

Plano de Estudos Completo: HTML5 do Zero ao Expert

Módulos 12-14: Microdados & Formatos, Web Components, Offline & Cache

Professor: Sandro Pereira

2025



Módulo 12

Microdados & Formatos



Módulo 13

Web Components



Módulo 14

Offline & Cache



Duração

6 aulas





MÓDULO 12 – MICRODADOS & FORMATOS

Adicionando dados estruturados para melhorar SEO e interoperabilidade

Objetivo

Aprender a adicionar dados estruturados para melhorar a visibilidade nos motores de busca e interoperabilidade

Tópicos Principais

-  Microdados: itemscope, itemtype, itemprop
-  JSON-LD no script type="application/ld+json"
-  Schema.org para vocabulários padronizados
-  Exemplo: Evento "Web Summit" para Google Events

🕒 2 aulas

```
<html>
  <head>
    <title>What is Microdata?</title>
  </head>
  <body itemscope itemtype="https://schema.org/WebPage">
    <article itemscope itemtype="http://schema.org/Article" itemprop="mainEntity">
      <meta itemprop="url" content="https://brandvantage.co/blog/what-is-microdata/" />
      <meta itemprop="image" content="https://brandvantage.co/blog/2020/09/20th-of-september-2020/" />
      <h1 itemprop="name headline">What is Microdata?</h1>
      <time itemprop="datePublished" datetime="2020-09-20">20th of September 2020</time>
      <div itemprop="articleBody">
        Hello and welcome to this example!
      </div>
    </article>
  </body>
</html>
```

Evergreen



Dados estruturados ajudam os motores de busca a entender o conteúdo e podem resultar em rich snippets nos resultados de pesquisa

Dados Estruturados



Microdados

Atributos para marcar conteúdo diretamente no HTML

```
itemscope
itemtype="https://schema.org/Event">
  itemprop="name">Web Summit
  itemprop="startDate">2025-11-04
```



JSON-LD

Script com dados estruturados separados do conteúdo

```
type="application/ld+json">
{
  "@context":
"https://schema.org",
  "@type": "Event",
  "name": "Web Summit"
}
```



Schema.org

Vocabulários padronizados para diferentes tipos de conteúdo

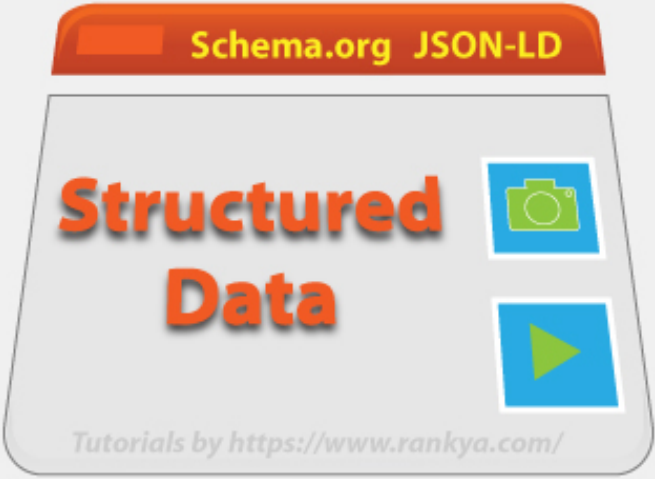
- Event
- Person
- Organization
- Product
- Recipe



