

MIT em Desenvolvimento Full Stack

Front-end Jamstack com Gatsby

Agenda

Aula 6: Otimização de Carregamento de Imagens e Melhoria de Desempenho em Páginas Gatsby.

- Recapitulando.
- Inclusão de Imagens.
- Auditoria de Desempenho.



The background is a complex, low-poly geometric pattern composed of numerous triangles. The color palette is a gradient of greens and blues, ranging from very dark, almost black, in the bottom-left corner to a bright, vibrant green in the top-right corner. The triangles vary in size and orientation, creating a textured, crystalline effect.

Recapitulando

Your Filesystem

CMS

Private API

Database

source
plugins

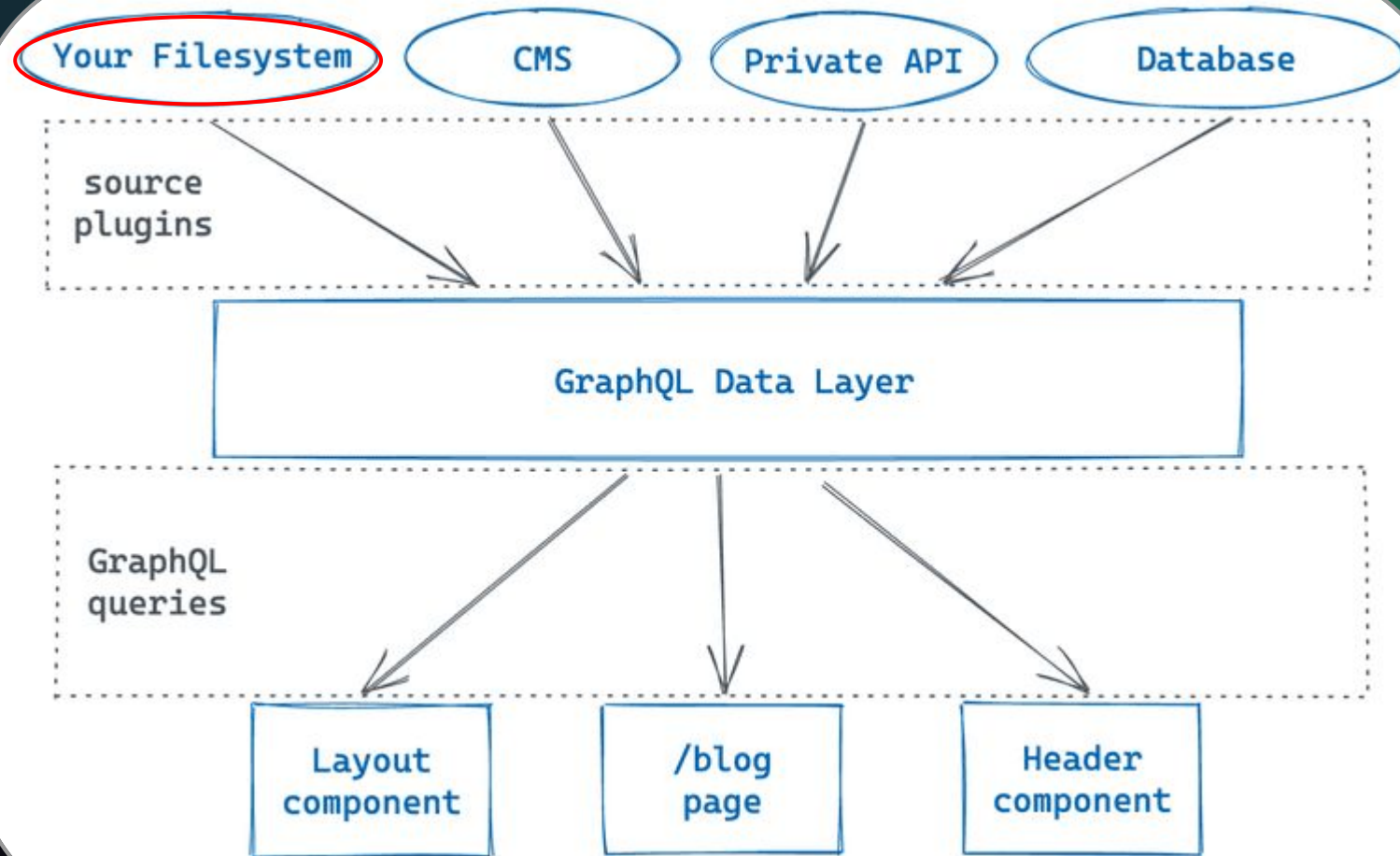
GraphQL Data Layer

GraphQL
queries

Layout
component

/blog
page

Header
component



JS index.js x

src > pages > blog > JS index.js > [e] BlogPage

```
23   return (  
24     <Layout>  
25       <p>Esses são os últimos posts:</p>  
26       {  
27         data.allMdx.nodes.map(node => (  
28           <article key={node.id}>  
29             <h2>  
30               <Link to={`/blog/${node.frontmatter.slug}`}>  
31                 {node.frontmatter.title}  
32               </Link>  
33             </h2>  
34             <p>Posted: {node.frontmatter.date}</p>  
35           </article>  
36         ))  
37       }  
38     </Layout>  
39   )  
40 }  
41  
42 export const Head = () => <title>Posts</title>  
43  
44 export default BlogPage
```

Esta é uma simples lista de arquivos obtida pelo GraphQL + gatsby-source-filesystem

JS index.js x

src > pages > blog > JS index.js > [🔗] BlogPage

```
1  import * as React from 'react'
2  import { Link, useStaticQuery, graphql } from 'gatsby'
