Simplifying Programming – Beginner

This material was developed to assist beginners in their programming learning journey. It covers everything from basic concepts to creating interactive web pages using HTML, CSS, and JavaScript. No matter where you're starting from, this guide is designed to make programming accessible and straightforward, helping you build a solid foundation for your career in technology. Get ready to dive into this world and take your first steps as a web developer!

por Júlia Bacchi

Initial Tips

1 Problem-Solving

A crucial skill is being able to solve problems independently. Before asking for help, try to understand the error or challenge on your own. Review the code, experiment with different solutions, and use documentation to find answers. This helps strengthen your problem-solving ability and makes you a more confident programmer.

2 Support Resources

Take advantage of valuable resources from sites like MDN (Mozilla Developer Network) and W3Schools to find programming content and information. Explore their resources to solve doubts and deepen your knowledge. Don’t hesitate to use them!

3 Constant Practice

It’s not enough to just watch content or read about programming; you need to apply what you learn! When you learn a new concept or technique, replicate it in your own project, even if it’s simple. Constant practice helps solidify knowledge and develop your skills.

4 Don’t Get Discouraged

It’s normal to feel frustrated while learning to program. Remember that everyone goes through this. When you hit a wall, take a break, take a deep breath, and look for video tutorials or online forums. Seeing someone explain or solve a problem can make all the difference.

Introduction to Programming Languages

When you start learning programming, the first step is usually to become familiar with markup and styling languages, HTML and CSS. But what exactly are these languages?

HTML (HyperText Markup Language)

HTML is the markup language used to create the structure of a web page. Think of HTML as the skeleton of a page: it defines the basic layout and the elements you see on a screen, such as headings, paragraphs, images, and links. Each of these elements is represented by tags in the HTML code, which organize and structure the content.

CSS (Cascading Style Sheets)

CSS is the language used to define the visual appearance of web pages. If HTML is the skeleton, CSS is the skin and clothes: it allows you to style the elements created with HTML, controlling colors, fonts, spacing, and overall layout. With CSS, you can transform a simple page into something visually appealing and responsive, adjusting the design for different devices and screen sizes.

Where to Start

Starting with HTML and CSS is a great way to understand the fundamentals of the web. These languages are relatively easy to learn, and their effects are immediately visible in the browser, which makes learning motivating. With HTML, you create the foundation; with CSS, you bring the page to life. Together, they form the cornerstone for any web developer.

VS Code: Your Development Environment

Visual Studio Code (VS Code) is a powerful code editor widely used by developers around the world. It is known for its versatility and ease of use, making it an excellent choice for both beginners and professionals. In this section, you will learn how to set up and use VS Code to optimize your workflow and code effectively.

Getting Started

1

Download

Download VS Code from code.visualstudio.com for the version compatible with your operating system and follow the initial setup instructions.

2

Creating the Project

On your computer, create a new folder where you will store the files for your new project.

In VS Code, go to File and locate the created folder.

3

How to Start

The project should be opened in its entirety in VS Code, as individual files do not provide a complete view. By opening the project folder, all files and resources will be accessible.

Understanding the Environment

- **Explorer:** In the left sidebar, the Explorer displays all the files and folders in your project, allowing you to easily navigate between them.
- **Editor:** This is the main working area. Each open file will appear in a tab at the top.
- **Integrated Terminal:** VS Code includes a terminal for running commands directly within the editor interface. Access it via **View > Terminal**.

Creating the Files

1

First File

To start an HTML document, create a file in the Explorer and name it **index.html**. For the basic HTML structure, type **!** (or **html:5**) and press Enter. The essential tags will be filled in automatically.

2

Subfolders

You can include subfolders such as **styles** for CSS and JavaScript files, and **images** to organize and include all images for your project.

3

Previewing

To open the HTML file in a browser, locate **index.html** in the project folder, or double-click it in VS Code. Now you're all set to start coding and tracking your progress!

Important!

> Save regularly after each modification! This practice helps prevent loss of progress and is essential for keeping your work organized and efficient. After saving, return to the browser and refresh the page: This will ensure that the changes made in the code appear on the web.

Online Code Editors

Don’t want to install an editor on your computer?

Try online tools like CodeSandbox! You can write, test, and share code directly in your browser. At every stage of your programming journey, tools like this provide a convenient and accessible way to experiment with projects without needing to set up a local environment. They are especially useful for quick prototyping and real-time collaboration with other developers.

