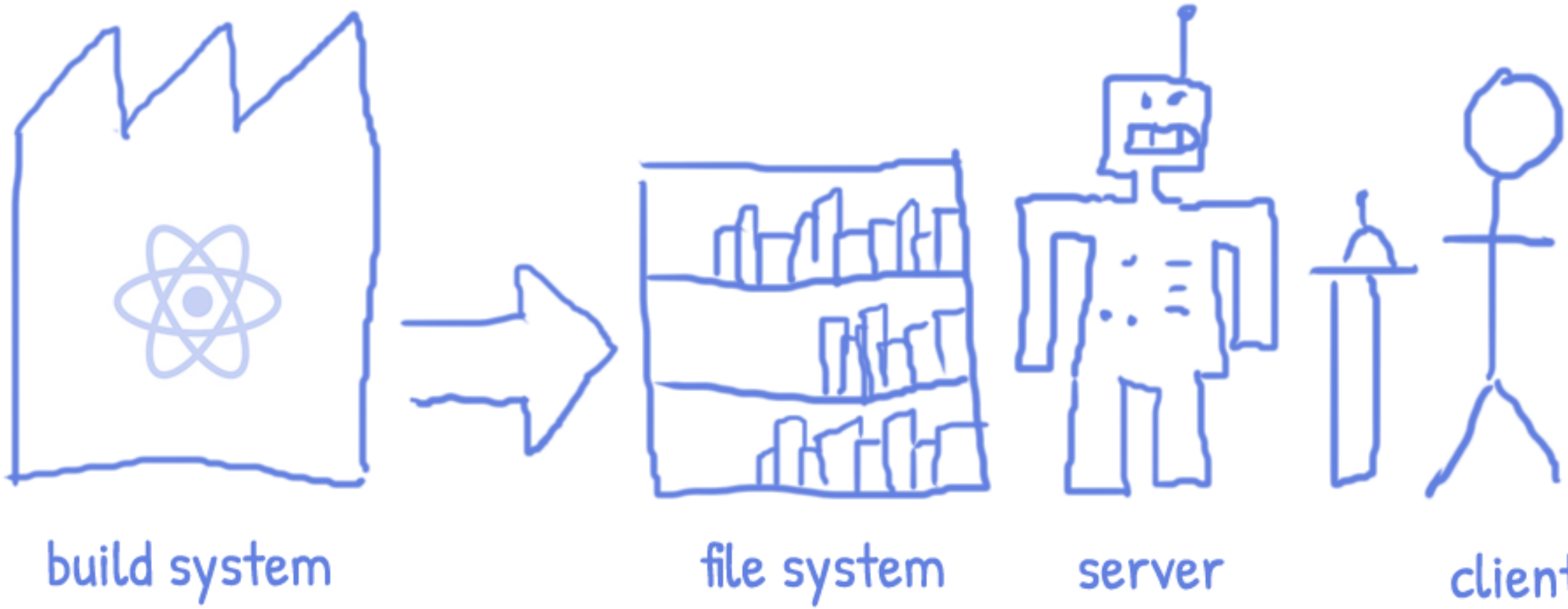
React + SSR

Conteúdo

- Definições
  - Static rendering
  - Server-side rendering
- Features do SSR
  - getInitialProps
  - prefetch
- "Caveats" do SSR
  - FOUC
  - Global "window"
- Resumo / Dúvidas / Bate-papo

Static rendering

- a.k.a. CSR (Client-side rendering)
- Servidor responde requisições com um HTML estático (sem processamento)
- HTML estático, inicialmente com pouco ou nenhum conteúdo referencia js, css e outros recursos
- Página é montada depois que recursos são carregados
- Exemplo: create-react-app, Gatsby.js


