Explorando novas fronteiras com Astro

Gabriel Napoleão

Software Engineer

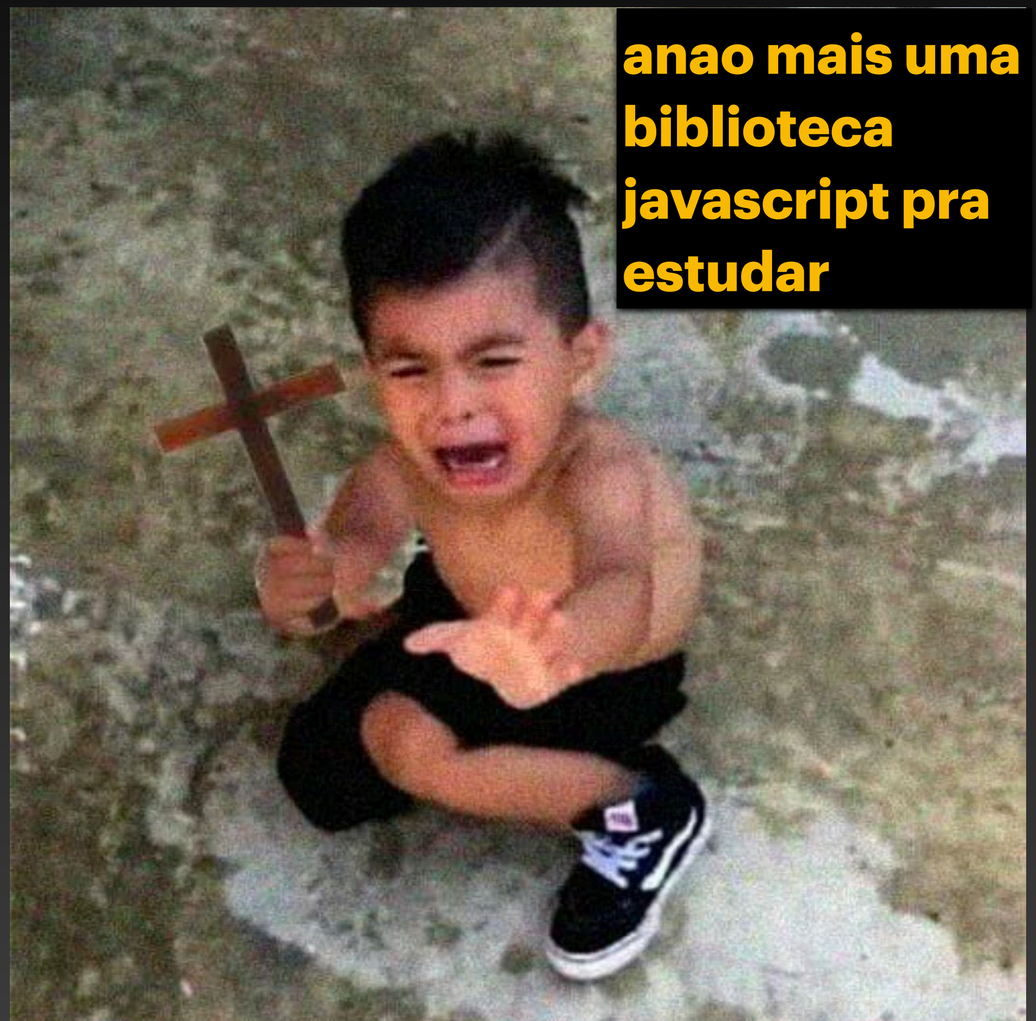
/whoamindx

whoamindx.github.io

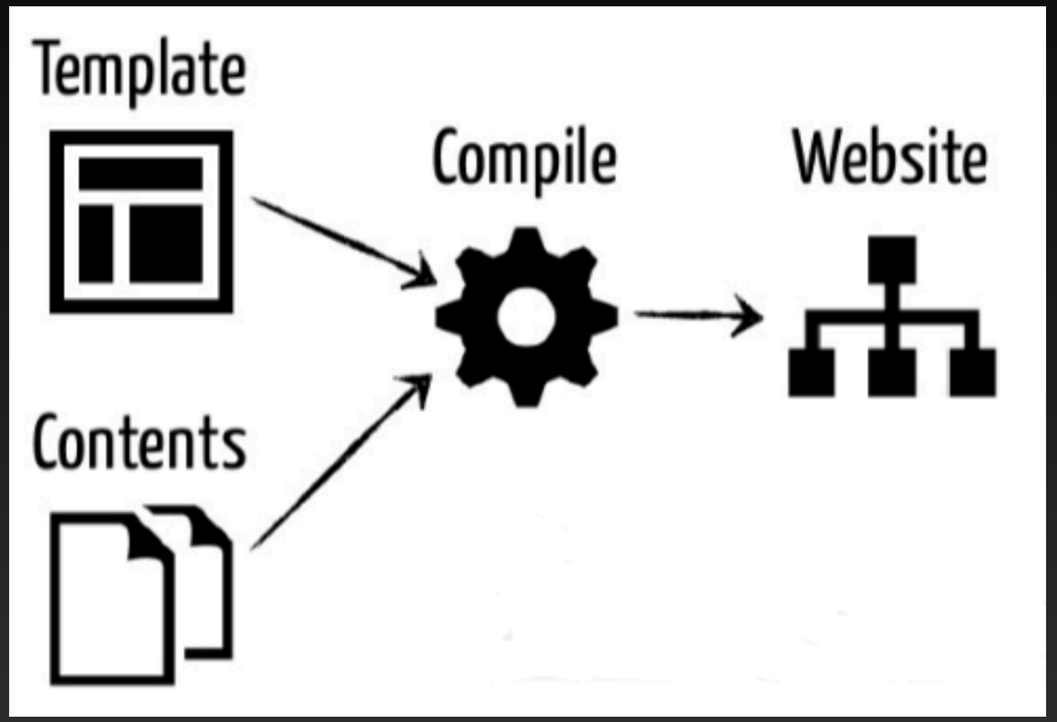
@whoamindx

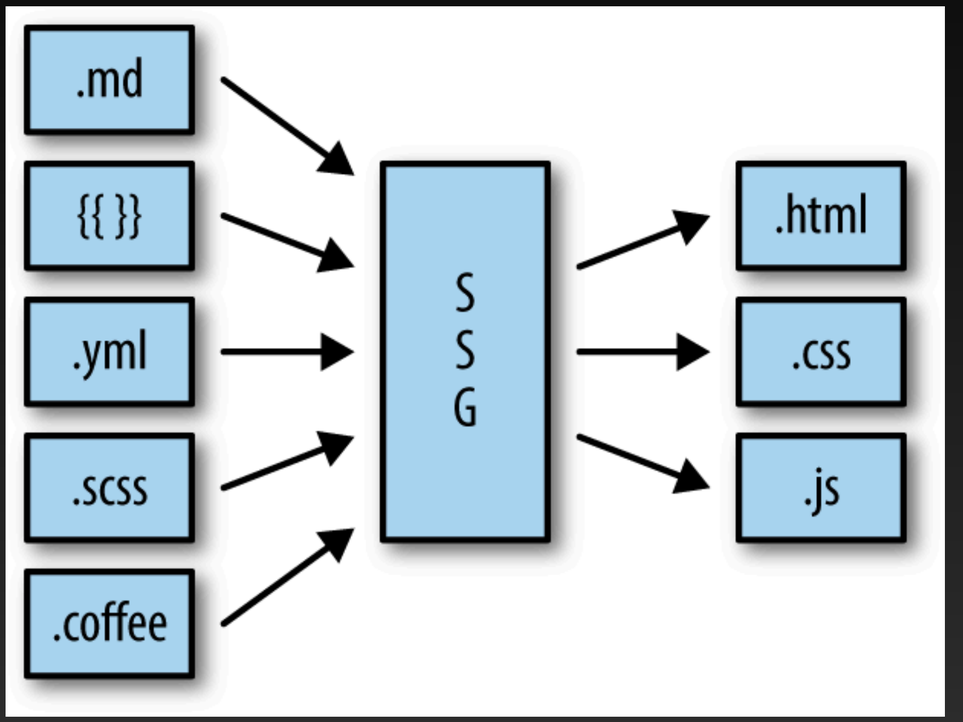
* MAIS UMA PALESTRA DE UM FRAMEWORK JAVASCRIPT ALEATÓRIO ACONTECE*

- VOCÊS:




enyo
Ractive.js
Sammy.js
GWT
dojo
canjs
mochikit
jQuery
ember
Knockout.
ext js
METEOR


Template
Compile
Website
Contents


.md
{{ }}
.yml
.scss
.coffee
S
S
G
.html
.css
.js

SSG

vs

SSR

CSR

Static Site Builder

Astro is a web framework for building content-driven websites like blogs, marketing, and e-commerce.

Astro is MPA

Bundle Size

Angular ~ 500 KB
React - 150 KB (NextJS)
Vue - 80 KB

Official UI Framework Integrations

UI Frameworks

@astrojs/**alpinejs**

@astrojs/**lit**

@astrojs/**preact**

@astrojs/**react**

@astrojs/**solid-js**

@astrojs/**svelte**

@astrojs/**vue**

Astro não resolve todos os problemas