Commands and Shortcuts

Check out some of the amazing functionalities in the world of coding!

Save and View

- **Save File:** Press **Ctrl + S** (Windows/Linux) or **Command + S** (macOS) to save the file you're editing. This ensures that all changes are recorded on disk.
- **Refresh Page:** After saving changes to your code, press **Ctrl + R** (Windows/Linux) or **Command + R** (macOS) to refresh the page in the browser and view the updates.

HTML Structure

- **Basic Structure:** Type **!** and press Enter, or **html:5** and press Enter.

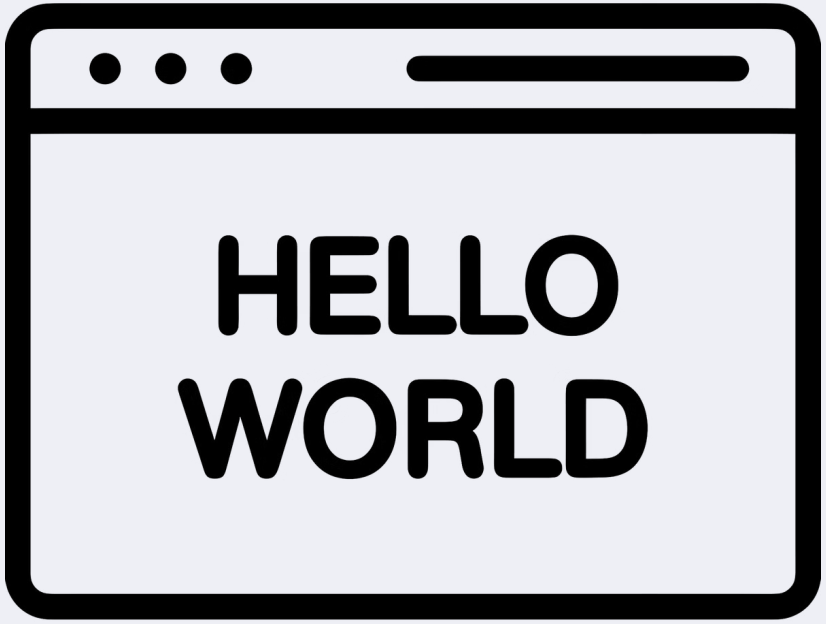
Navigation and Editing in VS Code

- **Search for Files and Commands:** Use **Ctrl + P** (Windows/Linux) or **Command + P** (macOS) to open the quick search bar and find files or execute commands.
- **Global Search:** To search for a word or phrase throughout the project, use **Ctrl + Shift + F** (Windows/Linux) or **Command + Shift + F** (macOS).
- **Switch Between Open Files:** Use **Ctrl + Tab** (Windows/Linux) or **Command + Tab** (macOS) to switch between open files in VS Code.

Development Tools

- **Inspect Elements:** Use **Ctrl + Shift + I** (Windows/Linux) or **Command + Option + I** (macOS) to open the developer tools in the browser and inspect the HTML and CSS structure of the page.

These shortcuts are practical and help improve efficiency while coding, especially for beginners. Integrating them into your workflow can make a significant difference in productivity.



<h1>Hello, World!</h1>

This message is traditionally the first step in any coding journey. It represents the first functional code and marks the beginning of your adventure in technology.

This is a basic and important exercise, as it is a way to test your development environment, ensuring that everything is set up correctly.

Moreover, it’s a simple reminder that great journeys begin with small steps.

HTML: Structuring Your Page

Understanding HTML structure is fundamental for any web developer. HTML is the foundation of all web pages, defining how content is organized and presented. When you understand the structure, you can create more organized, accessible, and search engine optimized pages. Additionally, a good structure makes code maintenance and collaboration easier, ensuring that other developers can understand and work with your code effortlessly. In summary, mastering HTML is the first step toward creating functional and efficient websites.