build system
file system
server
client

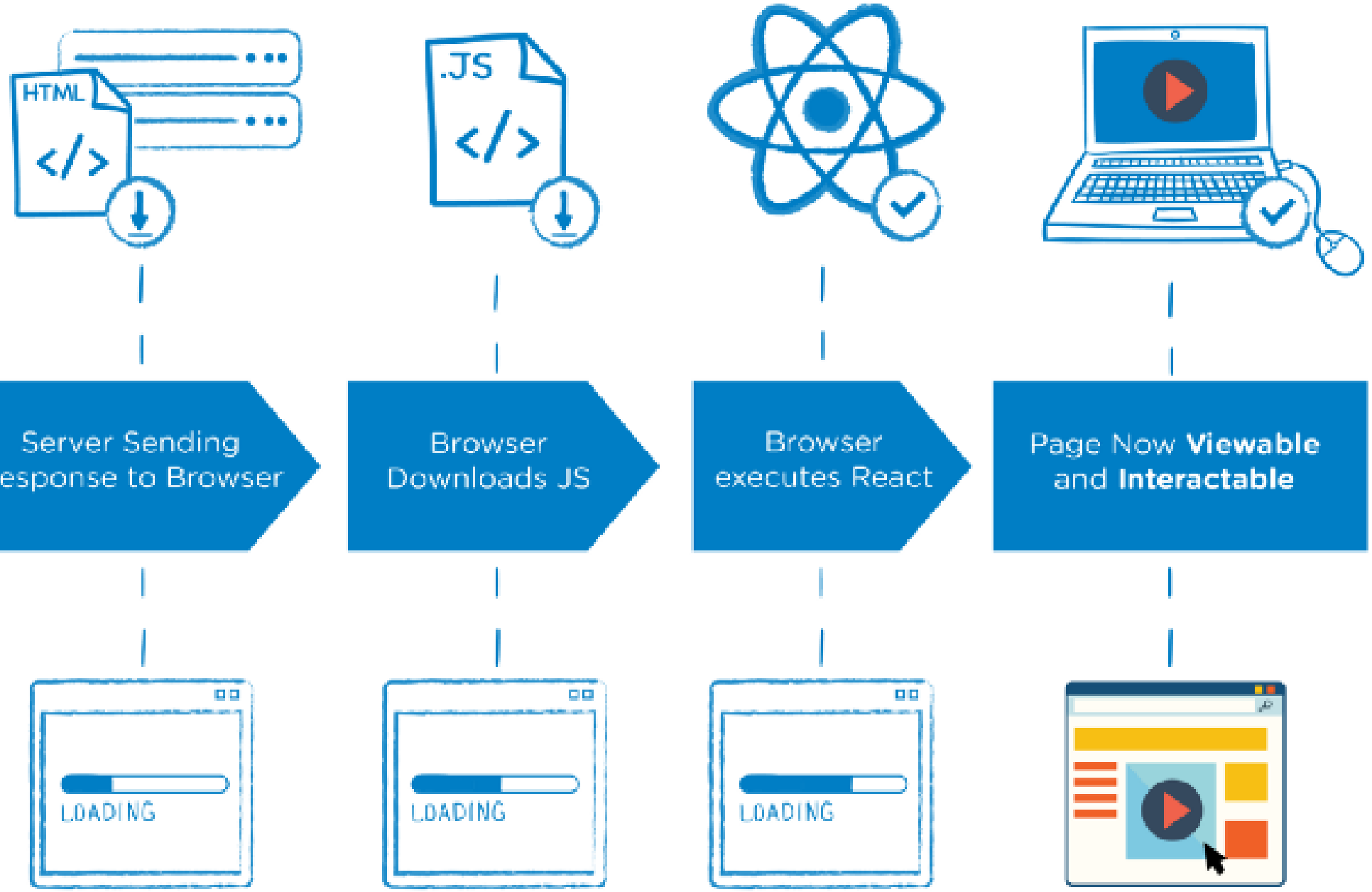
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <!-- meta tags... -->
    <title>My page</title>
  </head>
  <body>
    <div id="root"></div>
    <!-- more scripts... -->

