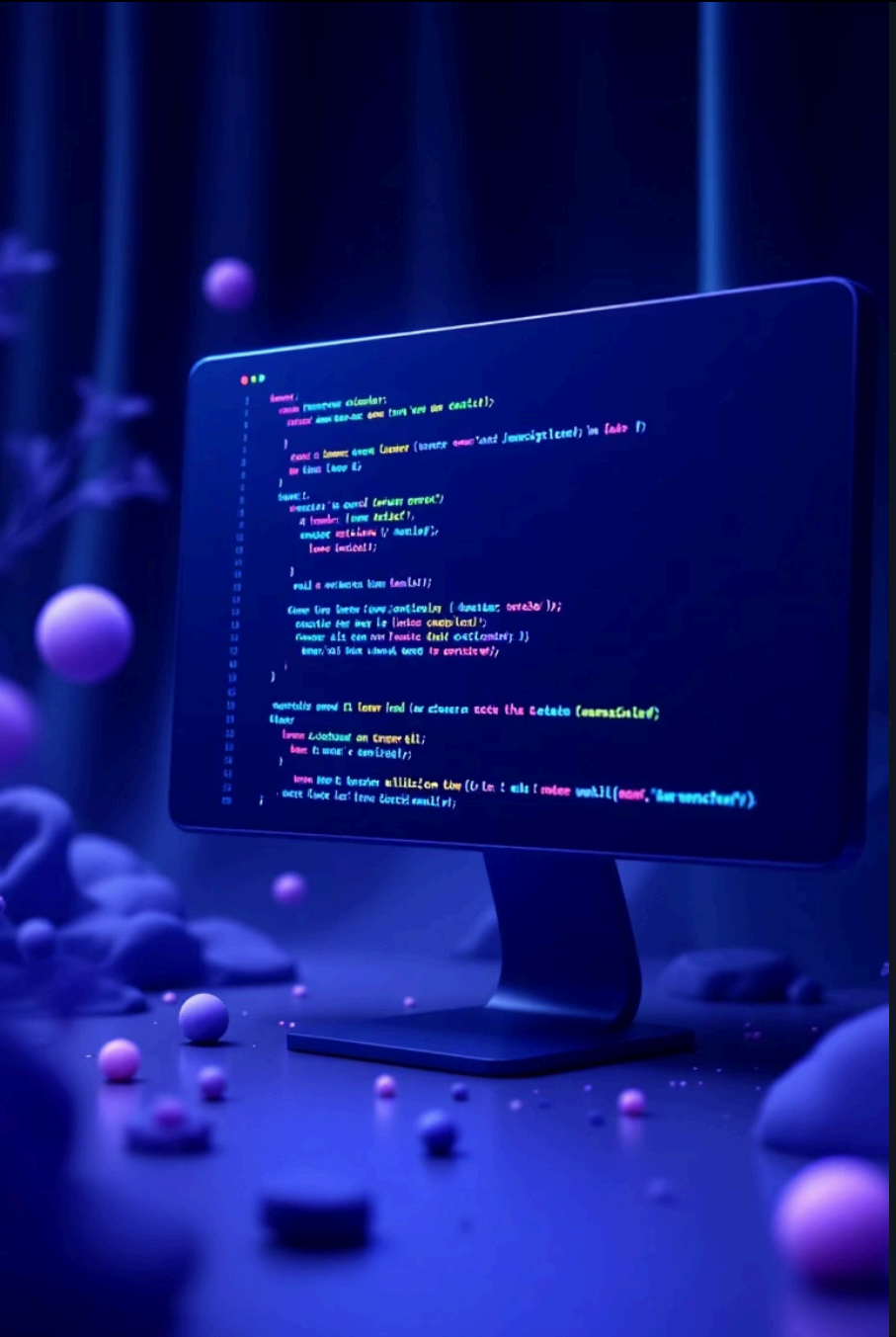


Linguagem de Programação JavaScript

01 - Introdução, fundamentos, front-end e elementos essenciais do front

A História do JavaScript

Origens (1995)

Criado por Brendan Eich na Netscape, o JavaScript passou por diferentes nomes: começou como Mocha, evoluiu para LiveScript e finalmente recebeu o nome JavaScript.