Basic Structure of HTML

Code	Description
<!DOCTYPE html>	Document type declaration. Informs the browser that the document is HTML5.
<html>	Root element. Contains all the content of the page.
<head>	Head of the document. Includes information about the document, such as the title and links to CSS files.
<title>	Within the <head>. Defines the page title (appears in the browser tab).
<meta charset="UTF-8">	Within the <head>. Defines the character encoding for the document.
<body>	Body of the document. Contains the visible content of the page, such as text, images, and links.
<h1>, <h2>, <h3>, etc.	Headings of different levels.
<p>	Defines a paragraph of text.
<a href="URL" target="_blank">	Creates an external link to other pages. The target attribute makes it open in a new tab.
Ir para a seção	For navigating within the same page.
<img src="URL" alt="Descrição">	Within the <body>. Adds an image.

Code Example

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>My Page</title>
</head>
<body>
    <h1>Welcome to HTML!</h1>
    <p>This is a paragraph of text.</p>
    <a href="https://www.example.com" target="_blank">Visit our site</a>
    <img src="image.jpg" alt="Image Description">
</body>
</html>
```

Text Formatting Tips

Italic: *SheCodes Workshops*

Bold: **SheCodes Workshops**

Line Break: Welcome to my website
 We hope you enjoy your stay

Section Separator: Welcome to my website<hr /> Discover latest updates below

HTML Semantics

This means using elements that clearly describe the purpose of the content they enclose. When used correctly, these elements clearly indicate the function of sections within a page. Using semantic tags improves accessibility, makes the code easier for other developers to understand, and helps search engines better index the content.

Semantic Elements

What are the basics?

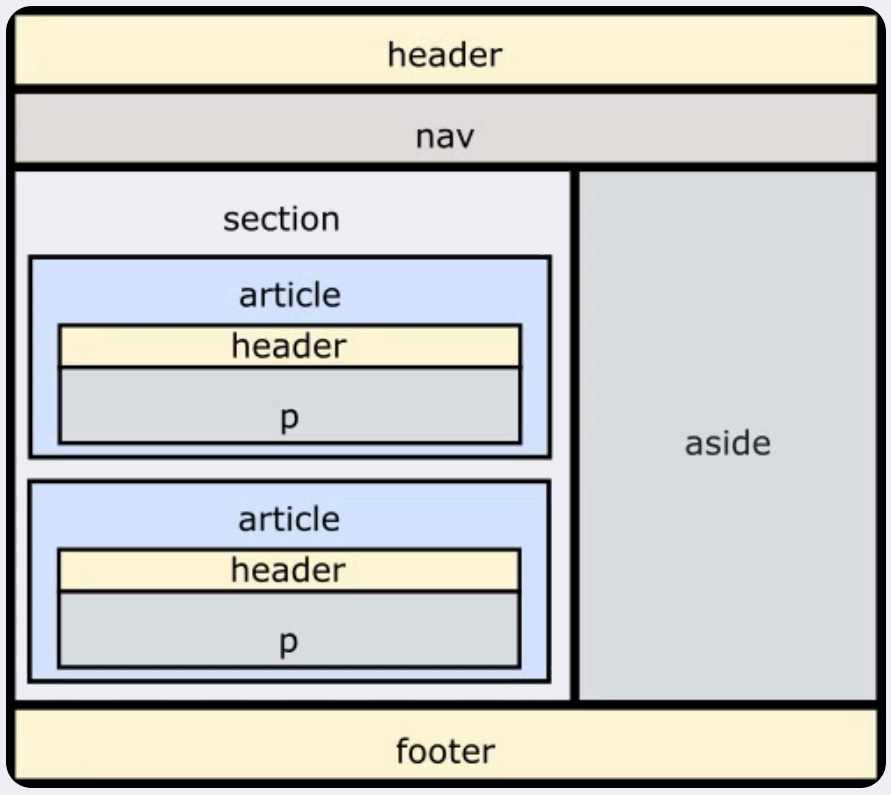
<header> for the header of a page.

<nav> for a section of navigation links.

<section> for grouping related content.

<article> for independent content, such as a blog post.

<footer> for the footer of the page.



▼ **<aside>**

> This element may not be used as frequently, but it is part of semantic HTML and can be useful in specific contexts, such as highlighting information related to the main content, a sidebar with links, or citations.

▼ **Here’s how to structure this in code**