Example 2

- Without Markup
- Microdata
- RDFa
- JSON-LD

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Restaurant",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Sunnyvale",
    "addressRegion": "CA",
    "postalCode": "94086",
    "streetAddress": "1901 Lemur Ave"
```



Dados estruturados permitem rich snippets nos resultados de busca, aumentando a visibilidade e taxa de cliques

MÓDULO 13 – WEB COMPONENTS (INTRO)

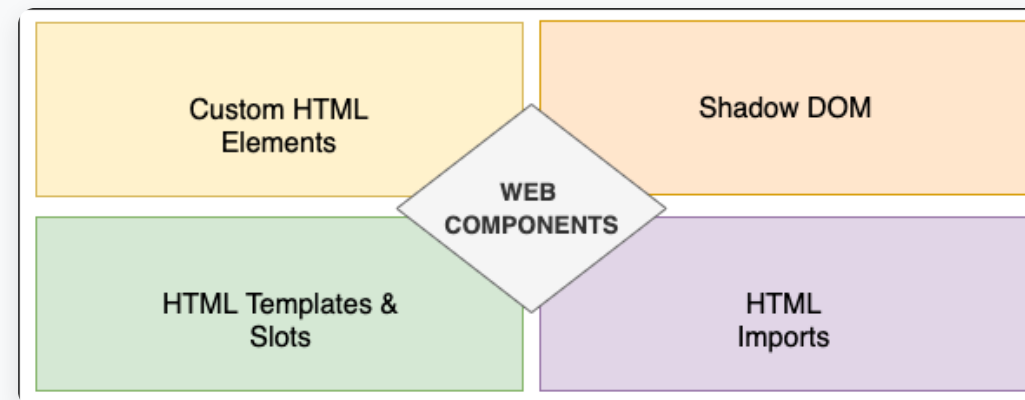
Criando componentes reutilizáveis com HTML, CSS e JavaScript

Objetivo

Aprender a criar componentes reutilizáveis e encapsulados que funcionam em qualquer framework

Tópicos Principais

- 🧩 Custom elements com nomes com hífen
- 📄 Template e slot para conteúdo reutilizável
- 📦 Shadow DOM (modo aberto) para encapsulamento
- 🔗 Exemplo prático: criar reutilizável



Web Components são padrões web nativos que permitem criar elementos personalizados reutilizáveis, independentes de frameworks

Componentes Reutilizáveis



Custom Elements

Elementos HTML personalizados com nomes contendo hífen

```
class CartaoPorto extends HTMLElement {  
  connectedCallback() {  
    this.innerHTML = `  
      Cartão do Porto  
    `;  
  }  
}  
  
customElements.define('cartao-porto',  
  CartaoPorto);
```



Template e Slot

Estruturas reutilizáveis com conteúdo dinâmico



Shadow DOM

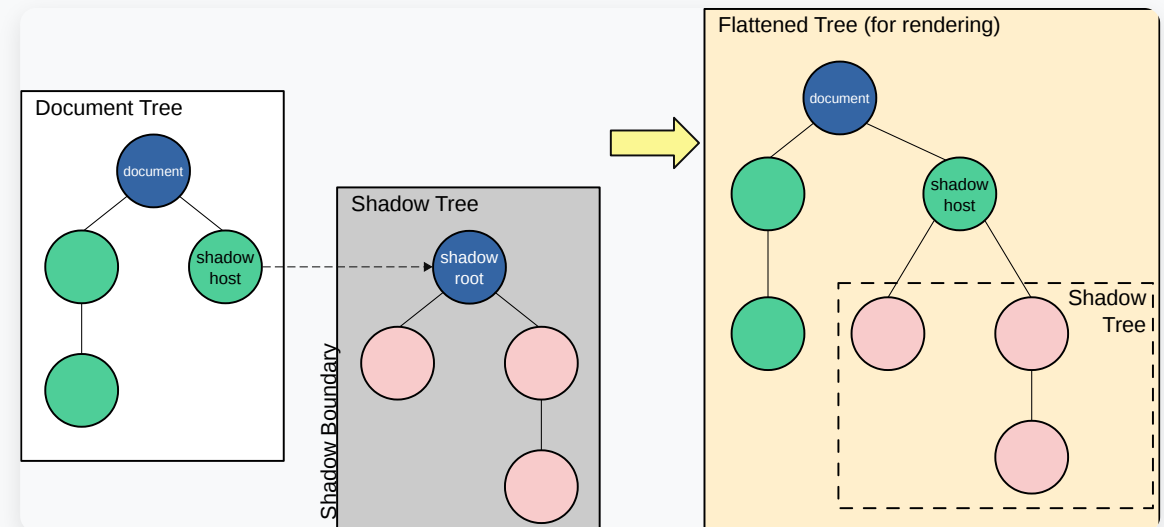
Encapsulamento de estilo e comportamento