E ele deixa isso bem claro na sua documentação...

> Dica
>
> Se o seu projeto cai no segundo campo de “aplicações”, Astro pode não ser a escolha certa para seu projeto... **e tá tudo certo!** Veja o Next.js como uma alternativa mais focada em aplicações do que o Astro.

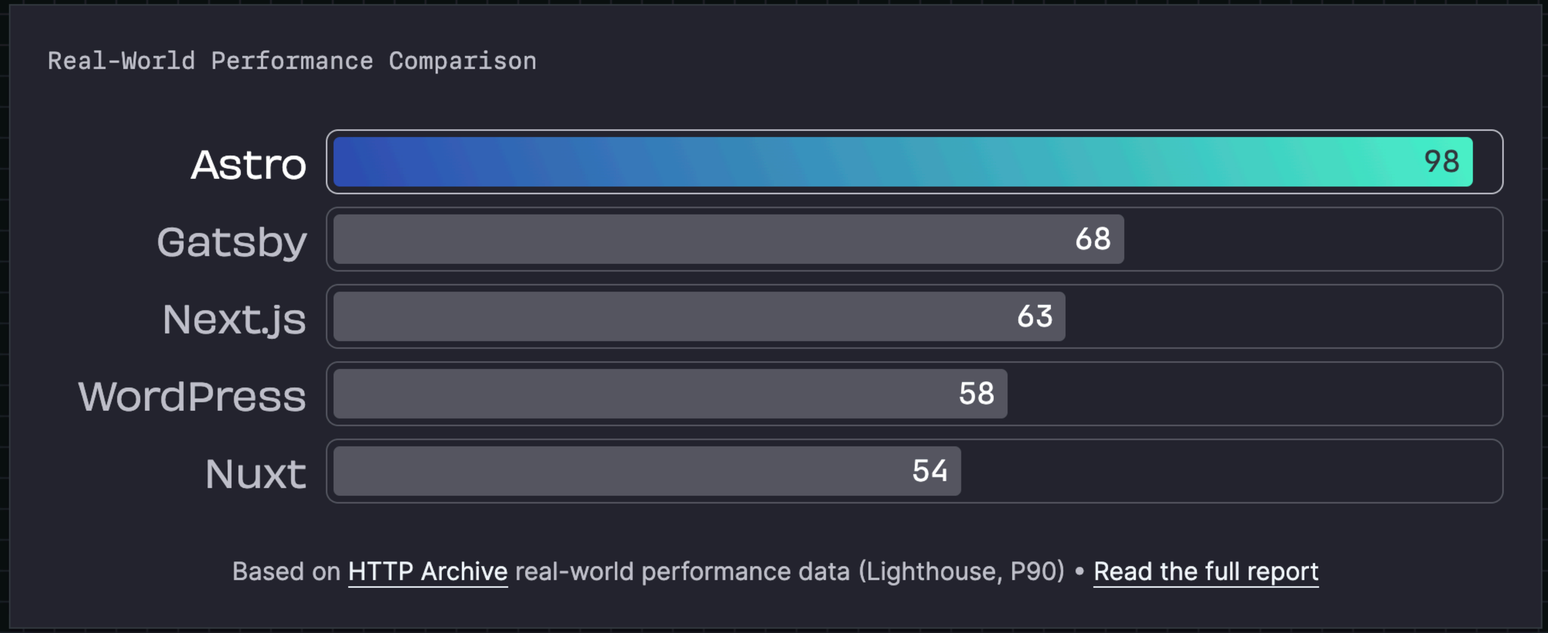
Astro é...

1. Focado em conteúdo: Astro foi feito para websites ricos em conteúdo.
2. Servidor em primeiro lugar: Websites são mais rápidos quando eles renderizam HTML no servidor.
3. Rápido por padrão: Deve ser impossível construir um website lento com Astro.
4. Fácil de utilizar: Você não precisa ser um especialista para construir algo com Astro.
5. Cheio de funcionalidades, porém flexível: Mais de 100 integrações Astro para escolher.

Absurdamente rápido...

Full speed

Islands optimize your website like no other web framework
can. Leverage Astro's unique page load performance
to improve conversion rates, Core Web Vitals, and SEO.

