **getAttribute()**, **setAttribute()**, and **removeAttribute()** on data input (interactive) elements such as <input>, <select>, and <textarea>, or on non-interactive elements such as <p>, <h1>, and <article>, there are some notable differences in their application and behavior due to the nature and function of these elements.

Interactive elements have special attributes and unique behaviors because they are designed for user interaction, and their values can change in real time (based on user input). For these elements, the actual field value (**value**) is not retrieved using **getAttribute("value")**, but by directly accessing the **value** property.

getAttribute("value") will return the initial value assigned to the **value** attribute when the page was loaded (if it was defined in the HTML). However, if the user modifies the input field value, that change will **not** be reflected when using **getAttribute()**; you would need to use **input.value** to obtain the updated value.

Elements that do not have direct user interaction, unlike input fields, tend to have more static attributes that are less affected by the state of the document.

ABOUT THE AUTHOR.

After several decades dedicated to improving the energy efficiency of heating and cooling equipment and systems, I faced a major life change: retirement. Upon retiring, I felt disoriented and without purpose, which affected my emotional well-being. To overcome this feeling, I sought a new activity that would keep me active and motivated.

I decided to share my experiences and the knowledge I had accumulated over more than four decades working in the industry. This is how this project was born—with the goal of publishing articles, calculation tools, and guides on the various disciplines I had practiced throughout my engineering career, including web programming, which I used to optimize project management, investments, finance, and marketing. Based on my own experience, web programming is a powerful tool for solving complex problems in both industrial environments and technical services, as well as in everyday life.

My commitment to learning and technological renewal has driven me to stay updated and deepen my knowledge in new areas.

This **Second Part of PROGRAMMING STEP BY STEP AND MORE** was created from the conviction that JavaScript and the understanding of the DOM are essential in today's digital world.

This project is designed to help people of all ages enhance their professional skills and achieve their career goals. Whether starting a new career or seeking new ways to apply one's experience, this resource provides the tools and opportunities needed to grow.

Learning a new programming language, such as JavaScript, is an excellent way to make good use of free time, strengthen one's résumé, and stay up to date in an increasingly digital job market.

The Author

PROGRAMMING STEP BY STEP AND MORE ®

An Instructive Guide - Book 2 · Part Two

JAVASCRIPT & DOM

Published in Paperback and Digital
Miami, Florida, USA · 2025

Translated from Spanish by René F. Ruano Domínguez
Originally published in Spanish as *"PROGRAMANDO PASO A PASO Y MÁS* – INSTRUCTIVO – LIBRO 2: JAVASCRIPT & DOM*"*

Publisher: Corporate Luxury Group, LLC