3  import Layout from '../components/layout'
4
5  const BlogPage = () => {
6
7    const data = useStaticQuery(graphql`
8      query {
9        allMdx(sort: { frontmatter: { date: DESC } }) {
10          nodes {
11            frontmatter {
12              date(formatString: "MMMM D, YYYY")
13              title
14              slug
15            }
16            id
17            excerpt
18          }
19        }
20      }
21    `)
```

Esta é a consulta do GraphQL que alimenta a lista de itens da página

```
> .cache
> blog
  ≡ esse-novo-post.mdx
  ≡ meu-primeiro-post.mdx
  ≡ sim-mais-um-post.mdx
> node_modules
> public
> src
  > components
    JS footer.js
    # footer.module.css
    # layout.css
    JS layout.js
  > images
  > pages
    > blog
      JS {mdx.frontmatter_slug}.js
      JS index.js
      JS 404.js
      JS index.js
    .gitignore
  JS gatsby-config.js
  {} package-lock.json
  {} package.json
  ⓘ README.md
```

JS {mdx.frontmatter_slug}.js

src > pages > blog > JS {mdx.frontmatter_slug}.js > ...

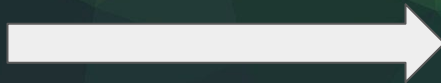
```
1 import * as React from 'react'
2 import { graphql } from 'gatsby'
3 import Layout from '../components/layout'
4
5 const BlogPost = ({ data, children }) => {
6   return (
7     <Layout pageTitle={data.mdx.frontmatter.title}>
8       <p>{data.mdx.frontmatter.date}</p>
9       {children}
10     </Layout>
11   )
12 }
13
14 export const query = graphql`
15   query ($id: String) {
16     mdx(id: {eq: $id}) {
17       frontmatter {
18         title
19         date(formatString: "MMMM D, YYYY")
20       }
21     }
22   }
23 `
24
25 export const Head = ({ data }) => <title>{data.mdx
26
27 export default BlogPost
```