```
const shadow = this.attachShadow({mode: 'open'});  
shadow.appendChild(template.content.cloneNode(true));
```



Exercício Prático

Criar componente reutilizável com template, slot e shadow DOM



```
<custom-message>  
  #shadow-root (open)  
    <style>...</style>  
    <slot name="emoji">  
      <span>  
    </slot>  
    <slot name="message" < slot>  
      <span> reveal  
    </slot>  
    <span slot="emoji">👋 </span>  
    <span slot="message">A very important message!</span>  
</custom-message>
```



Web Components funcionam em qualquer framework JavaScript e oferecem encapsulamento real de estilo e comportamento

MÓDULO 14 – OFFLINE & CACHE

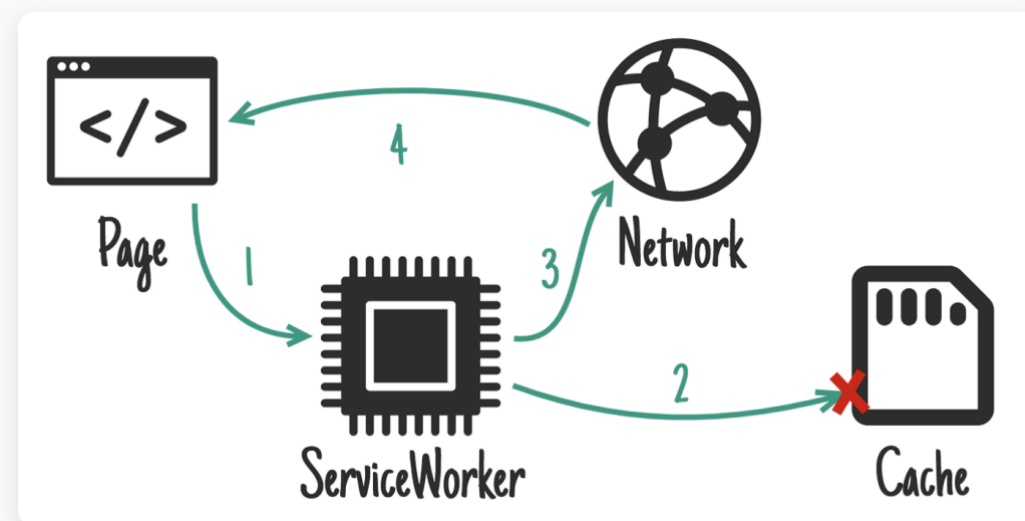
Criando aplicações que funcionam offline e otimizando cache

Objetivo

Aprender a criar aplicações que funcionam offline e otimizar o cache para melhor desempenho

Tópicos Principais

- ⚙️ Service Worker básico (registro)
- 📁 Cache API: `cache.addAll`
- 📄 Manifest.json: `name`, `icons`, `start_url`, `display`
- 📌 Instalação "Add to Home Screen"



Service Workers permitem interceptar requisições de rede e servir conteúdo do cache, essencial para PWAs e experiências offline

Aplicações Offline



Service Worker

Script que intercepta requisições de rede

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js')
    .then(reg => console.log('SW
registered'))
}
```



Cache API

Armazenamento de recursos para uso offline

```
caches.open('v1').then(cache => {
  cache.addAll(['/','/styles.css',
'/app.js'])
})
```



