PROGRAMMING STEP BY STEP AND MORE

AN INSTRUCTIVE GUIDE

UP-TO-DATE, VERSATILE, ADVANCED

BOOK 2 – Document Object Model 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 – DOM – Book 2, Second Part is the natural continuation of a series that has earned its place in the developer community thanks to its didactic and accessible approach. In this volume, René F. Ruano Domínguez provides a didactic, step-by-step roadmap to DOM mastery, blending pedagogical clarity with practical, illustrated exercises for modern web development.

The author simplifies complex concepts through clear, practical explanations, using an innovative learning system with illustrated techniques and progressive exercises. Each chapter reflects his dedication to offering a structured experience where theory and practice combine to build a deep understanding of the DOM

The content has been designed to strengthen both conceptual foundations and real-world application: from the basics of DOM manipulation to advanced techniques for event integration and interactive dynamics. In this way, the book goes beyond being a study guide—it becomes a bridge toward mastery in web development, showing how to fully leverage the DOM's potential to build more efficient, scalable, and professional projects.

With the same enthusiasm he shared in Book 1 – First Part, this second installment reaffirms the author's commitment to producing specialized, accessible, and highly valuable materials. His purpose is clear: to bring readers closer to the possibilities of the DOM through a practical and approachable methodology, consolidating a learning path that contributes to professional growth in the field of web development.

I am confident that you will enjoy this journey as much as I have, immersing yourself in the fascinating world of the DOM. Within these pages, you will find a valuable resource to support the creation of new projects and to strengthen your personal development as programmers, providing practical tools and knowledge that will make a real difference in your professional path.

With respect and admiration,

Idalmy Baluja Conde

Book 2- Part Two

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 – Document Object Model DOM

René F. Ruano Domínguez

1. Introduction

Mastering the DOM (Document Object Model) is essential for any developer who wants to enhance web applications. The DOM connects the logic of any programming language with the structure and behavior of a webpage, turning code into interactive and dynamic experiences.

Modern languages such as JavaScript, TypeScript, Python, Java, C#, Ruby, and PHP, along with various frameworks, benefit from the DOM's capabilities to expand projects and deliver more robust and scalable solutions. At an intermediate or advanced level, developers can manipulate elements, integrate events, and update interfaces in real time, significantly improving the quality and usability of their applications.

In short, the DOM is not just a technical tool—it is the bridge that transforms code into tangible experiences, enriching user interaction and consolidating web development as a comprehensive and multidimensional discipline.

Key Features of the DOM

a) Interactivity and Dynamism: The DOM allows developers to build highly interactive and dynamic web applications. With proper use, page elements can be updated and manipulated in real time, greatly enhancing the user experience.

b) Performance Optimization: Advanced knowledge of the DOM helps optimize web application performance. Developers can reduce costly operations and improve code efficiency, resulting in faster load times and smoother responses.

c) Cross-Browser Compatibility: Mastering the DOM ensures that web applications work consistently across different browsers and devices. This is crucial for providing uniform user experience and avoiding compatibility issues.

d) Solving Complex Problems: Advanced DOM skills enable developers to tackle complex challenges related to interaction and element

manipulation. This includes debugging errors and implementing efficient solutions for specific problems.

e) Enhancing User Experience: By learning to manipulate the DOM effectively, developers can design more intuitive and engaging interfaces. This includes adding, removing, or modifying elements in response to user actions.

Summary

An intermediate to advanced mastery of the DOM is essential for any JavaScript developer aiming to build modern, efficient, and high-quality web applications.

In this book, we recommend using the Visual Studio Code editor. Once you save your work in the editor, the program runs automatically, and you can view the results instantly in your browser. To reinforce learning, the system also connects to an online webpage where the code is styled and adapted to modern standards. There, you'll find examples and explanations that expand on the book's content, helping you practice and strengthen your skills.

