

How can JavaScript make a page interactive?

JavaScript is a programming language that runs in the browser. It can react to clicks, key presses and other events to make pages dynamic and interactive.

This page is for **absolute beginners** who already know a little HTML and CSS. First you build the structure with HTML, then you style it with CSS, and finally you use JavaScript to make the page respond to the user.

1. FROM STATIC TO DYNAMIC

JavaScript reacts to user actions

With only HTML and CSS, the page does not change by itself. With JavaScript you can respond when a user activates a button, types text, or uses a pointer or keyboard.

```
<p id="message" aria-live="polite"></p>
<button type="button" id="hello-button">Show message</button>
<script>
  const button = document.getElementById("hello-button");
  const message = document.getElementById("message");
  button.addEventListener("click", function () {
    message.textContent = "Hello from JavaScript!";
  });
</script>
```

Copy this example into the Sandbox, run the code and click the button to display a message without interrupting the visitor.

In small examples the `<script>` tag is often placed at the bottom of the `<body>`. In bigger projects, JavaScript is usually put in a separate file (for example `script.js`) and linked from the HTML page.

2. WORKING WITH THE PAGE (DOM)

JavaScript can change HTML and CSS

The browser represents the page as a tree of elements called the **DOM** (Document Object Model). JavaScript can find elements and change their text, classes or attributes.

```
<p id="status" aria-live="polite">Original text</p>
<button type="button" id="change-button">Change text</button>
<script>
  const changeButton = document.getElementById("change-button");
  changeButton.addEventListener("click", function () {
    const status = document.getElementById("status");
    status.textContent = "The text is now changed!";
  });
</script>
```

Here JavaScript looks up the element with `id="status"` and updates its text when the button is activated. Use `classList` when JavaScript needs to change appearance, so the visual rules remain in CSS.

3. ACCESSIBLE FORM INPUT

Use a form instead of prompt()

A visible, labeled form can be reviewed and corrected without blocking the rest of the page. Display its result in the document.

```
<form id="name-form">
  <label for="name">Your name:</label>
  <input id="name" name="name" required>
  <button type="submit">Show greeting</button>
</form>
<p id="greeting" aria-live="polite"></p>
<script>
  const form = document.getElementById("name-form");
  form.addEventListener("submit", function (event) {
    event.preventDefault();
    const name = document.getElementById("name").value.trim();
    document.getElementById("greeting").textContent =
      "Hello, " + name + "!";
  });
</script>
```

4. PRACTICE

What can you try yourself?

- Change the message text to your own sentence in the example above.
- Add a second button that toggles a high-contrast CSS class with `classList`.
- Create a five-second timer that updates an `aria-live="polite"` status message.

Start with small scripts, test them in the Sandbox and slowly add more ideas as you get comfortable.

5. WHAT TO DO NEXT?

Continue your JavaScript learning path

After you understand the basic idea of JavaScript, continue with these steps:

- **Practise more JavaScript:** go to the [JavaScript codes page](#) and try the ready-made examples in the Sandbox.
- **Combine HTML, CSS and JavaScript:** build a simple page with all three and make buttons, counters or small interactive elements.
- **Learn more from the blog:** read posts on blog.playcode123.com about functions, event listeners and external JavaScript files.
- **Practise with assignments:** open the [HTML, CSS and JavaScript exercises \(PDF\)](#) and try to create small interactive pages.

Remember: you do not need to understand everything at once. Take small steps, test your code in the Sandbox and learn by playing.