Em 1997, a linguagem foi padronizada como ECMAScript (ES), garantindo consistência entre diferentes navegadores.

Evolução Moderna

Hoje, JavaScript é a linguagem fundamental dos navegadores web. Com a criação do Node.js em 2009 por Ryan Dahl, expandiu-se para o ambiente servidor.

Esta evolução permitiu que desenvolvedores utilizassem a mesma linguagem tanto no cliente quanto no servidor.

JavaScript no Front-end

Responsabilidade Principal

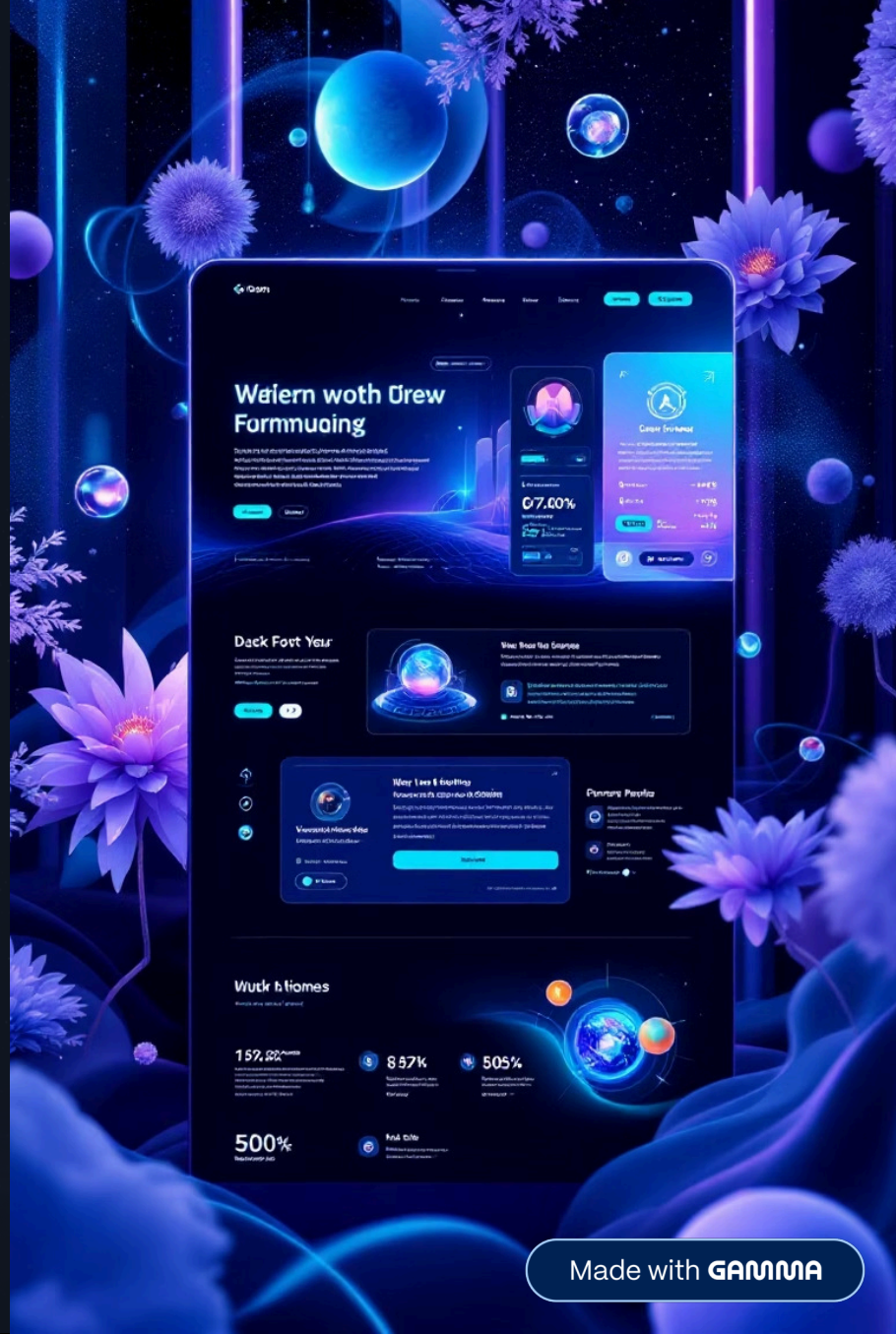
Cria a interatividade e a dinâmica das páginas web, incluindo reações a eventos do usuário, animações visuais e exibição dinâmica de dados.