```
<header>
  <h1>My Website</h1>
</header>
<nav>
  <ul>
   <li><a href="#home">Home</a></li>
   <li><a href="#about">About</a></li>
  </ul>
</nav>
<main>
  <p>Main content of the page.</p>
</main>
<footer>
  <p>&copy; 2024 My Website</p> </footer>
```

Grouping with

The ** ** tag is one of the most versatile in HTML, primarily used for grouping elements and creating sections on a web page. Although it does not add style or behavior on its own, it serves as a container that can be styled with CSS and manipulated later.

How to Use

Grouping Elements: Group blocks of content, such as texts, images, and other elements.

Creating Layouts: Build structured layouts using techniques like **flexbox** and **grid**.

Applying Styles: Add classes or IDs to the ** ** to apply specific styles.

> ### Tips!
>
> **Organization:** Use ** ** to organize your code into logical sections, making maintenance and understanding easier.
>
> **Semantics:** When possible, use more semantic HTML5 tags, such as **<header>**, **<section>**, and **<footer>**, to make content more accessible and improve SEO (Search Engine Optimization).

Incorporating Images

Images enrich web content, making it more engaging and informative. Use the <img> tag to insert images into your pages.

Images not only enhance web content, making it more attractive and informative, but they can also be adjusted and optimized to improve user experience.

Understanding the Attributes

To insert an image, use the **src** (source) attribute to specify the image path. The **alt** (alternative text) attribute provides an alternative description for the image, which is important for accessibility.

▼ **Example:**

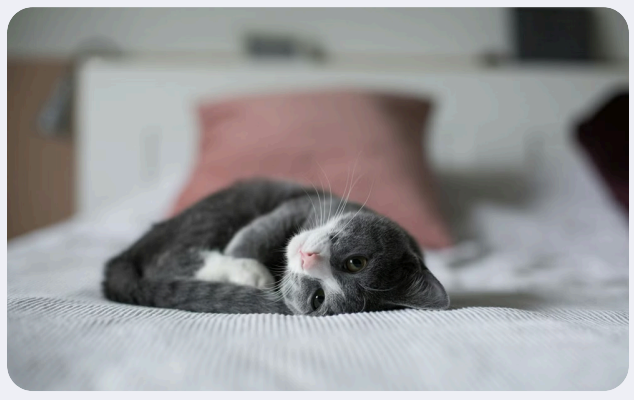
```
<img src="path/to/image.jpg"
alt="Image Description">
```

Adjusting Size

The size of the image can be controlled using the **width** and **height** attributes, adjusting the values as needed. It is considered best practice to separate style (CSS) from structure (HTML), as it makes the code more organized, easier to maintain, and allows for global adjustments more easily.

▼ **Example:**

```
.image {
max-width: 100%; height: auto;
}
```



Visual Elements

For vector images, consider using the SVG format. It is scalable and maintains quality at any size. Icons are also widely used to guide users intuitively. A common example is the Font Awesome library, which provides icons for social media, tools, actions, and more.

▼ **Example:**

```
<i class="fab fa-instagram"></i>
```

Lists, Tables, and Forms

Lists

Ordered Lists (<ol>): Create a numbered list.

Unordered Lists (<ul>): Create a list with bullet points.

List Items (<li>): Define the items within lists.

▼ **Example:**

```
<ul>
 <li>Item 1</li>
 <li>Item 2</li>
</ul>
```

Tables

Table Elements (<table>, <tr>, <td>, <th>): Create tables with rows and columns.

▼ **Example:**

```
<table>
 <tr>
  <th>Header 1</th>
  <th>Header 2</th>
 </tr>
 <tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
  </tr>
</table>
```

Forms

Form (<form>): Contains elements for data collection.

Input Fields (<input>, <textarea>, <select>): Used to receive information.

Buttons (<button>, <input type="submit">): To submit the form.

▼ **Example:**