Este é o template de arquivos MDX que vai gerar rotas dinâmicas por causa do nome desse arquivo entre {}

```
JS index.js U x
src > pages > blog > JS index.js > [e] BlogPage
7   const data = useStaticQuery(graphql`
8     query {
9       allMdx(sort: { frontmatter: { date: DESC } }) {
10         nodes {
11           frontmatter {
12             date(formatString: "DD/MM/YYYY")
13             title
14             slug
15           }
16           id
17           excerpt
18         }
19       }
20     }
21 `)
```

BlogPage

Lista de arquivos obtida pelos nomes dos arquivos que coincidem com o slug de cada post.



```
JS {mdx.frontmatter_slug}.js U x
src > pages > blog > JS {mdx.frontmatter_slug}.js > [e] BlogPost
14  export const query = graphql`
15    query ($id: String) {
16      mdx(id: {eq: $id}) {
17        frontmatter {
18          title
19          date(formatString: "MMMM D, YYYY")
20        }
21      }
22    }
23 `
```

BlogPost

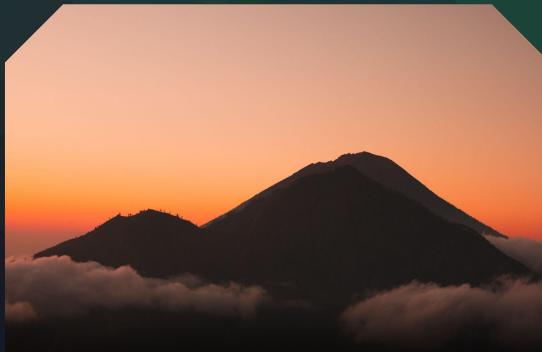
Template de MDX que pega os dados do GraphQL que consulta o front matter de cada arquivo.

Inclusão de Imagens

StaticImage

É para fontes de imagens estáticas ,
como um caminho de arquivo
codificado ou URL remoto.

Em outras palavras, a fonte da sua
imagem sempre será a mesma toda
vez que o componente for
renderizado..



GatsbyImage

Este componente é para fontes de
imagem dinâmicas , como se a fonte
da imagem fosse passada como um
argumento de prop.



Q Pesquise imagens de alta resolução

Explorar

Anunciar

Unsplash+

Entrar

Enviar uma foto



Editorial

Unsplash+

Em destaque

Momentos Aconchegantes

Wallpapers

Renderizações 3D

Arquitetura E Interiores

Natureza

Texturas E Padrões

Fotografia De Rua

Unsplash

Fonte de recursos visuais da internet.

Fornecidos por criadores de todo o mundo.

Tendência: flor, fundos de tela, planos de fundo, feliz, amor

Apoiado por SQUARESPACE

Q Pesquise imagens de alta resolução

Sim, é realmente grátis.

Todas as imagens podem ser baixadas e usadas para projetos pessoais e comerciais.

[Saiba mais sobre nossa licença](#)

Coleções de tendência

Ver tudo



Street Photography

Selecionadas por Aedrian



Warm and Muted



▼ JAMSTACK_007

> .cache

▼ blog

▼ esse-novo-post

≡ index.mdx

mario-von-rotz-0gfdSWav8RQ-unsplash.jpg

▼ meu-primeiro-post

≡ index.mdx

job-savelsberg-yjqVLkuBR_Q-unsplash.jpg

▼ sim-mais-um-post

≡ index.mdx

karsten-winegeart-ymsMuL2y6yk-unsplash.jpg





Dica

Às vezes, as imagens que você baixa da internet podem ter qualidade um pouco alta demais.



Se você sabe que seu site renderizará apenas uma imagem com 1.000 pixels de largura, não faz sentido ter uma imagem de origem com 5.000 pixels de largura.

Todos esses pixels extras significam trabalho extra para processar e otimizar suas imagens, o que pode retardar o tempo de construção.

Como orientação geral, é uma boa ideia pré-otimizar seus arquivos de imagem redimensionando-os para que não tenham mais do que o dobro do tamanho máximo em que serão renderizados.

Por exemplo, se o seu layout tiver 600 pixels de largura, a imagem de resolução mais alta necessária será de 1200 pixels (para compensar 2x a densidade de pixels).

