Gestão do Bloqueio de Renderização

Guia de referência para Async, Defer e Lazy Loading

O Que São Recursos de Bloqueio de Renderização?

Recursos que impedem a página de ser exibida até que sejam totalmente processados. O navegador não consegue pintar o conteúdo no ecrã enquanto estes ficheiros não forem resolvidos.

CSS Externo

O navegador precisa de carregar e processar o CSS antes de qualquer renderização.

Scripts Síncronos

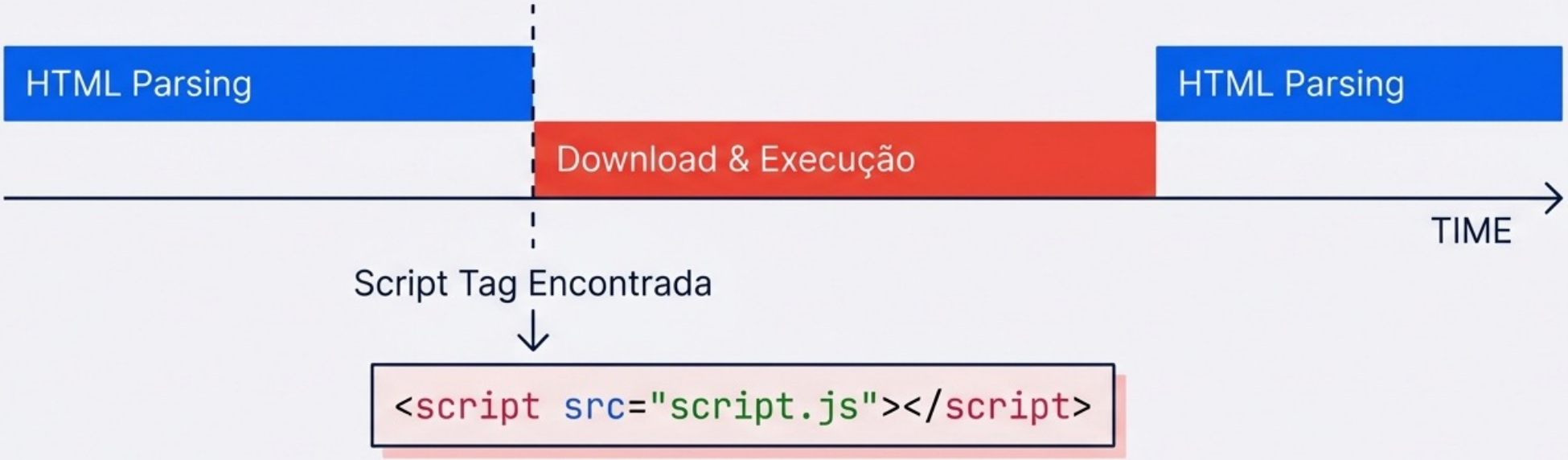
Bloqueiam o parsing do HTML até serem transferidos e executados.

Fontes Externas

Podem atrasar significativamente a renderização do texto visível.

O Comportamento Padrão dos Scripts

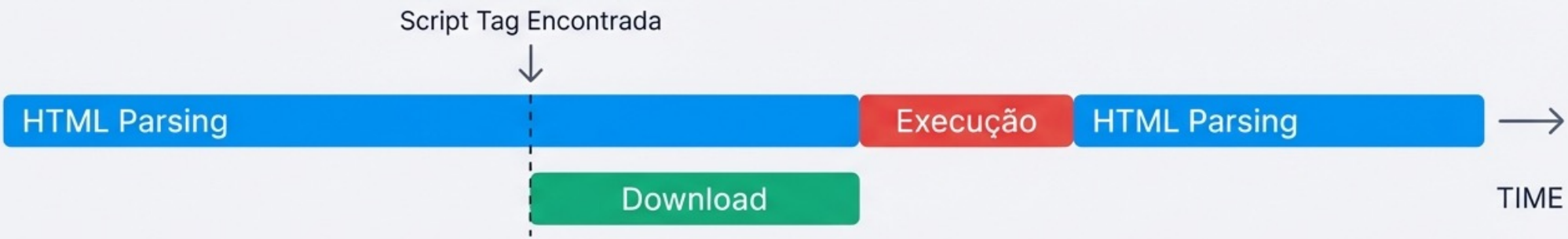
Sem atributos de otimização, o JavaScript é agressivo na sua prioridade.