Additionally, each topic includes QR codes in the right margin. Simply scan them with your phone's camera and tap the link displayed on the screen. This will take you directly to the related webpage, where you can review and validate the exercises online.

The QR code shown here is just a sample—in this case, it leads to the first page of the learning system.

Conclusions Book 2 –DOM. (DOCUMENT OBJECT MODEL)

We have studied and practiced the main structures, methods and functionalities of the DOM (DOCUMENT OBJECT MODEL).
We've explained and verified that selecting elements on web pages is a basic operation in the DOM. We've applied modern methods like **querySelector** and **querySelectorAll** and verified that they offer an efficient way to select elements based on CSS selectors.

Modifying the DOM content has been a task we've been working on throughout this chapter. By using methods like **textContent**, **innerHTML**, and **appendChild** , we learned how to dynamically update information on the page.

We also dedicate a significant amount of time to learning and practicing event operations and their importance in creating interactive interfaces. When handling events like **click**, mouseover ,and **submit,** we're asked to project responses to potential user actions and update the DOM accordingly.

The factors that influence DOM efficiency and performance were not left out of the chapter. We confirmed that DOM performance is a critical factor in the user experience. We explained in detail the tools for minimizing reflow and repainting, and using techniques like **requestAnimationFrame** to improve the performance of web applications.

Finally, we detailed libraries and frameworks like **React** and **Vue** that use the Virtual DOM concept to optimize performance. By maintaining a virtual representation of the DOM and performing updates efficiently, these frameworks enable the creation of large-scale web applications.

The DOM continues to evolve today. With new features and browser improvements, we can expect web development to become even more dynamic and efficient.

The DOM is the foundation for technologies like Web Components and Shadow DOM. These technologies allow for the creation of reusable and encapsulated components, making it easier to build modular web applications.

If you've reached the end of **Book 2 - Part 2 of this how-to**, and have studied, drilled, and practiced the hundreds of examples and algorithms presented, you can now feel confident in designing up-to-date, modern web pages. But you still need to master the higher-level features of javascript. We'll cover them in the **Book 3 - Part Three (Javascript**, **intermediate and Advanced level**, **Node.js) of Programming Step by Step and More**.

16 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

17 APPENDIX 1 - Script for all the web pages created.

17.1 DOM. (DOCUMENT OBJECT MODEL)

18 APPENDIX 2. 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 or P	Method or Property	Functionality (Purpose)
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	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 or P	Method or Property	Functionality (Purpose)
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, 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.

M or P	Method or Property	Functionality (Purpose)
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 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).

M or P	Method or Property	Functionality (Purpose)
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.
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.

MOTIVATIONAL PHRASES.

"Success is the sum of small efforts repeated day after day."
Robert Collier

"The only limit to your fulfillment tomorrow will be your doubts today."
Franklin D. Roosevelt

"It doesn't matter how slow you go as long as you don't stop."
Confucius

"The path to success and greatness is to develop what you already have."
ZigZiglar

"The secret of success lies in knowing more than others."
Aristotle Onassis

"It's not about how many times you fall, but how many times you get up."
Vince Lombardi

"Success is not the key to happiness. Happiness is the key to success. If you love what you do, you will be successful."
Albert Schweitzer

"The only way to do great work is to love what you do."
Steve Jobs

"The biggest risk is not taking any risks. In a world that's changing so fast, the only strategy that's guaranteed to fail is not taking risks."
Mark Zuckerberg

"Motivation drives us to begin, and habit keeps us going."

Jim Ryun

"It's not enough to tell the truth; you have to prove that you have it."
Baltasar Gracián

If you can't fly, run; if you can't run, walk; if you can't walk, crawl, but keep moving toward your goal."
Martin Luther King Jr

“No matter your physical age, you can always motivate your mind to face tomorrow.”
From the Author

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 MAS* – INSTRUCTIVO – LIBRO 2: JAVASCRIPT & DOM*"*

Publisher: Corporate Luxury Group, LLC