```
<form action="/submit">
<label for="name">Name:
</label>

<input type="text"
id="name" name="name">

<input type="submit"
value="Send"> </form>
```

CSS: Styling Your Page

As we discussed earlier, CSS is the language used to style HTML elements. Now that we understand the basic structure of a page, let's explore various CSS properties to create visually appealing and attractive web pages.

How to apply CSS to an HTML Document

The two main ways to apply styles to your project are: Internal CSS and External CSS.

1. **Adding Internal CSS to HTML:** Directly in the HTML, use the **<style>** tag within the **<head>** section. This is useful for styles that are specific to a single page and do not need to be reused on other pages.
2. **Adding External CSS:** The CSS file needs to be linked to the HTML. This is done using the **<link rel="stylesheet" href="style.css" />** tag within the **<head>** section of your HTML document. The **href** attribute defines the path to the file, so it should match the name of your CSS file.

▼ **Note!**

> Both approaches have their uses and benefits, but generally, external CSS is considered best practice due to its ability to keep the code cleaner and more organized, facilitate maintenance, and improve the performance of your project.

Selectors: Classes and IDs

To apply styles efficiently, we use selectors that help identify which HTML elements should receive certain styles. Two of the most common selectors are classes and IDs. While both are used to apply styles, they have distinct purposes and rules of use:

Classes

Allow you to apply styles to multiple HTML elements at once. They are very useful when you want to style multiple elements similarly or apply a specific set of styles to different parts of the page.

- **Definition:** Classes are defined in CSS with a period (.) followed by the class name.
- **Usage in HTML:** To use a class in an HTML element, you should assign it to the **class** attribute of that element.

▼ **Example:**

In HTML:

```
<p class="description">This is an example of how to use classes in CSS to style different elements.</p>
<p class="description">Classes allow you to apply common styles to multiple elements.</p>
```

In CSS:

```
.description {
    color: #666;
    font-size: 1.2em;
    line-height: 1.6;
    margin-bottom: 15px;
}
```

IDs

Are used to identify a single element on the page. They are ideal when you need to apply styles to a specific element that does not repeat anywhere else on the page.

- **Definition:** IDs are defined in CSS with a hash (**#**) followed by the ID name.
- **Usage in HTML:** To use an ID, you should assign it to the **id** attribute of the HTML element.

▼ **Example:**

In HTML:

```
<header id="header">
    Welcome to My Website
</header>
```

In CSS:

```
#header {
    background-color: blue;
    color: white;
    padding: 20px;
    text-align: center;
}
```

Customizing Text and Colors

Text Options

Customize text with properties such as alignment, size, weight, font family, and decoration to improve readability and style.

▼ **Example:**

text-align: left | right | center | justify | initial | inherit;

Font-size: Sets the size of the font (px)

Font-weight: Defines the thickness of the font (numbers 100, 200 or normal | bold | bolder | lighter)

Font-family: Georgia, serif;

text-decoration: underline;

Color Variations

Explore different ways to define colors. You can apply colors to various elements such as backgrounds, text, icons, and even images.

▼ **Example:**

Background Color:

- Solid Color: `background: green;`
- Gradient: `background: linear-gradient(#e66465, #9198e5);`

Text Color:

- Named Color: `color: blueviolet;`
- Hexadecimal: `color: #00ff00;`
- RGB: `color: rgb(135, 93, 241);`
- RGBA (with transparency): `color: rgba(135, 93, 241, 0.5);`