    <script src="/my-page/static/js/main.676bc650.chunk.js"></script>
  </body>
</html>
```

Static rendering

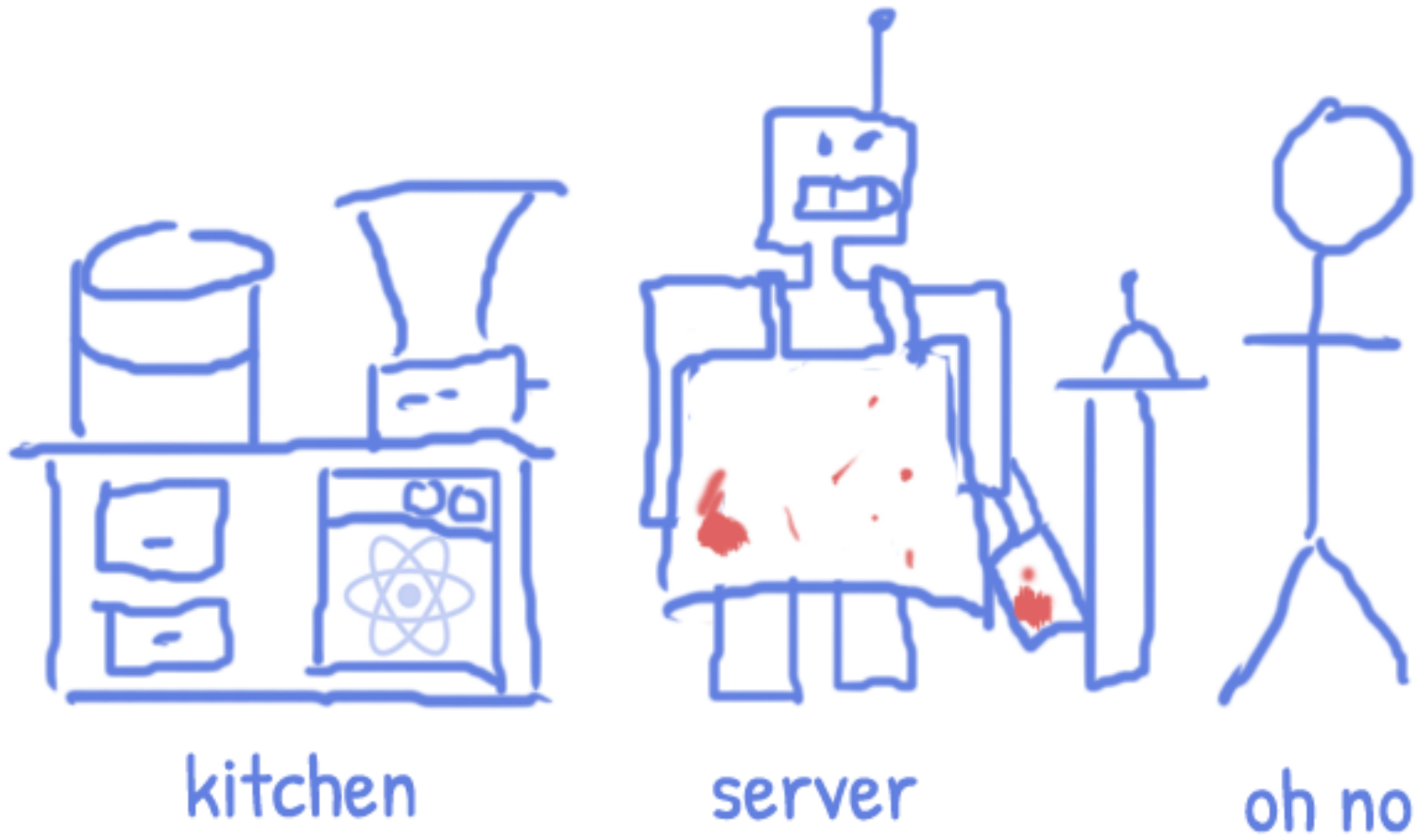
- Crawlers podem não executar JS
  - Página pode aparecer sem conteúdo para os search engines e previews
- Conteúdo dinâmico somente após carregamento e execução do JS
  - Não é possível usar conteúdo dinâmico para SEO

CSR

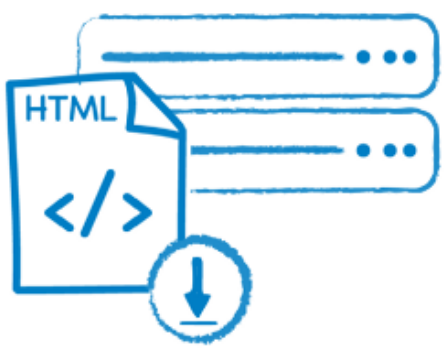

HTML
.JS
Server Sending Response to Browser
Browser Downloads JS
Browser executes React
Page Now **Viewable** and **Interactable**
LOADING
LOADING
LOADING

Server-side rendering

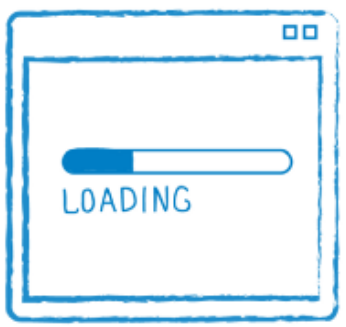
- Servidor processa requisições e retorna HTML montado dinamicamente
- Página vem pré-renderizada do servidor (com conteúdo mas não interativa), renderiza por completo no client
    - ocorre "hydration" no client: Assimilação do HTML já gerado com o Virtual DOM do React
- Exemplos: next.js, after.js

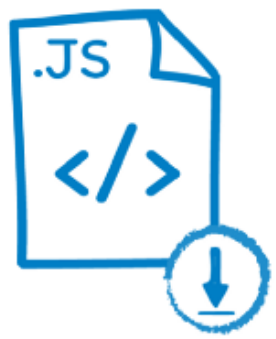

kitchen
server
oh no

SSR

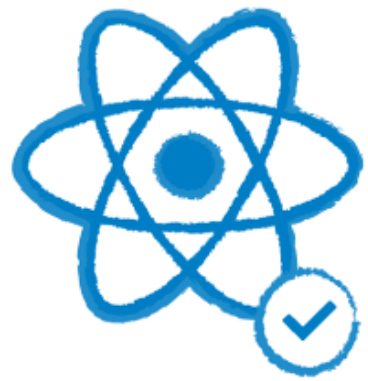


Server Sending Ready to be rendered HTML Response to Browser





Browser Renders the page, **Now Viewable**, and Browser Downloads JS



Browser executes React

Page Now **Interactable**

Server-side rendering

getInitialProps

- permite fazer chamadas assíncronas antes de responder a requisição
- pode usar dados da requisição para isso (query params, headers...)
- exemplo: usar id de uma rota para preencher <title> com nome do recurso
- permite indexar páginas individuais nos motores de busca
- next.js e after.js suportam

Server-side rendering

getInitialProps (exemplo)

https://www.netflix.com/br/title/80192098

⬇️