3

 index.mdx M Xblog > esse-novo-post >  index.mdx

```
1  ---
2  title: "Meu Outro Post"
3  date: "2023-09-15"
4  slug: "meu-outro-post"
5  hero_image: "./mario-von-rotz-0gfdSWav8RQ-unsplash.jpg"
6  hero_image_alt: "Mount Agung, Jungutan, Karangasem Regency, Bali, Indonesia"
7  hero_image_credit_text: "Mario von Rotz"
8  hero_image_credit_link: "https://unsplash.com/pt-br/fotografias/uma-montanha-coberta-de-nu"
9  ---
10
11  Esse é o meu outro post.
```



```
npm install gatsby-plugin-image gatsby-plugin-sharp gatsby-transformer-sharp
```

4

JS gatsby-config.js x

JS gatsby-config.js > ...

```
1  module.exports = {  
2    siteMetadata: {  
3      title: `Estudos do Jamstack com Gatsby`,  
4    },  
5    plugins: [  
6      {  
7        resolve: "gatsby-source-filesystem",  
8        options: {  
9          name: `blog`,  
10         path: `${__dirname}/blog`,  
11       },  
12     },  
13     "gatsby-plugin-mdx",  
14     "gatsby-plugin-image",  
15     "gatsby-plugin-sharp",  
16     "gatsby-transformer-sharp",  
17   ],  
18 }
```

5

Local Filesystem

gatsby-source-filesystem

GRAPHQL DATA LAYER

File nodes



gatsby-transformer-sharp



gatsby-plugin-mdx

MDX nodes



GraphQL query

BlogPost
PageTemplate
<GatsbyImage />



```
1 query MyQuery {  
2   allMdx(sort: {frontmatter: {date: DESC}}) {  
3     nodes {  
4       frontmatter {  
5         date(formatString: "DD/MMM/YYYY")  
6         title  
7         hero_image_alt  
8         hero_image_credit_link  
9         hero_image_credit_text  
10        hero_image_alt  
11        hero_image {  
12          childImageSharp {  
13            gatsbyImageData  
14          }  
15        }  
16      }  
17    }  
18  }  
19 }  
20 }
```



6

Variables Headers

+ GraphQL

```
{  
  "data": {  
    "allMdx": {  
      "nodes": [  
        {  
          "frontmatter": {  
            "date": "15/Oct/2023",  
            "title": "Mais um Post",  
            "hero_image_alt": "Diamond Beach,  
Iceland",  
            "hero_image_credit_link":  
"https://unsplash.com/pt-br/fotografias/um-iceberg-  
no-meio-de-um-corpo-de-agua-ymsMuL2y6yk?  
utm_content=creditShareLink&utm_medium=referral&utm_  
source=unsplash",  
            "hero_image_credit_text": "Karsten  
Winegeart",  
            "hero_image": {  
              "childImageSharp": {  
                "gatsbyImageData": {  
                  "layout": "constrained",  
                  "backgroundColor": "#c8c8c8",  
                  "images": {  
                    "fallback": {  
                      "src":  
"/static/246566921a61699f99928521eafc3e4/815fa/kars  
ten-winegeart-ymsMuL2y6yk-unsplash.jpg",  
                      "srcSet":
```



Dica

Como o **GraphQL** sabe como adicionar campos extras ao campo **hero_image**?

Quando **Gatsby** constrói seu site, ele cria um esquema **GraphQL** que descreve os diferentes tipos de dados na camada de dados.

À medida que o **Gatsby** constrói esse esquema, tenta “adivinhar” o tipo de dados de cada campo. Este processo é chamado de **inferência de esquema**.

Gatsby pode dizer que o campo **hero_image** do seu **MDX** corresponde a um **File** e portanto permite consultar os campos **File** desse nó.

Da mesma forma, o **gatsby-transformer-sharp** pode dizer que o arquivo é uma imagem, portanto também permite consultar os campos **ImageSharp** desse nó.