```
<table>
 <tr>
  <th>Header 1</th>
  <th>Header 2</th>
 </tr>
 <tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
  </tr>
</table>
```

Fundamentals of CSS

Among the essential fundamentals of CSS are element layout and positioning, spacing techniques, dimension control, and manipulation of images and backgrounds. These concepts are important for creating responsive and organized designs, providing a solid foundation for those starting in web development.

Layout and Positioning

Flexbox

A simplified approach to creating layouts that adjust to the available space. It's great for creating one-dimensional layouts (row or column). Using properties like **flex-direction, justify-content,** and **align-items**, you can control the distribution and alignment of elements.

Grid

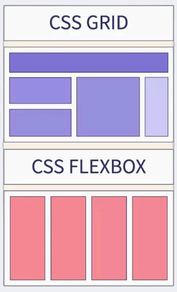
A two-dimensional system that allows for more complex layouts. It uses rows and columns to organize elements, providing precise control over where elements are positioned. Key properties include **grid-template-columns, grid-template-rows,** and **grid-area.**

Position

Element positioning: **static, relative, absolute, fixed,** and **sticky.**

Display

Element display types: **block, inline, inline-block, none**, etc.



Spacing

Margin

External space around an element, creating distance between adjacent elements.

Border

Adds borders around an element. Properties include **border-width, border-style,** e **border-color**.

Border-Radius

Rounds the corners of borders, allowing for elements with softened or circular corners.

Padding

Internal space around the content of an element, separating it from the element's own borders.

Dimensions

Width

Width: Defines the width of an element.

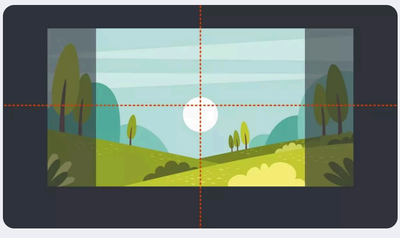
Max-width and Min-width: Limit the maximum or minimum width an element can have

Height

Height: Defines the height of an element.

Max-height and Min-height: Limit the maximum or minimum height an element can have.

Backgrounds and Objects



Background Image

Allows you to add a background image to any element. The **background-image** property sets the path to the image to be used.

Background properties

Background-Repeat: Controls if and how the background image is repeated (**repeat, no-repeat, repeat-x, repeat-y**).

Background-Position: Defines the initial position of the background image within the element.

Background-Size: Controls the size of the background image (**cover, contain**, or specific values in units).

Background-Attachment: Determines if the background should scroll with the content (**scroll**) or remain fixed (**fixed**).

Object-fit e Object-Position

Object-Fit: Adjusts how an image or video should be fitted inside the container (**fill, contain, cover, none, scale-down**).

Object-Position: Defines the position of the image or video within the container, especially when it is not fully covering the container.

JavaScript: Interactivity and Dynamism

JavaScript is the programming language that adds interactivity and dynamism to web pages. Understanding the basics allows you to create more engaging and interactive web experiences.

Introduction to JavaScript

Considered one of the core web technologies, JavaScript is used alongside HTML and CSS to create dynamic pages, including animations, user event responses, form validation, and much more.

How to include JavaScript in an HTML Page

To add JavaScript to a web page, you need to include it in your HTML file. This can be done in two main ways:

1. **Internal JavaScript:** Directly within the HTML file using the **<script>** tag. This code runs when the page loads.
2. **External JavaScript:** In a separate file with a .js extension, usually placed in a source subfolder. Include it in your HTML page using the **<script src="script.js"></script>** tag inside the **<body>**, after all the content.

How to Test?

In the **script.js** file, write console.log('Hello, World!');

▼ **Example:**

In HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>External JavaScript Example</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <script src="script.js"></script> <!-- Includes the external JavaScript file -->
</body>
</html>
```

In script.js

```
console.log('External JavaScript is working!');
```

Testing directly in the Browser

You can open the console by pressing **F12** or **Ctrl+Shift+I** (Windows/Linux) or **Cmd+Option+I** (Mac), then selecting the "Console" tab. Here, you can type JavaScript code and see the results instantly. It's a great way to experiment with small code snippets.

Basic JavaScript Concepts

1

Variables

Variables store data in JavaScript and can hold different types of data such as numbers, strings, booleans, objects, and arrays.

2

Operators

Operators are symbols that perform operations on variables and values, including arithmetic, comparison, and logical operations.

3

Functions

Functions are reusable blocks of code that perform a specific task. They help organize code and avoid repetition.

4

Conditional Structures

Conditional structures allow you to execute code based on conditions. **If / Else:** Executes a block of code if the condition is true or false.

5

Loops

Allow you to repeat an operation multiple times. **For Loop:** Repeats a block of code a specific number of times. **While Loop:** Repeats a block of code while a condition is true.

6

Selectors

Selectors are used to select HTML elements in JavaScript and manipulate them. For example, changing the text of an element.

7

Events

Events are actions that occur on a web page, such as clicking a button or moving the mouse. JavaScript allows you to respond to these events and create interactivity.

8

Console.log

This function displays a message in the browser's console. It's a useful debugging tool for checking variable values and the flow of code.

Applying the Concepts

In this section, we'll explore fundamental concepts that will bring your websites and applications to life, understanding how this language works and how to use it in your project.

Variables and Data Types

Variables

Variables store values that can be used and modified throughout the code. There are three keywords for declaring variables:

var: Declares a variable with either global or local scope, but it can cause confusion with its scope.

let: Declares a variable with block scope, which is safer and recommended.

const: Declares a variable that cannot be reassigned, meaning its value cannot change.

Data Types

Number: Numeric values, such as 42 or 3.14.

String: Represents text, such as "Hello, world!".