Funcionamento

O código é executado diretamente no navegador do usuário, manipulando elementos HTML e estilos CSS para criar experiências interativas.

Ferramentas Modernas

Frameworks populares como React, Angular e Vue.js são utilizados para construir interfaces de usuário complexas e responsivas.



JavaScript no Back-end

Revolução Node.js

A criação do Node.js em 2009 popularizou o JavaScript no desenvolvimento servidor, permitindo execução fora do navegador.

Funcionalidades

Gerencia lógica do servidor, acesso a bancos de dados, autenticação de usuários e comunicação com o front-end.

Aplicações Práticas

Permite criar APIs, gerenciar dados e construir a lógica completa de aplicações web modernas.

Exemplo Prático de Aplicação

Estrutura HTML

```
<h1 id="titulo">Olá!</h1>
<button id="btn">
  Clique aqui
</button>
<script src="script.js">
</script>
```

Lógica JavaScript

```
console.log("Página carregada!");

document.getElementById("btn")
 .addEventListener("click", () => {
 const titulo =
 document.getElementById("titulo");
 titulo.textContent =
 "Você clicou no botão
```

Seleção e Busca de Elementos

01

document.getElementById()

Busca um único elemento através de seu identificador único.

```
const title = document.getElementById("titulo");
```

02

document.querySelector()

Retorna o primeiro elemento que corresponde a um seletor CSS.

```
const btn = document.querySelector(".btn.primary");
```

03

document.querySelectorAll()

Retorna uma lista com todos os elementos que correspondem ao seletor.

```
document.querySelectorAll("li")
 .forEach(li => li.classList.add("ok"));
```

Manipulação de Eventos e Estilos

addEventListener()

Registra eventos como clique, entrada de dados ou envio de formulário.

```
btn.addEventListener("click",
  () => console.log("clicou"));
```

classList

Gerencia classes CSS de forma eficiente sem manipular strings diretamente.

```
title.classList.toggle("ativo");
```

style

Aplica estilos inline através de JavaScript (use com moderação; prefira classes).

```
title.style.color = "tomato";
```

Modificação de Conteúdo

textContent

Insere texto puro sem interpretar marcação HTML, garantindo segurança.

```
title.textContent = "Bem-vindo!";
```

innerHTML

Permite inserir HTML interpretável. Atenção: cuidado com vulnerabilidades XSS.

```
const box = document.querySelector("#box");
box.innerHTML = "<strong>oi</strong>";
```

createElement()