JS {mdx.frontmatter_slug}.js X

src > pages > blog > JS {mdx.frontmatter_slug}.js > [?] query

```
22 export const query = graphql`
23   query ($id: String!) {
24     mdx(id: {eq: $id}) {
25       frontmatter {
26         title
27         date(formatString: "MMMM D, YYYY")
28         hero_image_alt
29         hero_image_credit_link
30         hero_image_credit_text
31         hero_image {
32           childImageSharp {
33             gatsbyImageData(
34               width: 600
35               placeholder: BLURRED
36               formats: [AUTO, WEBP, AVIF]
37             )
38           }
39         }
40       }
41     }
42   }
43`
```

Webp

É um formato de imagem digital aberto do tipo raster, derivado do formato de vídeo VP8 e projeto irmão do formato WebM.

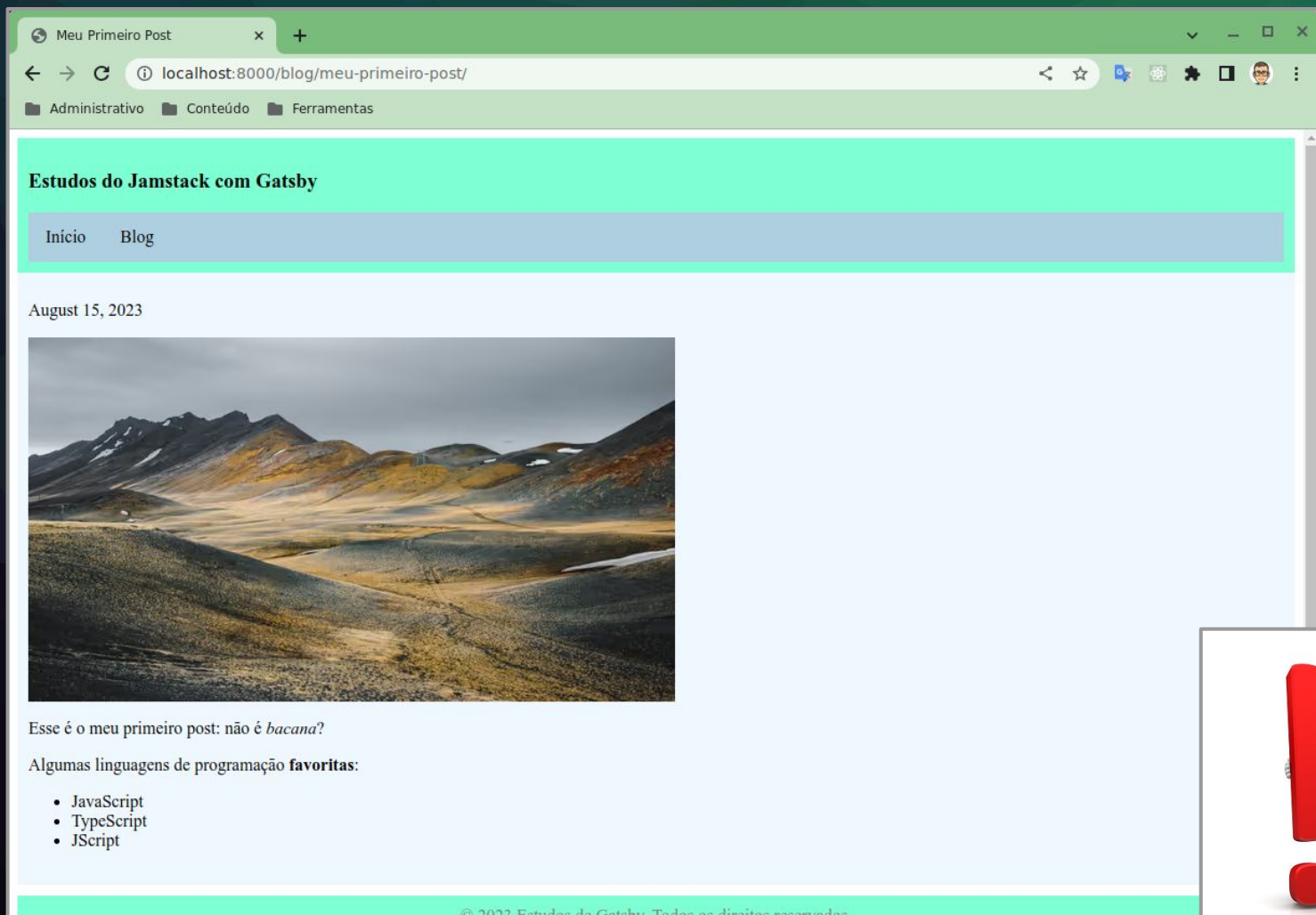
Desenvolvido pela Google, com o objetivo de diminuir o tamanho dos arquivos, garantindo uma transferência mais rápida, com qualidade.