Key Insight: Se o script for pesado ou a rede lenta, o utilizador vê um ecrã em branco durante todo este processo.

Modo Assíncrono (Async)

```
<script async src="script.js"></script>
```



- O script é transferido em paralelo ao HTML.
- Executa imediatamente assim que o download termina.
- O HTML é pausado apenas durante a execução.

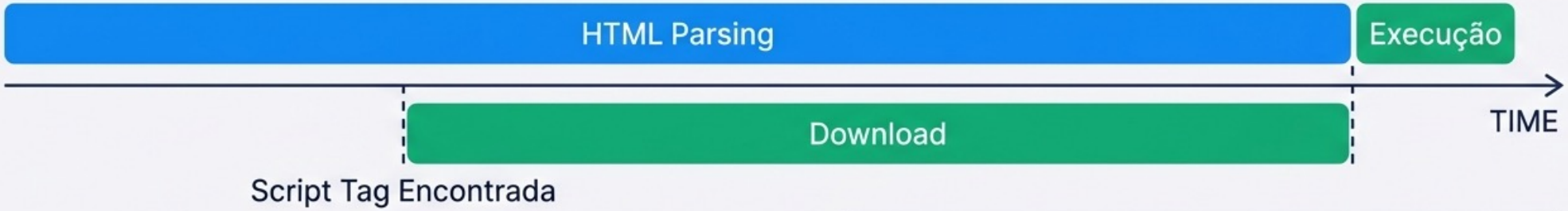
⚠ A ordem de execução não é garantida. O script que terminar o download primeiro executa primeiro.

🧠 Quando usar?

- Scripts independentes.
- Analytics e rastreadores.
- Qualquer código sem dependências.

Modo Diferido (Defer)

```
<script defer src="script.js"></script>
```



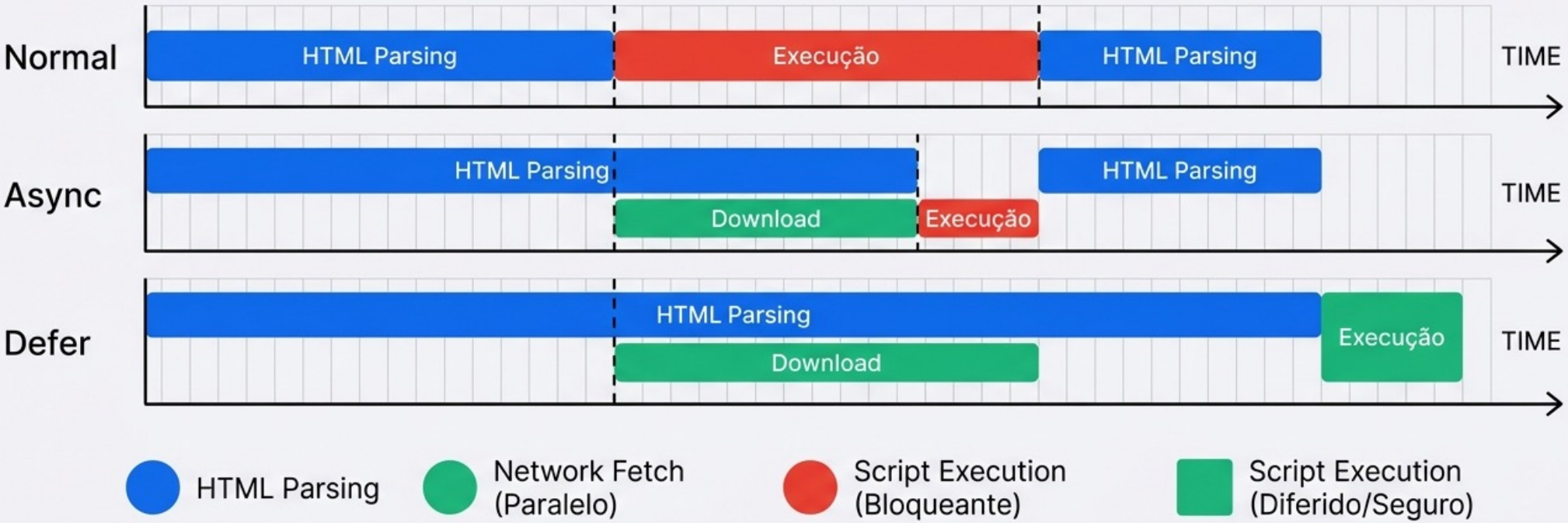
- Baixa o script em paralelo ao HTML.
- Executa apenas depois que o HTML estiver totalmente carregado.
- Mantém a ordem original dos scripts.