Boolean: Represents true or false values, such as true or false.

let age = 30; // Number

let name = "Ana"; // String

let isStudent = true; // Boolean

Operators

Arithmetic

let a = 5 + 3; // 8 let b = 10 % 3; // 1

Assignment

let x = 10; x += 5; // x é agora 15

Comparison

let isEqual = (5 === 5); // true

let isNotEqual = (5 !== 3); // true

Logical

let a = true && false; // false

let b = true || false; // true

Functions: Definition and How to Apply

Functions are blocks of code designed to perform specific tasks and can be reused in different parts of your program. They help organize code and avoid repetition.

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

Important to Know

Functions can accept parameters, which are values provided when the function is called, and return results, which are values produced after the function executes.

▼ **See here!**

```
function add(a, b) {
  return a + b;
}
console.log(add(5, 3)); // 8
```

More JavaScript Features

You will learn to use control structures to create dynamic logic, manipulate the DOM (Document Object Model) to transform and interact with your page's content, and work with events to make your applications interactive and responsive. Let's start turning your ideas into reality with the power of JavaScript!

Control Structures

Conditionals

if: Executes a block of code if the condition is true.

else: Executes a block of code if the **if** condition is false.

else if: Adds additional conditions.

switch: Replaces multiple **if** and **else** for value comparison.

▼ **Example:**

```
let score = 85;

if (score > 90) {
  console.log("Excellent");
} else if (score > 75) {
  console.log("Bom");
} else {
  console.log("Need to improve");
}
```

Loops

for: Iterates over a block of code a fixed number of times.

while: Iterates while the condition is true.

do...while: Iterates at least once, then continues while the condition is true.

▼ **Example:**

```
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}
let count = 0;
while (count < 5) {
  console.log(count); // 0, 1, 2, 3, 4
  count++;
}
```

DOM Manipulation

What is DOM?

The Document Object Model (DOM) is an interface that represents the structure of an HTML or XML document as a tree of objects. Each HTML element (like ** , <h1>, <p>,** etc.) is represented as a node in the DOM tree. This allows you, through JavaScript, to access, manipulate, and modify the content and structure of the web page dynamically.

Selecting and Modifying Elements

Example:

```
let header = document.querySelector("h1");
header.textContent = "New Title";
```

Explanation:

1. **document.querySelector("h1")**: This method is used to select the first <h1> element that appears in the document. **document** represents the whole page, and **querySelector** allows you to select elements using CSS selectors.
2. **header.textContent = "New Title";**: Here we are changing the text content of the **<h1>** element that was selected. The text within this element is replaced by "New Title". In other words, if the title was previously "Welcome", it will now be "New Title".

Events

Listening and responding to Events

In JavaScript, you can set up code to "listen" for events, such as clicks, key presses, mouse movements, and more. When the event occurs, a function is automatically executed in response.

Example:

```
document.querySelector("#button").addEventListener("click", function() { alert("Button clicked!");
});
```

Explanation:

1. **document.querySelector("#button")**: Here, we are selecting an element with the ID "button". The `#` in the selector indicates that we are targeting a specific ID.
2. **.addEventListener("click", function() {...})**: This method **addEventListener** is used to "listen" for a specific type of event on an element. In this case, we are listening for the "click" event (when the button is clicked).
3. **function() { alert("Button clicked!"); }**: This is the function that will be executed when the click event occurs. In this example, it shows an alert with the message "Button clicked!". You can replace this function with any action you want to occur when the button is clicked.

These are basic yet very powerful examples that will open the doors for you to create and control the behavior of your web pages!

link desconhecido

Congratulations on reaching this point!

Now that you've mastered the fundamentals of HTML, CSS, and JavaScript, you're ready to take the next steps and dive into more advanced concepts that will elevate your development skills.

Download

What’s Next...

The next module will help you create even more complex and interactive projects, which can open up new opportunities in your career.

Simplifying Programming - Intermediate

A bit about what you’ll learn!

- Deepen your knowledge of HTML, CSS, and JavaScript.
- Discover new elements and techniques to improve your website's accessibility.
- Explore Flexbox, Grid, animations, and how to create responsive layouts that work perfectly on any device.
- Learn how to manipulate APIs, use frameworks, and create more interactive and engaging user experiences.
- Build a real project from scratch to enhance your practical skills!

Keep practicing!

Want to continue!

If you enjoyed the content and found it helpful in simplifying programming, follow the next module guide and be a part of this project!