Une o que há de melhor em outros formatos de imagem, como a possibilidade de compressão do arquivo, a capacidade de usar transparência, e o suporte a animações.

8

`JS {mdx.frontmatter__slug}.js X``src > pages > blog > JS {mdx.frontmatter__slug}.js > query`

```
1  import * as React from 'react'
2  import { graphql } from 'gatsby'
3  import { GatsbyImage, getImage } from 'gatsby-plugin-image'
4  import Layout from '../components/layout'
5
6  const BlogPost = ({ data, children }) => {
7
8      const image = getImage(data.mdx.frontmatter.hero_image)
9
10     return (
11         <Layout pageTitle={data.mdx.frontmatter.title}>
12             <p>{data.mdx.frontmatter.date}</p>
13             <GatsbyImage
14                 image={image}
15                 alt={data.mdx.frontmatter.hero_image_alt}
16             />
17             {children}
18         </Layout>
19     )
20 }
```



Auditoria de Desempenho



Lighthouse

Lighthouse é uma ferramenta automatizada de código aberto para melhorar a qualidade das páginas da web.

Você pode executá-lo em qualquer página da web, pública ou que exija autenticação.

Possui **auditorias de desempenho, acessibilidade, aplicativos web progressivos** (PWAs) e muito mais.

O Lighthouse está incluído no Chrome DevTools.

Executar sua auditoria – e então resolver os erros encontrados e implementar as melhorias sugeridas – é uma ótima maneira de preparar seu site para entrar no ar.

Isso ajuda a garantir que seu site seja o mais rápido e acessível possível.

Mais um Post

← → ↺

fascinating-medovik-ed5e1e.netlify.app/blog/mais-um-post/


🔗 ⭐ 🗨 ⚙ 🧩 🖱 👤 ⋮

Administrativo Conteúdo Ferramentas

Estudos do Jamstack com Gatsby

Início Blog

October 15, 2023




Esse é mais um post.

© 2023 Estudos de Gatsby. Todos os direitos reservados.

Lighthouse >>

+ (novo relatório) 🔍 ⚙



Gerar um relatório do Lighthouse

Analisar o carregamento de página

Modo [Saiba mais](#)

- ☒ Navegação (padrão)
- ☐ Período
- ☐ Resumo

Dispositivo

- ☐ Dispositivo móvel
- ☒ Computador

Categorias

- ☒ Desempenho
- ☒ Acessibilidade
- ☒ Práticas recomendadas
- ☐ SEO
- ☐ App Web Progressivo

Plug-ins

- ☐ Anúncios do editor

Mais um Post

← → ↺

fascinating-medovik-ed5e1e.netlify.app/blog/mais-um-post/


🔗 ⭐ 🛠 ⚙ 🗑 👤 ⋮

Administrativo Conteúdo Ferramentas

Estudos do Jamstack com Gatsby

Início Blog

October 15, 2023



Esse é mais um post.

© 2023 Estudos de Gatsby. Todos os direitos reservados.