Manifest.json

Configuração da aplicação para instalação

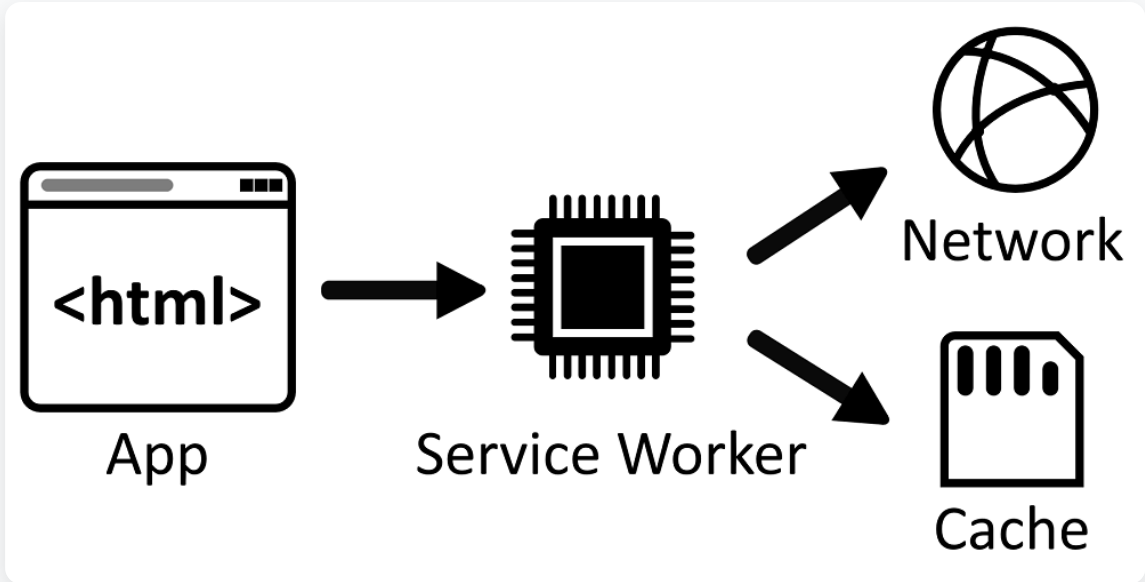
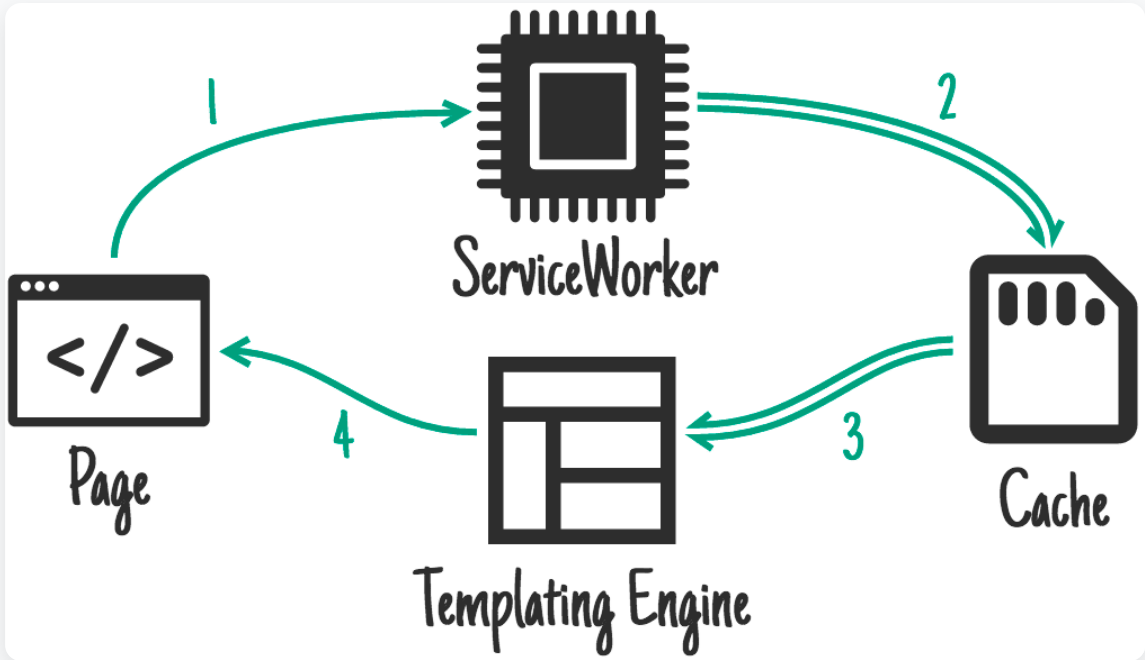
```
{
  "name": "Galeria Offline",
  "icons": [{...}],
  "start_url": "/",
  "display": "standalone"
}
```



