Cábula de JavaScript · Páginas web com vida

Holger Sebastião · www.sebastiao.org
Versão 1.0 · 2026-10-25

Incluir JavaScript — externo

```
<head>
  <link rel="stylesheet"
        href="css/style.css">
  <script src="js/script.js" defer></script>
</head>
```

Como no CSS: um ficheiro próprio na pasta `js`. `defer` significa que o script só corre quando a página inteira já carregou. Caso contrário, ainda não encontra os elementos.

```
website/
├── index.html
├── css/style.css
└── js/script.js
```

Saída e consola

```
console.log("Olá mundo");  // consola
console.log("Soma:", soma);
alert("Atenção!");         // aviso
```

Abrir a **consola**: **F12** ou Cmd+Alt+I → separador **Consola**. Aí aparecem também todos os erros a vermelho, com ficheiro e linha.

O que muda em relação ao Java?

JAVA	JAVASCRIPT
`int numero = 5;`	`let numero = 5;` sem tipo
`final`	`const`
`System.out.println`	`console.log`
`public int soma(…)`	`function soma(…)`
`==` com números	`===` sempre
`a.equals(b)`	`a === b` também para texto
`int[]` tamanho fixo	o array cresce com `push`
`7 / 2` → `3`	`7 / 2` → `3.5`

Igual ao Java: `if`, `else`, `for`, `while`, `switch`, `{ }`, `;`, `//`, `&&`, `||`, `!`. Maiúsculas e minúsculas contam.

Cálculos — Math

`Math.round(2.5)`	arredondar → `3`
`Math.floor(2.9)`	arredondar para baixo → `2`
`Math.random()`	aleatório de 0 até menos de 1
`10 % 3`	resto → `1`

Variáveis — let · const

```
let pontos = 12;        // pode mudar
pontos = pontos + 1;
const MAX_PONTOS = 15;  // fixo
const nome = "Ali";
```

`let` para valores que mudam, `const` para valores fixos. O JavaScript guarda o tipo sozinho. Programas antigos usam `var`: não usar mais.

Tipos de dados

TIPO	EXEMPLO
`number`	`42`, `3.5` (inteiros e decimais)
`string`	`"Ali"` ou `'Ali'`
`boolean`	`true`, `false`
`undefined`	variável sem valor
`null`	«nada» de propósito
Array	`[3, 8, 12]`
Objeto	`{ nome: "Ali", idade: 17 }`

`typeof pontos` → `"number"`

Texto — string

```
let nome = "Ali";
let saudacao = "Olá " + nome + "!";
let saudacao2 = `Olá ${nome}!`;  // igual
```

`saudacao.length`	número de carateres
`saudacao.toUpperCase()`	tudo em maiúsculas
`saudacao.includes("Ali")`	contém? → `true`
`saudacao.trim()`	tira os espaços das pontas

Entre **acentos graves** `` `…` ``, `${ }` insere o valor (marcador).

Número ou texto?

```
"5" + 3           // "53"  texto!
Number("5") + 3   // 8
Number("abc")     // NaN: não é número
(7.456).toFixed(2)  // "7.46"
```

O que vem de formulários é **sempre texto**. Converte com `Number(…)` antes de calcular.

Decisão — if · else

```
if (pontos >= 13) {
  console.log("muito bom");
} else if (pontos >= 5) {
  console.log("aprovado");
} else {
  console.log("reprovado");
}
```

`=== !==`	igual, diferente (valor **e** tipo)
`< > <= >=`	menor, maior …
`&& || !`	e, ou, não

`==` converte: `"5" == 5` dá `true`. Por isso, usa sempre `===`.

Escolha múltipla — switch

```
switch (dia) {
  case "Sáb":
  case "Dom":
    console.log("Fim de semana");
    break;
  default:
    console.log("Escola");
}
```

Ciclos

CICLO DE CONTAGEM

```
for (let i = 1; i <= 10; i++) {
  console.log(i);
}
```

COM TESTE NO INÍCIO

```
let soma = 0;
while (soma < 100) {
  soma = soma + 7;
}
```

COM TESTE NO FIM

```
let dado;
do {
  dado = Math.floor(Math.random() * 6) + 1;
} while (dado !== 6);
```

Funções

```
function calcularNota(pontos) {
  let nota = (17 - pontos) / 3;
  return Math.round(nota * 10) / 10;
}

let minhaNota = calcularNota(11);  // 2
```

Sem tipo de retorno nem tipos de parâmetro. Sem `return`, a função devolve `undefined`.

PARA LER: FUNÇÃO SETA

```
const dobrar = (numero) => numero * 2;
```

Forma curta, aparece muito na internet. Nós escrevemos `function`.

Arrays

```
let notas = [12, 8, 15];
notas.push(10);          // juntar no fim
console.log(notas[0]);   // 12
console.log(notas.length); // 4

for (let i = 0; i < notas.length; i++) {
  console.log(notas[i]);
}
for (let nota of notas) {   // cada elemento
  console.log(nota);
}
```

`notas.pop()`	tirar o último
`notas.includes(8)`	está lá? → `true`
`notas.indexOf(15)`	posição → `2`, senão `-1`
`notas.join(", ")`	`"12, 8, 15, 10"`

Objetos

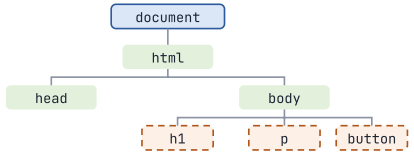
```
let aluno = {
  nome: "Ali",
  pontos: 12
};
console.log(aluno.nome);  // Ali
aluno.pontos = 13;
```

Cábula de JavaScript · Páginas web com vida

DOM · Alterar · Eventos · Formulários · Tempo · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.0 · 2026-10-25

O DOM — a página como árvore



O browser transforma o HTML numa árvore: o **DOM**. O JavaScript procura nela elementos e altera-os, e a página mostra logo a mudança.