```
<title>La casa de papel | Site Oficial Netflix</title>
```

Server-side rendering

prefetch

- permite carregar previamente conteúdo de páginas que o usuário pode acessar a partir da atual
  - exemplo: se estou na página principal (/), posso ir para /eventos
  - suporte a code splitting
  - next.js e after.js suportam, de maneiras diferentes

Server-side rendering

FOUC (Flash Of Unstyled Content)

- Acontece quando os estilos são processados depois de o HTML estar pronto
- Soluções:
  - incluir <style>s críticos no <head>
  - configurar css-in-js para SSR (styled-components, emotion)

Server-side rendering

Global window

- window existe somente no navegador, não no Node.js
- Soluções:
  - if (typeof window === "undefined") ...
  - next.js: dynamic component import opcional no SSR
  - delegar chamadas com window para client se possível
    - updates não são executados no servidor

Server-side rendering

Global window

Exemplos:

- Configurações específicas para server-side
  - URLs diferentes do backend para client e server-side
- Componentes diferentes em mobile e desktop. Como saber o tamanho da tela?
  - Server-side usa estimativa baseada no header User-Agent

Resumo

- Static rendering: devolve HTML e JS estáticos, renderização ocorre somente no client
- Server rendering: devolve HTML dinâmico, renderização inicial no servidor e hidratação no cliente
- Usar SSR quando:
  - HTML inicial precisa mudar de acordo com a rota e atender crawlers que não executam JS (para fins de SEO)
  - "Graceful degradation" caso o client não execute javascript

thanks ;)

https://github.com/vspedr