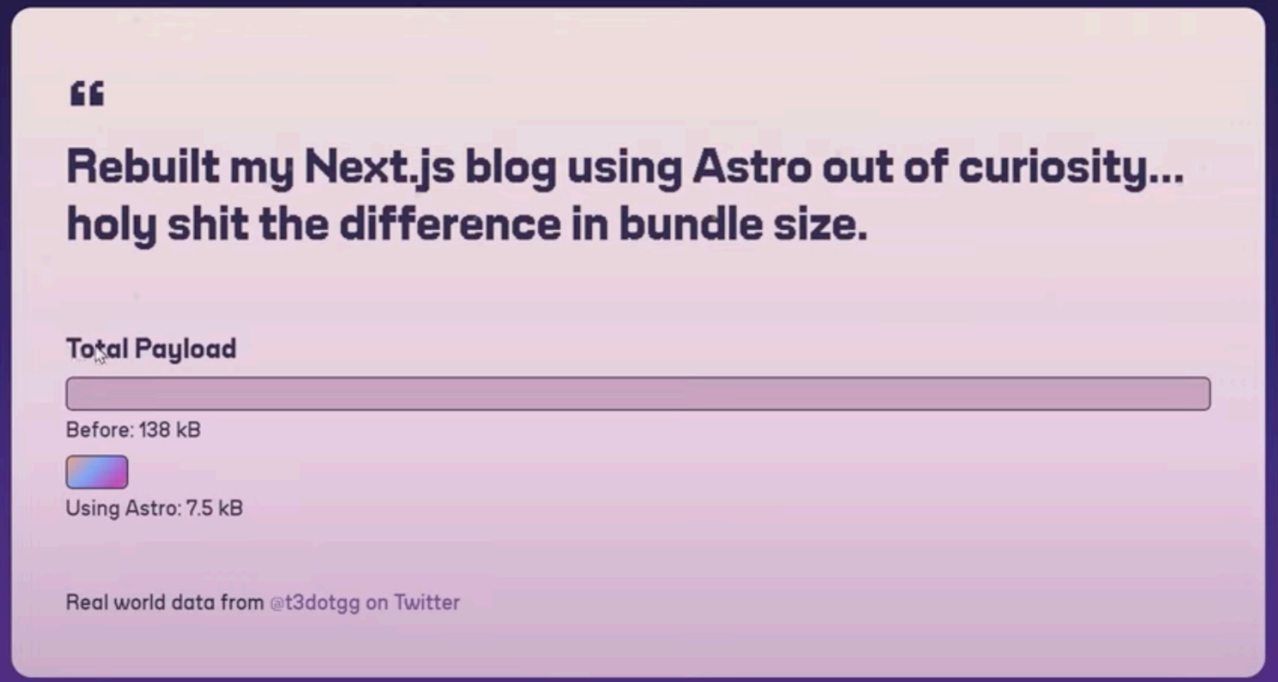


Absurdamente rápido...

Peak Performance

Site performance has a tangible impact on the success of your business. Astro sites can be up to 40% faster and use 90% less JavaScript compared to leading static site generators.

87% of websites currently fail Google's Core Web Vitals (CWV) benchmark. Astro is on a mission to change that.



Astro tenta compilar tudo ao máximo em HTML e CSS tentando evitar sempre que possível código JavaScript no lado do client

Astro Components

index.astro | SomeComponent.astro

```
---
const name = 'Astro';
---

<h1>Hello, {name}!</h1>
```

Astro Components

index.astro SomeComponent.astro

```
---
const items = ["Svelte", "Preact", "React", "Vue", "Solid"];
---

<ul>
  {items.map(item => (
    <li>{item}</li>
  ))}
</ul>

<style>
li { color: rebeccapurple }
</style>
```

- Importação de Components
- Importação de Dados
- APIs e Banco de Dados
- Criação de Variáveis

Routing system

Astro provides both **static** and **dynamic** route generator.

`.astro` page components as well as **Markdown** and **MDX** Files (.md, .mdx) within the `src/pages/` directory automatically become pages on your website.

```
# Example: Static routes
src/pages/index.astro          -> mysite.com/
src/pages/about.astro          -> mysite.com/about
src/pages/about/index.astro    -> mysite.com/about
src/pages/about/me.astro       -> mysite.com/about/me
src/pages/posts/1.md           -> mysite.com/posts/1
```

Dynamic routes (SSG)

An Astro page file can specify dynamic route parameters to generate multiple pages.

For example, `src/pages/authors/[author].astro` generates a **bio page for every author** on your blog.

This will generate three static pages:

- `/authors/peter`
- `/authors/steve`
- `/authors/peggy`

```
---
export function getStaticPaths() {
  return [
    { params: { author: "Peter" } },
    { params: { author: "Steve" } },
    { params: { author: "Peggy" } },
  ];
}

const { author } = Astro.params;
---

<div>Hello, my name is {author}!</div>
```

Conteúdo via MD, MDX ou seu CMS favorito

Deploy


Netlify
SSR
STATIC
Vercel
SSR
STATIC
GitHub Pages
STATIC
GitLab Pages
STATIC
Cloudflare Pages
SSR
STATIC
aws
AWS
STATIC
Flightcontrol
AWS via Flightcontrol
SSR
STATIC
sst
AWS via SST
SSR
STATIC
Google Cloud
SSR
STATIC
Firebase Hosting
STATIC
Heroku
STATIC
Microsoft Azure
STATIC

Suporta TS por default

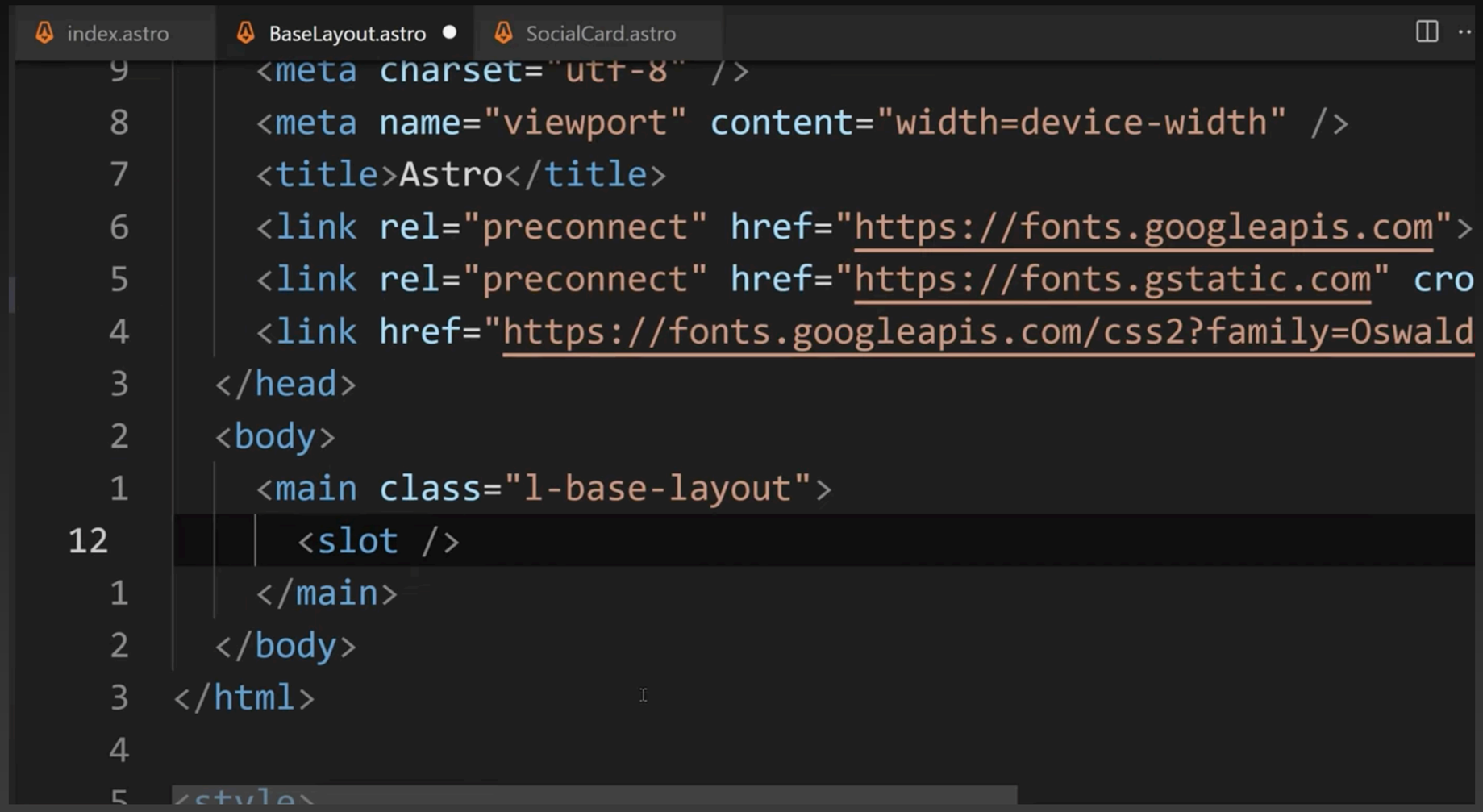
```astro
// SimpleComponent.astro
---
type Props = {
  title: string;
};

const { title } = Astro.props;
---

<div>
  <h3>{title}</h3>
  <section>
    <slot />
  </section>
</div>


// index.astro
<div>
  <SimpleComponent
  title="My first component">
      My first component slot render
   </SimpleComponent>
</div>
```

