

R19

Next.js do 0

Ou;

Criando um framework Reaet 19 em menos de 100 linhas de código.

O que é o React 19 ?

React Server Actions

Novos hooks e APIs

React Server Components

Compilador experimental

Diferenças de **Server** / **Client** Components?

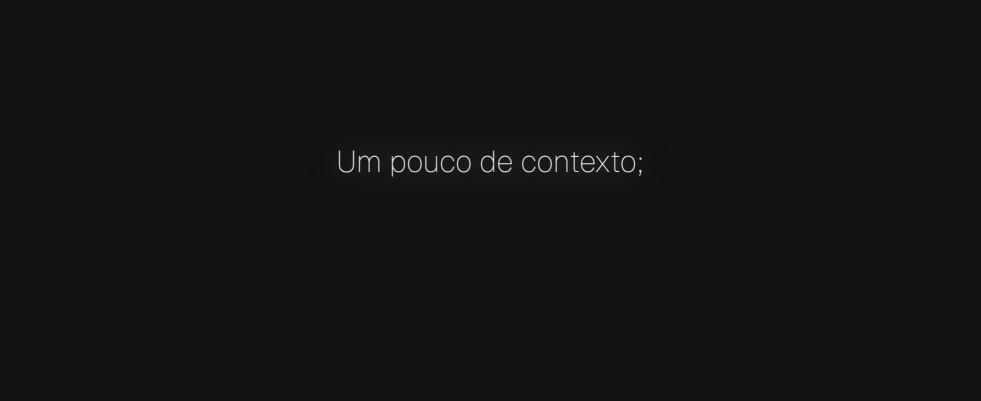
Server Components		Renderiza no servidor?	Renderiza no cliente?	
Client Components \sqrt{*}	Server Components	√	×	
	Client Components	√ *	✓	

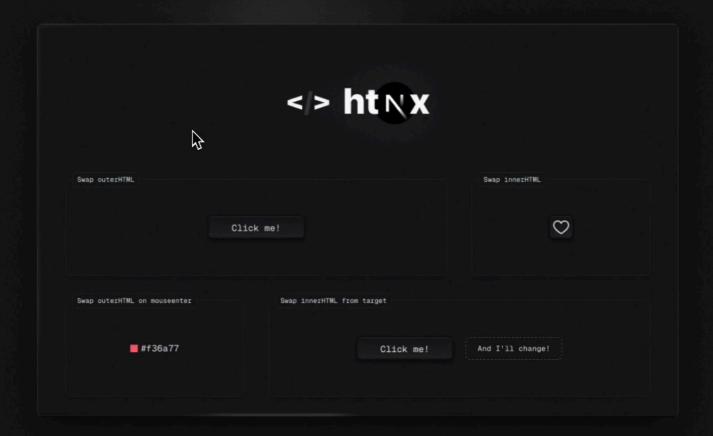
O que são **Server Actions** ?

Em resumo, um RPC, integrado no React.

React 18 VS React 19

```
async function create(data) {
 const response = await fetch("/api/post", {
   method: "POST",
   body: JSON.stringify(data)
 })
 return await response.json()
export default function CreateBlogPostForm() {
 const handleSubmit = async e \Rightarrow \{
    e.preventDefault()
   const post = await create({
      title: e.target.title.value,
      content: e.target.content.value
    })
   window.location.href = `/blog/${post.slug}`
 return (
   <form onSubmit={handleSubmit}>
      <input type="text" name="title" />
     <textarea name="content" />
      <button type="submit">Submit
    </form>
```





</bar>

hx-trigger="click"
hx-swap="innerHTML"
hx-indicator="#spinner"
hx-get="/random">

<button

Click me!

7 (9 0) 6 (9 6) 6 (9 6) 7 (9 6)

Criando um framework React 19

React Server Components



React Server Components API

```
express()
    .get("/*", async (req, res) \Rightarrow {
    let root = (await import(resolve("build/app", `.${req.path}/page.js`))).default(req.query)
    renderToPipeableStream(root, moduleBaseURL).pipe(res)
})
    .post("/*", bodyParser.text(), async (req, res) \Rightarrow {
        const actionReference = String(req.headers["rsa-reference"])
        const actionOrigin = String(req.headers["rsa-origin"])

const [filepath, name] = actionReference.split("#")
        const action = (await import(`.${resolve(filepath)}`))[name]

let args = await decodeReply(req.body, moduleBaseURL)
```

const root = (await import(resolve("build/app", `.\${actionOrigin}/page.js`))).default(req.query)

const returnValue = await action.apply(null, args)

})