Add to Home Screen

Prompt para instalação da aplicação

```
window.addEventListener('beforeinstallprompt',
e => {
  e.preventDefault();
  deferredPrompt = e;
});
```



Service Workers rodam em background separado da página, permitindo cache estratégico e funcionalidade offline mesmo quando a página está fechada



Exercício Prático

Criar galeria que funciona offline após a primeira visita

Objetivo

Criar galeria que funciona offline após a primeira visita

Passos para completar

1 Service Worker

Criar SW para interceptar requisições

2 Estratégia de Cache

Implementar **cache first** para imagens

3 Manifest.json

Configurar `name`, `icons`, `start_url`

4 Instalação

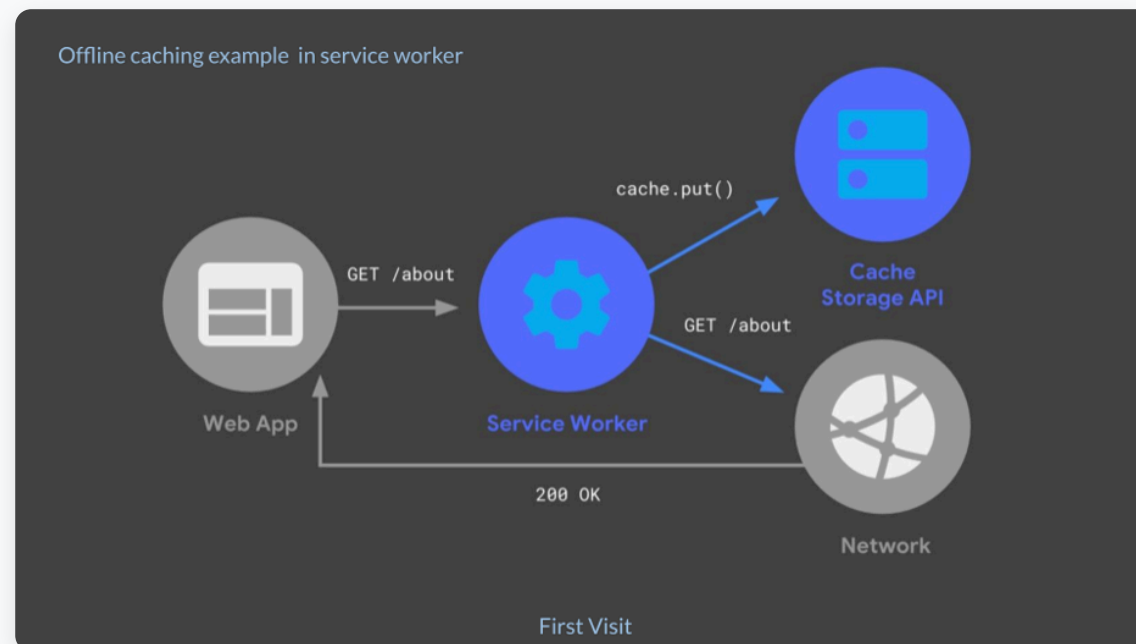
Adicionar prompt **Add to Home Screen**

5 Teste Offline

Desativar rede e validar funcionamento

6 DevTools

Validar com Chrome DevTools



Use a aba **Application** → **Service Workers** no Chrome DevTools para testar e depurar o comportamento offline

RESUMO DOS MÓDULOS 12-14



Módulo 12

- ✓ Microdados: itemscope, itemtype, itemprop
- ✓ JSON-LD para dados estruturados
- ✓ Schema.org para vocabulários padronizados
- ✓ Rich snippets nos resultados de busca



Módulo 13

- ✓ Custom elements com nomes com hífen
- ✓ Template e slot para conteúdo reutilizável
- ✓ Shadow DOM para encapsulamento
- ✓ Componentes independentes de framework



Módulo 14

- ✓ Service Worker para controle de rede
- ✓ Cache API para armazenamento offline
- ✓ Manifest.json para instalação PWA
- ✓ Add to Home Screen

→ Próximos Módulos



Módulo 15: HTML+CSS+JS



Módulo 16: Validação



Módulo 17: Boas Práticas