Layouts são Components “Especiais"



index.astro ● | BaseLayout.astro | SocialCard.astro

```
6  ---
5  import BaseLayout from "../layouts/BaseLayout.astro";
4  import SocialCard from "../components/SocialCard.astro";
3  ---
2
1  <BaseLayout>
7    <SocialCard />
1  </BaseLayout>
2
```

⚠ Cuidado

Atributos HTML vão ser convertidos para strings, portanto não é possível passar funções e objetos para elementos HTML. Por exemplo, você não pode atribuir um manipulador de eventos a um elemento HTML em um componente Astro:

nao-faca-isso.astro

```
---
function tratarClique () {
    console.log("botão clicado!");
}
---
<!-- ❌ Isso não funciona! ❌ -->
<button onClick={tratarClique}>Nada vai acontecer quando você me clicar!</butto
```

Ao invés disso, use um script do lado do cliente para adicionar o manipulador de evento, como você faria no JavaScript vanilla:

faca-isso-em-vez-disso.astro

```
<button id="botao">Clique em Mim</button>
<script>
  function tratarClique () {
    console.log("botão clicado!");
  }
  document.getElementById("botao").addEventListener("click", tratarClique);
</script>
```

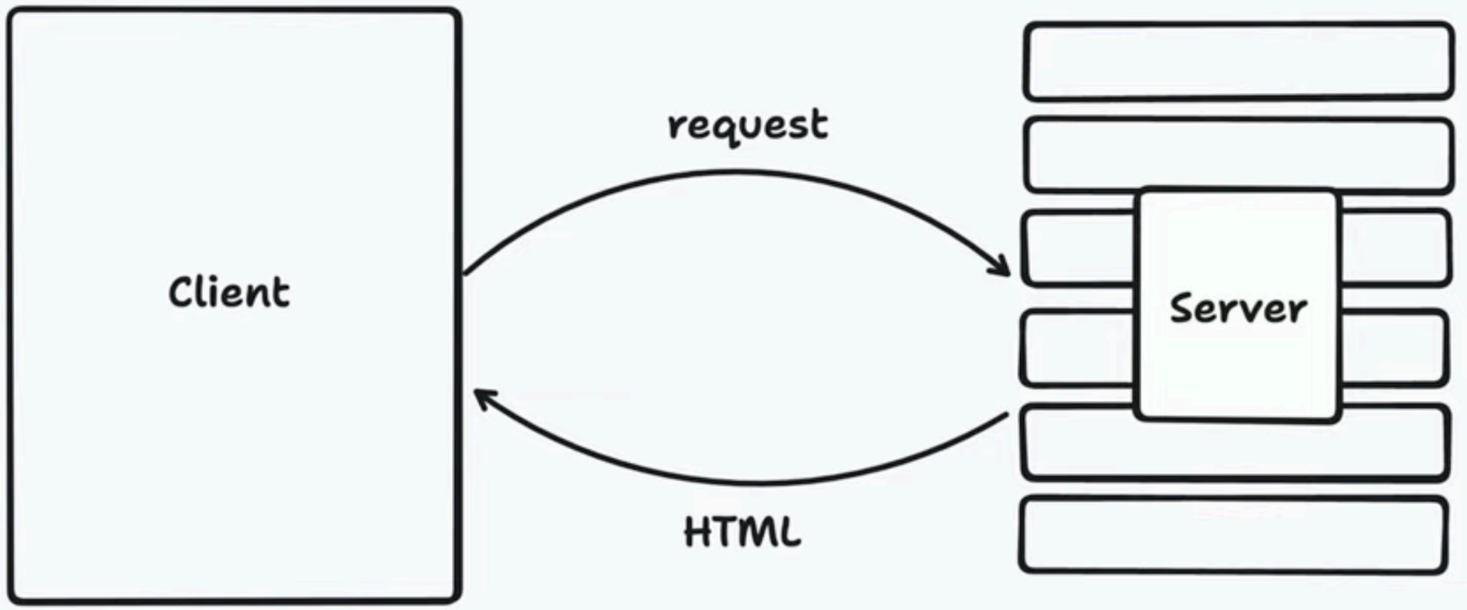
Cosméticos

Jequiti

Brasil no corpo e na alma

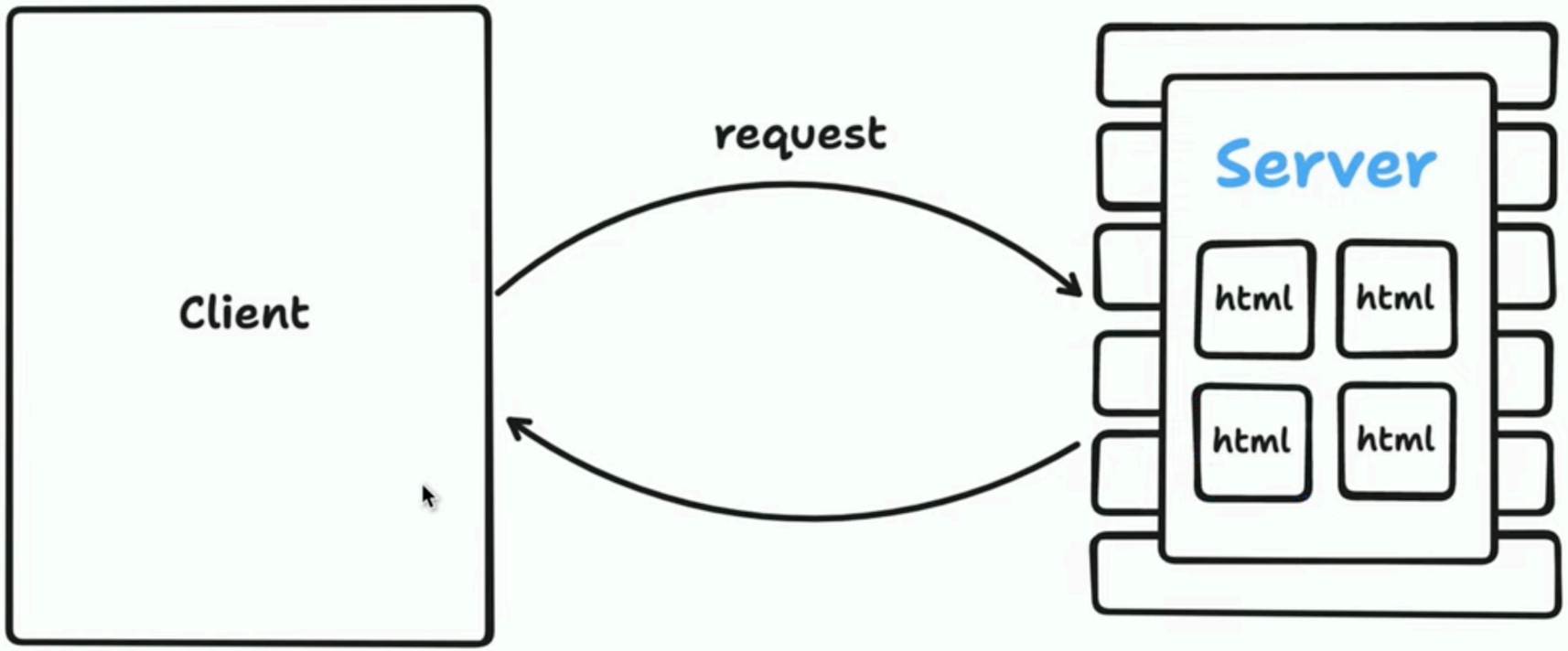
astro

Rendering Options



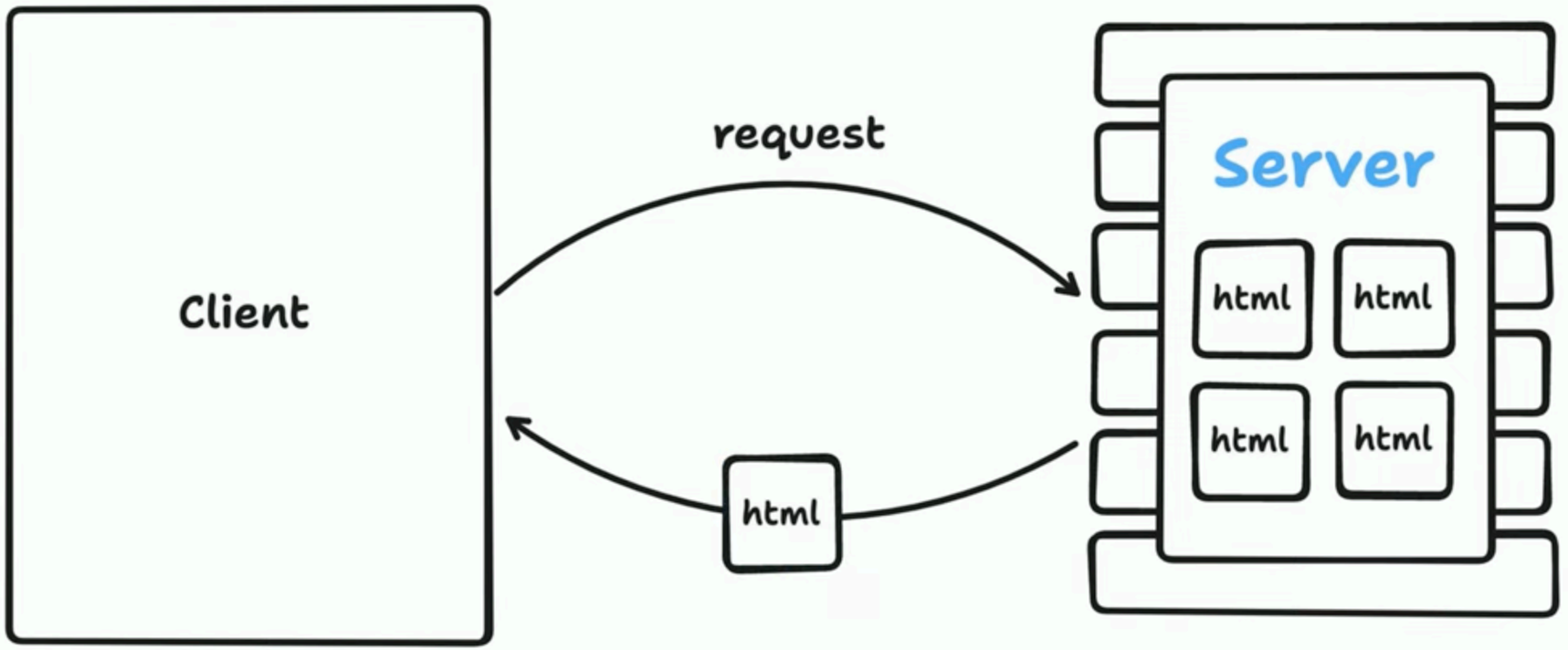
Static Site Generation

Default

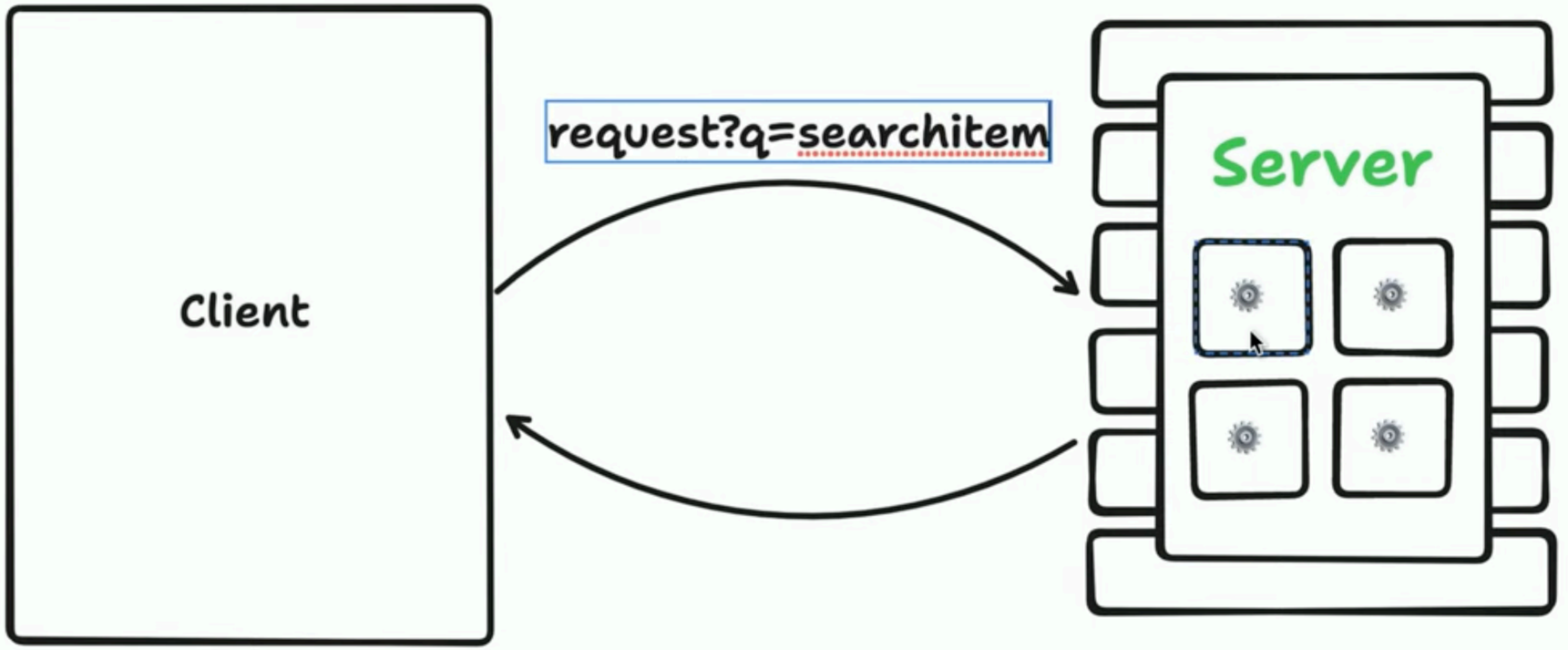