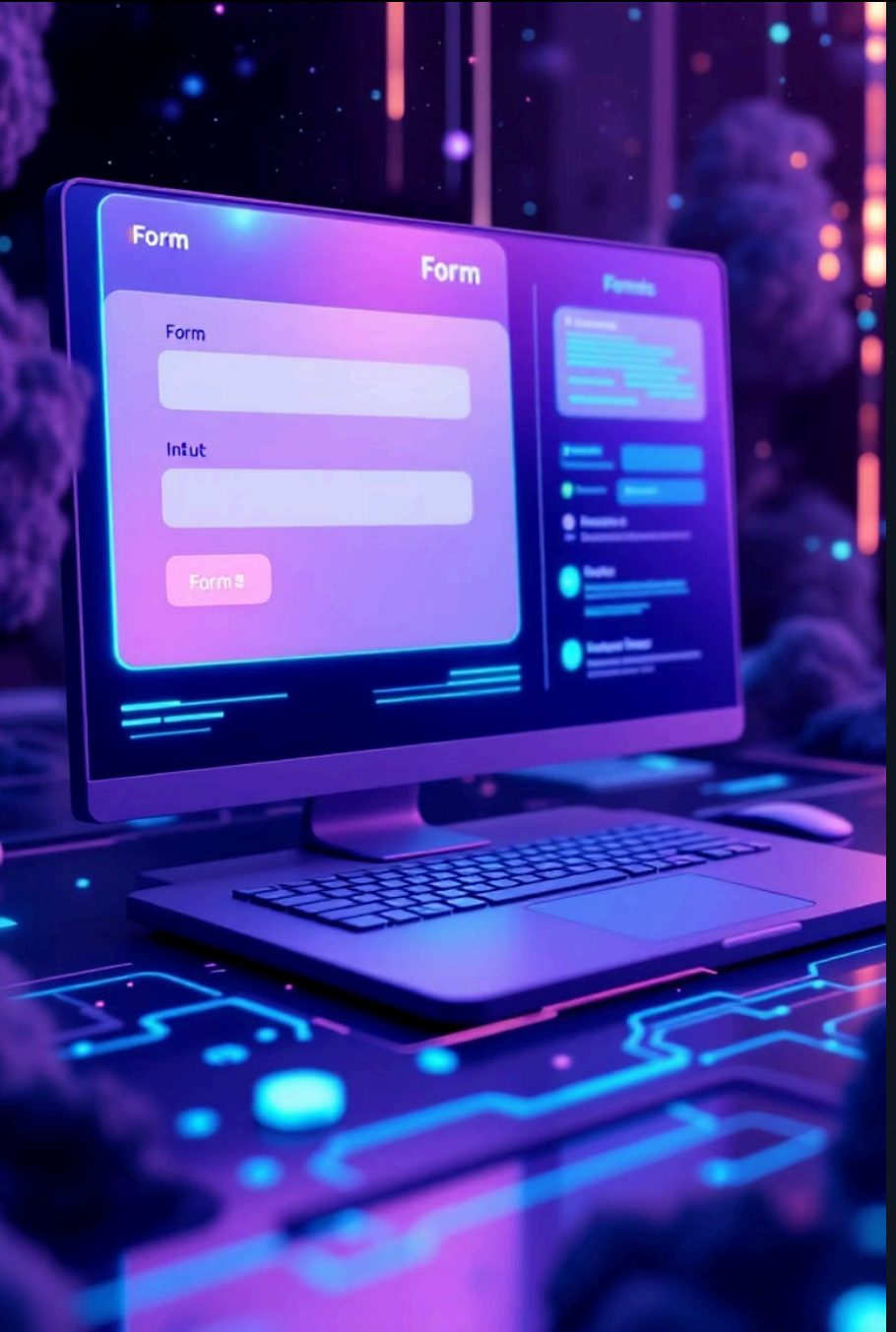
Cria novos elementos HTML dinamicamente através de JavaScript.

```
const li = document.createElement("li");
li.textContent = "Item";
```

Métodos de Inserção

append(), appendChild(), prepend(), before(), after() inserem elementos no DOM.

```
document.querySelector("ul").append(li);
```



Atributos, Formulários e Remoção

Atributos

getAttribute(), setAttribute() e dataset permitem ler e modificar atributos HTML.

```
btn.setAttribute("aria-pressed",
"true");
console.log(btn.dataset.userId);
```

Campos de Formulário

Propriedades value, checked e files permitem interação com elementos de entrada.

```
const nome =

document.querySelector("#nome")
.value;
const marcado =

document.querySelector("#ok").ch
ecked;
```

Remoção de Elementos

O método remove() elimina elementos do DOM de forma definitiva.

```
li.remove();
```

Eventos Avançados e Requisições

Controle de Eventos

event.target, currentTarget, preventDefault() e stopPropagation() controlam o fluxo e delegação de eventos.

Requisições Assíncronas

fetch, Promise, async/await e JSON permitem comunicação com APIs externas.

```
async function carregarTodo() {
  const r = await fetch("https://jsonplaceholder.typicode.com/todos/1");
  const data = await r.json();
  console.log(data.title);
}
carregarTodo();
```

Este exemplo demonstra como realizar requisições assíncronas para obter dados de uma API, processar a resposta em formato JSON e utilizar os dados na aplicação.