🧠 **Quando usar?**

- Scripts que manipulam o DOM.
- Scripts com dependências.
- Código principal da aplicação.

✅ **Melhor prática moderna para a maioria dos casos.**

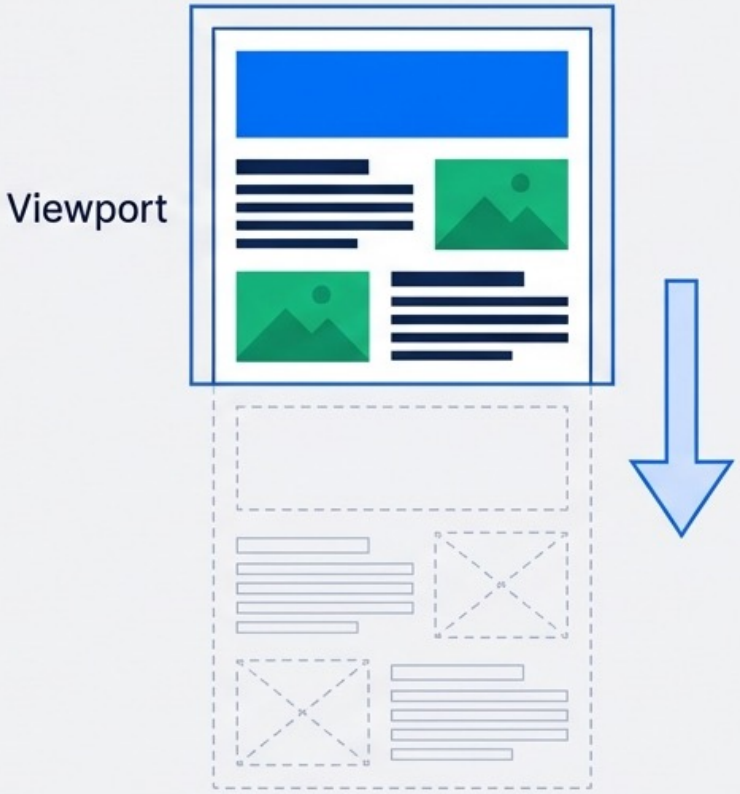
Comparação Visual da Linha de Tempo



✓ **O defer é o único que move a execução totalmente para fora do caminho crítico do parsing.**

Lazy Loading (Carregamento Preguiçoso)

Técnica onde os recursos só são **carregados quando** entram **na área visível do ecrã** (viewport).



🎯 Objetivo

- Melhorar o tempo de carregamento inicial.
- Reduzir o consumo de dados do utilizador.
- Melhorar métricas de desempenho (LCP).

Implementação de Lazy Loading

```
<img src="imagem.jpg" loading="lazy" alt="...">
```

O atributo diz ao navegador para adiar o download até que o elemento esteja próximo de aparecer no ecrã.

🔎 Pode ser usado para

Imagens (img)

Vídeos

Iframes

Componentes dinâmicos

💡 **Pro Tip** Não aplicar 'lazy' na imagem de destaque (LCP) no topo da página. Isso atrasa a renderização visual inicial.

Matriz de Decisão: Estratégias de Scripts

Estratégia	Quando carrega	Ordem garantida	Bloqueia renderização
Normal	Imediato	Sim	**SIM**
Async	Paralelo	Não	Durante execução
Defer	Paralelo	Sim	**NÃO**

Stratégia note:

- Use **Defer** por defeito.
- Use **Async** para exceções (analytics).
- Use **Normal** apenas se for estritamente necessário bloquear o parser.

Checklist: Como Evitar Bloqueios

1. **Scripts:** Usar `defer` para a lógica da aplicação e `async` para scripts de terceiros.
2. **Fallback:** Se não usar defer, colocar os scripts no final da tag `<body>`.
3. **Media:** Aplicar `loading="lazy"` em imagens e iframes abaixo da dobra.
4. **CSS:** Minificar ficheiros CSS.

4. **CSS:** Minificar ficheiros CSS.

5. **Avançado:** Usar `preload` ou `preconnect` para recursos críticos.
6. **Fontes:** Garantir carregamento otimizado para não bloquear o texto.

A **velocidade** de renderização é a primeira impressão do utilizador. Não o faça esperar.