Static Site Generation

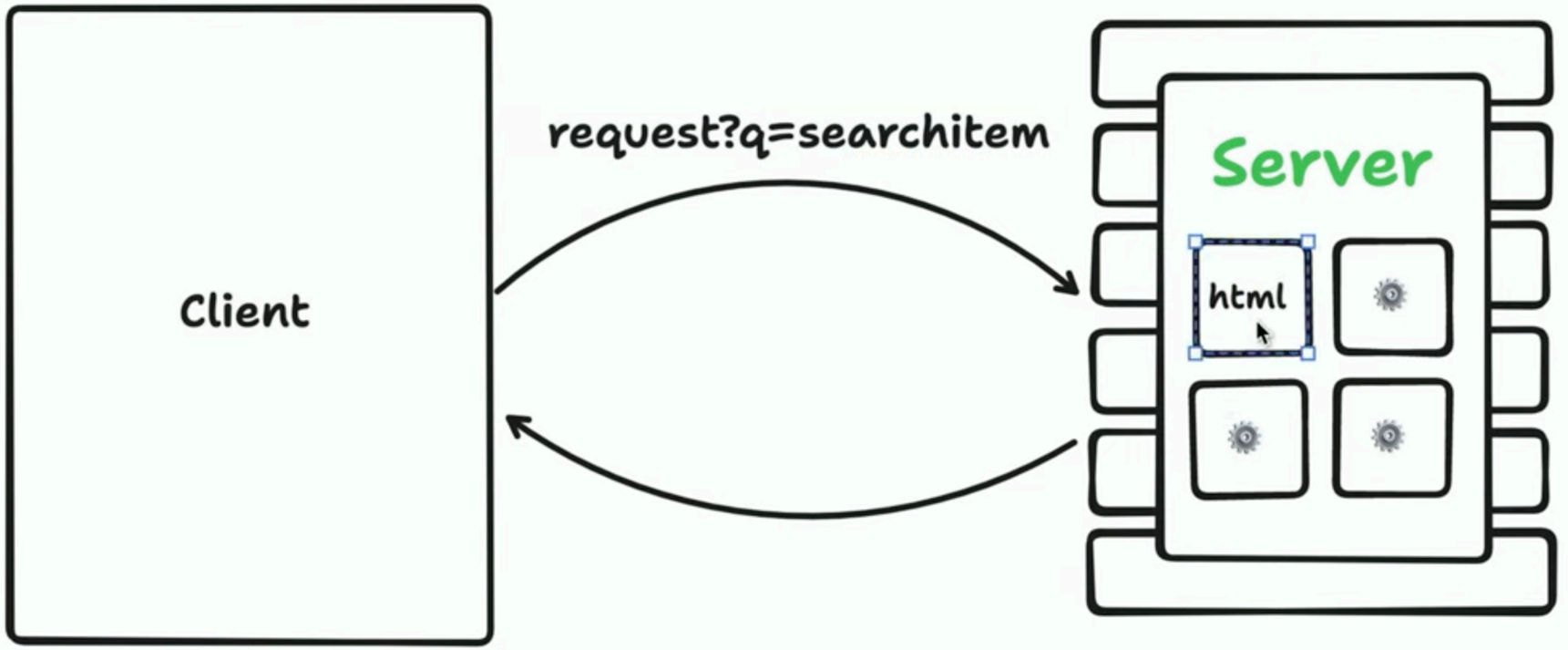
Default



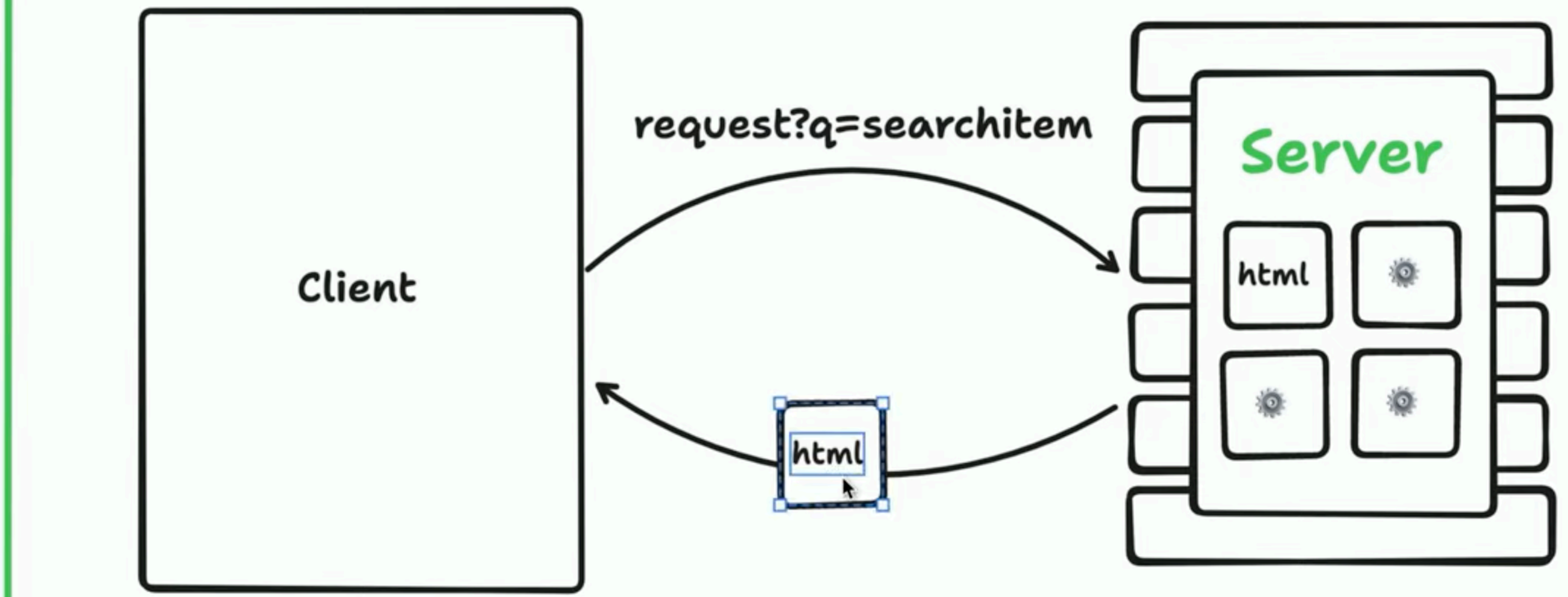
Server Side Rendered

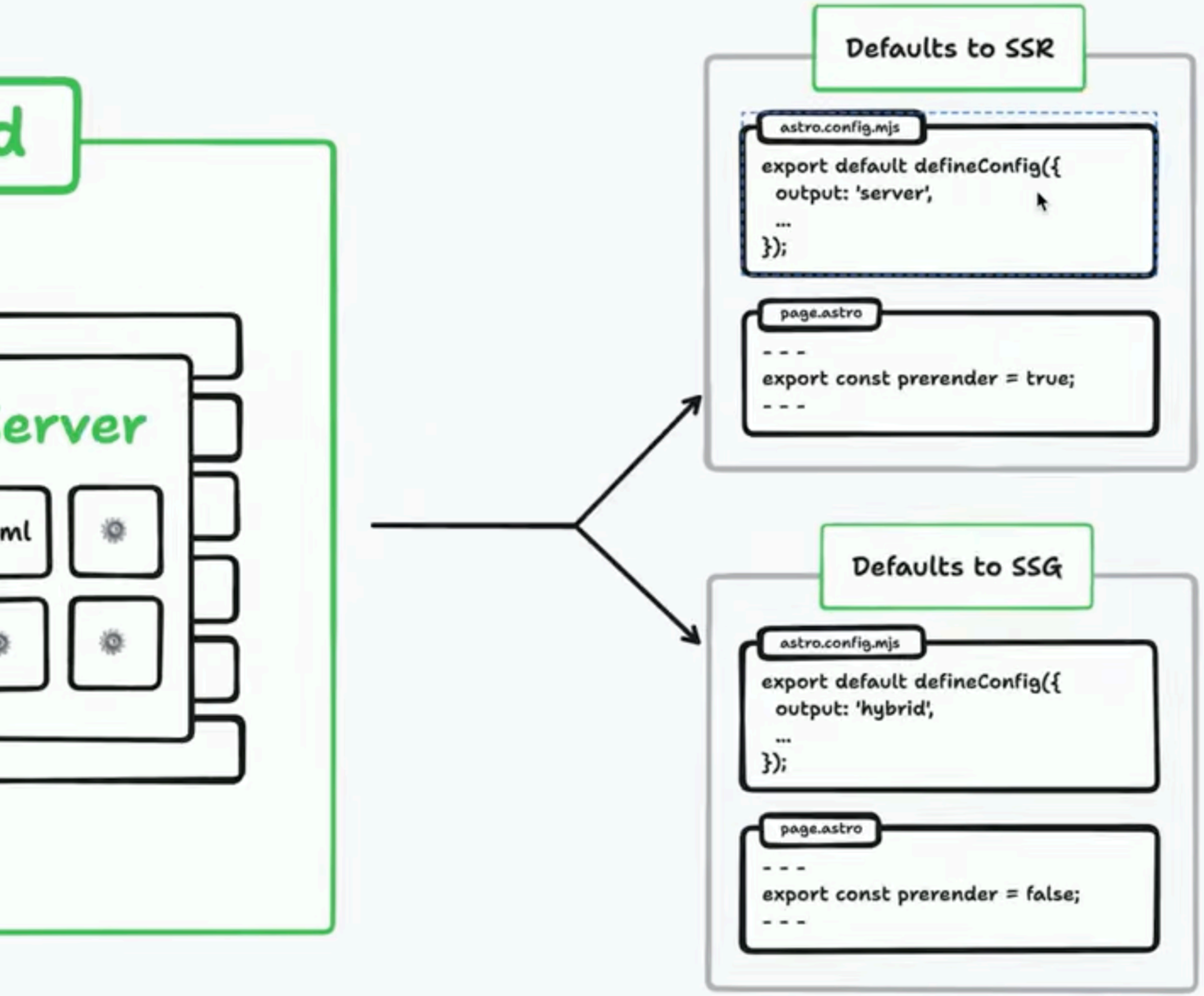


Server Side Rendered



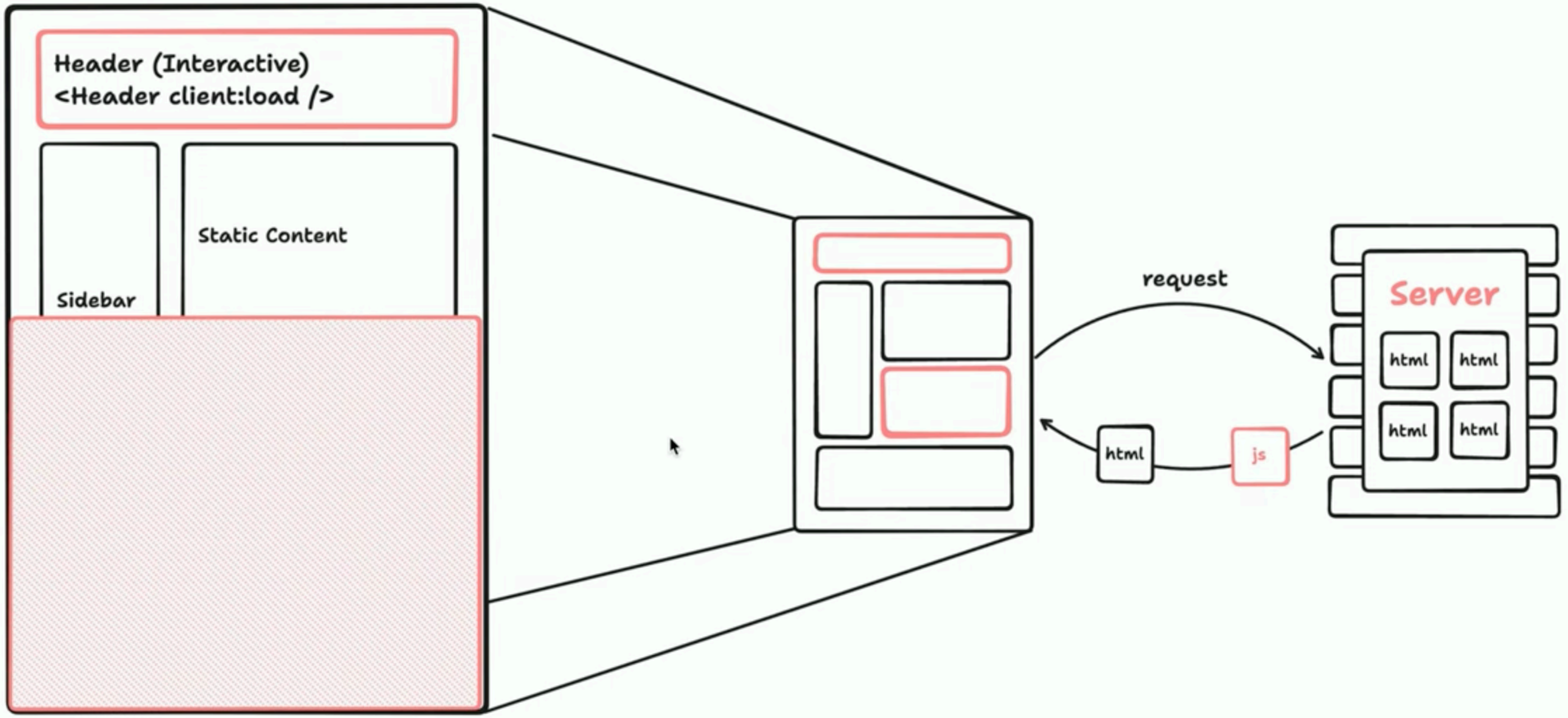
Server Side Rendered



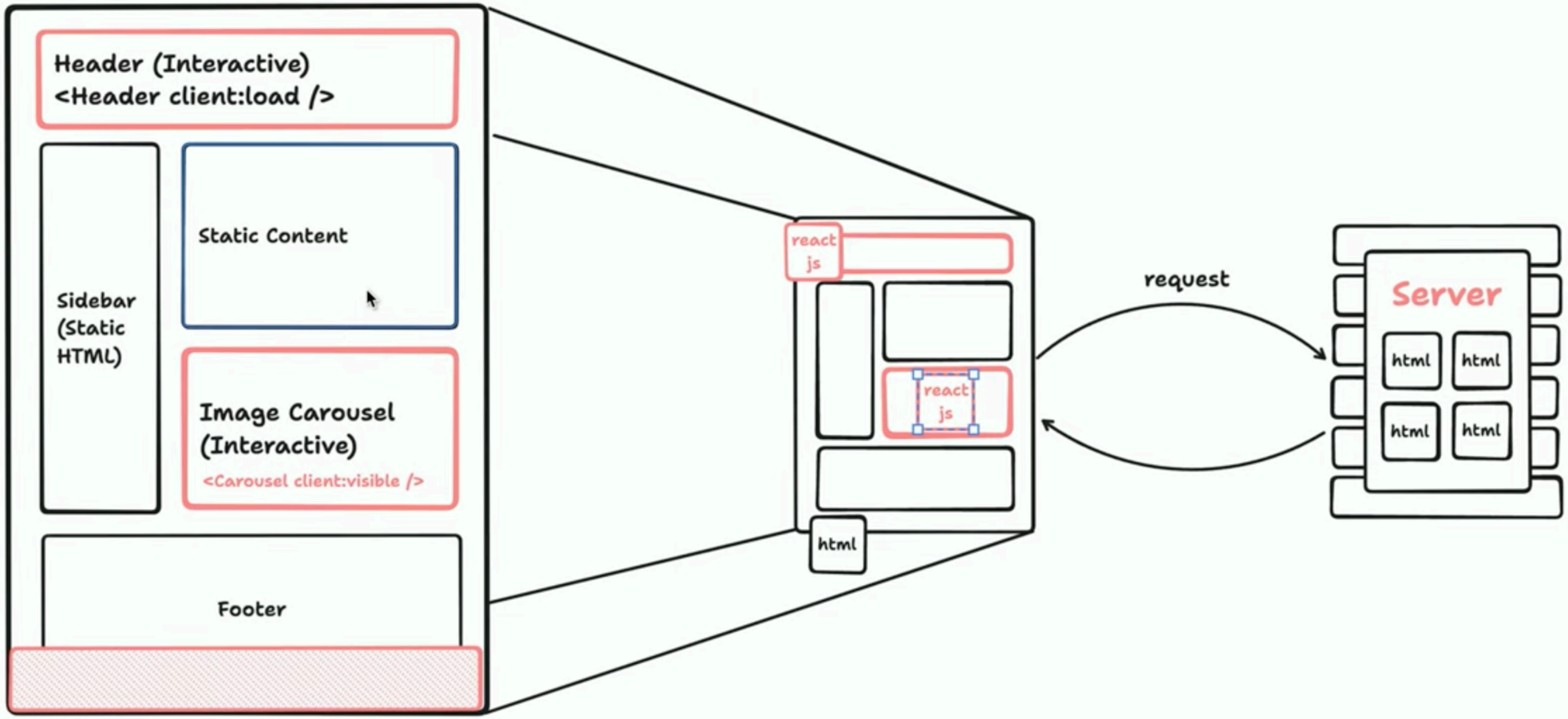

d
erver
ml
Defaults to SSR
astro.config.mjs
export default defineConfig({
output: 'server',
...
});
page.astro
- - -
export const prerender = true;
- - -
Defaults to SSG
astro.config.mjs
export default defineConfig({
output: 'hybrid',
...
});
page.astro
- - -
export const prerender = false;
- - -

Astro Islands

Astro Islands



Astro Islands



Criando uma ilha