Elementos Lighthouse >> ⚙ ⋮ ✕

+ 19:43:27 - fascinating-medovi 🔒

🏠 https://fascinating-medovik-ed5e1e.netlify.app/blog/...

100

Desempenho

88

Acessibilidade

100

Práticas recomendadas

100

Desempenho

Os dados são estimados e podem variar. O desempenho é calculado diretamente a partir das métricas. [Ver calculadora.](#)

50-89

90-100

MÉTRICAS

Abrir visualização

●

First Contentful Paint

0,4 s

●

Largest Contentful Paint

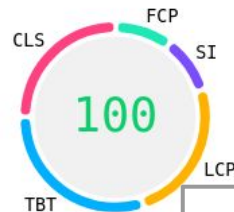
0,4 s

Lighthouse Scoring Calculator

Device type: Desktop Versions: v10, v11, v12

latest
[v10](#)

	Value	Metric Score	Weighting
● FCP (First Contentful Paint)	387 ms	100	10%
● SI (Speed Index)	387 ms	100	10%
● LCP (Largest Contentful Paint)	387 ms	100	25%
● TBT (Total Blocking Time)	58 ms	100	30%
● CLS (Cumulative Layout Shift)	0,00	100	25%



Learn more about scoring at [web.dev/performance-scoring](#).

► Scores under 5/100 are not supported by this UI. Why?



Largest Contentful Paint

Tempo de renderização do maior elemento de conteúdo visível.

Uma medida do tempo de carregamento de sua página que verifica quanto tempo leva para o maior elemento acima da dobra (imagem, texto etc.) carregar.

Acima da dobra significa que a métrica considera apenas o que aparece na página, sem rolar para baixo.

Limite ideal: menos de 2,5 segundos.

LCP

Largest Contentful Paint



Cumulative Layout Shift

Deslocamento Cumulativo de Layout.

Mede a estabilidade visual de um carregamento de página focando em mudanças inesperadas de layout não causadas por uma interação do usuário.

Às vezes, quando uma página é carregada, os elementos mudam inesperadamente e frustram os usuários.

Por exemplo, você pode carregar uma página e começar a ler um parágrafo, apenas para carregar uma imagem que empurra o parágrafo para baixo na página.

Limite ideal: menor que 0,1.

CLS

Cumulative Layout Shift



Total Block Time

Tempo total de bloqueio.

Mede a quantidade total de tempo que uma página está bloqueada para responder à entrada do usuário.

Essas entradas incluem cliques do mouse ou toques no teclado.

Limite ideal: menos de 200 milissegundos.

First Contentful Paint

Primeira Exibição de Conteúdo.

É o tempo que leva até que a primeira parte do conteúdo seja carregada na página. O conteúdo deve vir do DOM (Document Object Model) da página.

O DOM inclui conteúdo de página padrão, como imagens e texto.

Limite ideal: menos de 1,8 segundos.

FCP

First Contentful Paint



Speed Index

Índice de velocidade.

Mede todo o processo de carregamento das partes visuais de uma página, capturando um vídeo do carregamento da página e verificando a diferença entre os frames.

A duração total mede essencialmente quanto tempo leva para ir da tela em branco à página completa.

Limite ideal: menos de 3,4 segundos.

	Bom	Deve Melhorar	Ruim
First Contentful Paint (FCP)	0 - 1000 ms	1000 ms - 3000 ms	> 3000 ms
Speed Index (SI)	0 - 4300 ms	4400 ms - 5800 ms	> 5800 ms
Largest Contentful Paint(LCP)	0 - 2500 ms	2500 ms - 4000 ms	> 4000 ms
Time to Interactive (TTI)	0 - 3800 ms	3900 ms - 7300 ms	> 7300 ms
Total Blocking Time (TBT)	0 - 300 ms	300 ms - 600 ms	> 600 ms
Cumulative Layout Shift(CLS)	0 - 0.1	0.1 - 0.25	> 0.25