Encontrar elementos

```
<h1 id="titulo">Calculadora de notas</h1>
<p class="aviso">…</p>
```

```
let titulo =
  document.getElementById("titulo");
let aviso =
  document.querySelector(".aviso");
```

`getElementById("titulo")`	o elemento com este id, **sem** `#`
`querySelector(".aviso")`	o **primeiro** que corresponde, com seletor CSS

! Se não encontrar nada, devolve `null`. Depois surge o erro `Cannot read properties of null`: verifica o id e usa `defer`.

Guardar no browser

```
localStorage.setItem("nome", "Ali");
let nome = localStorage.getItem("nome");
localStorage.removeItem("nome");
```

Mantém-se ao recarregar, mas só neste computador e só como texto. Não guardes nada secreto.

Elementos novos

```
let lista =
  document.getElementById("lista");
let item = document.createElement("li");
item.textContent = "Leite";
lista.append(item);   // juntar no fim
item.remove();        // voltar a apagar
```

Alterar o conteúdo

```
titulo.textContent = "Novo título";
aviso.innerHTML = "Agora <b>negrito</b>";
```

`textContent`	só texto, as tags aparecem como texto
`innerHTML`	texto com tags HTML
`value`	conteúdo de um campo de entrada

! Nunca mostres o que os utilizadores escrevem com `innerHTML`, usa `textContent`: caso contrário, alguém pode injetar HTML e scripts alheios.

Alterar o aspeto

```
/* style.css */
.erro {
  color: #b3261e;
  font-weight: bold;
}
```

```
aviso.classList.add("erro");
aviso.classList.remove("erro");
// liga se desligado, desliga se ligado:
aviso.classList.toggle("erro");
```

Melhor assim: o aspeto fica no CSS, o JavaScript só troca a classe.

```
titulo.style.color = "red";   // também dá
titulo.style.backgroundColor = "yellow";
```

Em `style`, `background-color` passa a `backgroundColor`.

Tratar muitos elementos

```
let ligacoes =
  document.querySelectorAll("nav a");
for (let ligacao of ligacoes) {
  ligacao.classList.add("menu");
}
```

`querySelectorAll` devolve uma lista. Com `for … of` trata-se cada elemento, um a um.

Reagir a cliques — eventos

```
<button id="botao">Clica aqui</button>
<p id="saida"></p>
```

```
let botao =
  document.getElementById("botao");
let saida =
  document.getElementById("saida");
let cliques = 0;

function contar() {
  cliques = cliques + 1;
  saida.textContent = cliques + " cliques";
}

botao.addEventListener("click", contar);
```

! Passa `contar` **sem** parênteses. Com `contar()` a função corre logo uma vez e nunca no clique.

EVENTO	ACONTECE QUANDO …
`"click"`	se clica
`"input"`	se escreve no campo
`"change"`	a escolha mudou
`"submit"`	o formulário é enviado
`"keydown"`	se carrega numa tecla
`"mouseover"`	o rato passa por cima

PARA LER: NO HTML

```
<button onclick="contar()">Clica</button>
```

Também funciona, mas mistura HTML e JavaScript. Nós separamos: `addEventListener`.

Calcular em direto — input

```
let campo =
  document.getElementById("pontos");

function mostrar() {
  let pontos = Number(campo.value);
  saida.textContent =
    "Nota: " + calcularNota(pontos);
}

campo.addEventListener("input", mostrar);
```

Recalculado a cada tecla, sem precisar de botão.

Tratar um formulário — calculadora de notas

```
<form id="calculadora">
  <label for="pontos">Pontos</label>
  <input type="number" id="pontos"
         min="0" max="15" required>
  <button type="submit">Calcular</button>
</form>
<p id="resultado"></p>
```

```
let calculadora =
  document.getElementById("calculadora");
function avaliar(evento) {
  evento.preventDefault();
  let campo =
    document.getElementById("pontos");
  let pontos = Number(campo.value);
  let nota = calcularNota(pontos);
  document.getElementById("resultado")
    .textContent = "Nota: " + nota;
}

calculadora.addEventListener("submit",
                             avaliar);
```

`preventDefault()` impede que o browser envie o formulário e recarregue a página: o campo e o resultado ficam visíveis.

Temporização

```
// uma vez após 2000 ms = 2 segundos
setTimeout(mostrarAviso, 2000);
// a cada segundo, até parar
let relogio = setInterval(tique, 1000);
clearInterval(relogio);
```

Também aqui: nome da função **sem** parênteses.

Erros típicos

- `… is not defined`: erro no nome ou nas maiúsculas/minúsculas.
- `"5" + 3` dá `"53"`: falta o `Number()`.
- `=` em vez de `===` no `if`: atribui em vez de comparar.
- Script no head sem `defer`: os elementos são `null`.
- Não acontece nada: vê a consola (F12) e mostra valores com `console.log`.
- Ponto de paragem: separador **Fontes** → clica no número da linha. O programa para aí.