Por padrão, Astro vai renderizar automaticamente todos os componentes de UI para apenas HTML e CSS, **removendo todo o JavaScript do lado do cliente automaticamente.**

src/pages/index.astro

```
<MeuComponenteReact />
```

Isso pode soar restrito, mas esse comportamento é o o que mantém websites Astro rápidos por padrão e proteje desenvolvedores de acidentalmente enviar JavaScript desnecessário ou indesejado que pode tornar seu website lento.

Tornar qualquer componente de UI estático em uma ilha interativa requer apenas uma diretiva `client:*`. Astro, então, vai fazer um build automaticamente e empacotar seu JavaScript do lado do cliente para uma performance otimizada.

src/pages/index.astro

```
<!-- Este componente agora é interativo na página!
     O resto do seu website continua estático. -->
<MeuComponenteReact client:load />
```

Com ilhas, JavaScript do lado do cliente só é carregado para os componentes explicitamente interativos que você demarcar utilizando diretivas `client:*`.

E por causa das interações serem definidas a nível de componente, você pode lidar com diferentes prioridades de carregamento para cada componente baseado em seu uso. Por exemplo, `client:idle` informa um componente para carregar quando o navegador estiver inativo, e `client:visible` informa um componente para carregar apenas quando ele entrar na janela de visualização.

Obrigado pela sua atenção!