.listen(3001)

renderToPipeableStream({ returnValue, root }, moduleBaseURL).pipe(res)

Server-side Rendering API

```
express()
    .get("/*", async (req, res) ⇒ {
    const url = new URL(req.url, `http://${req.headers.host}`)
    url.port = "3001" // Forward to the RSC server

return http.get(url, async rsc ⇒ {
    let Root = () ⇒ use(createFromNodeStream(rsc, resolve("build/") + "/", moduleBaseURL))
    const Layout = (await import(resolve("build/_layout"))).default

renderToPipeableStream(createElement(Layout, { children: createElement(Root) }), {
    bootstrapModules: ["/build/_client.js"],
    importMap: { ... }
    }).pipe(res)
```

}) })

.listen(3000)

E o que sobra pro **Client** ?

```
const callServer = async (id: string, args: unknown[]) ⇒
   (
      await createFromFetch(
      fetch(getUrl(), {
         method: "POST",
         body: await encodeReply(args),
         headers: { "rsa-origin": location.pathname, "rsa-reference": id }
```

const Shell: FC<{ data: ReactNode }> = ({ data }) ⇒ use(data)

const data = createFromFetch(fetch(getUrl()), { callServer, moduleBaseURL })

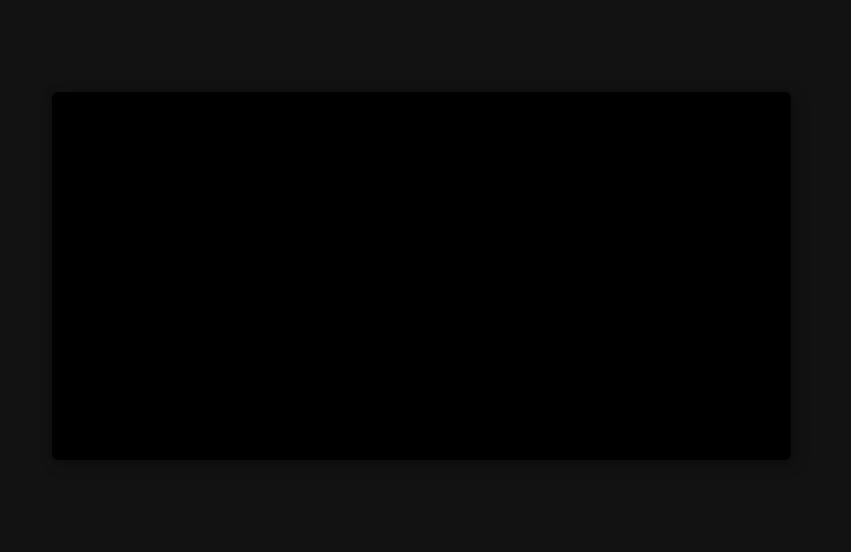
hydrateRoot(document.getElementById("root")!, createElement(Shell, { data }))

{ callServer, moduleBaseURL }

}),

).returnValue

Isso funciona?



Mas e o código?

```
export const createReference = (e: string, path; string, directive; string) \Rightarrow {
                                                                                                                            // Server-side Rendering
 const id = '/${relative(".", path).replace("src", "build").replace(\\..+$/, ".js")}#${e}
                                                                                                                            express()
  const mod = '${e == "default" ? parse(path).base.replace(/\..+$/, "") : ""}_${e}
  return directive == "server"
   ? `const ${mod}=()⇒{throw new Error("This function is expected to only run on the server")};${mod}.$$typeof=Symbol.1
      `${e == "default" ? "export default {" : `export const ${e} = {`}$$typeof:Symbol.for("react.client.reference"),$$i
                                                                                                                               .get("/*", async (req, res) ⇒
                                                                                                                                renderToPipeableStream(
await Bun. $`rm -rf ./build/
const entries = (await readdir(resolve("src"), { recursive: true })).reduce(
                                                                                                                                  "/build/"
  (acc. file) \Rightarrow {
                                                                                                                                ).pipe(res)
   const ext = file.match(/\..+$/)
   if (!ext) return acc
                                                                                                                               .post("/*", bodyParser.text(), async (reg. res) ⇒ {
   const path = resolve("src", file)
   if (file.endsWith("page.tsx")) acc["pages"].push(path)
                                                                                                                                const returnValue = await (
   else if (ext[0].match(/\.tsx?$/)) acc["components"].push(path)
                                                                                                                                 await import(`.${resolve(filepath)}`)
   else acc["assets"].push(path)
   return acc
                                                                                                                                const root = (
  { pages: [], components: [], assets: [] } as Record<string, string[]>
                                                                                                                                 ).default(req.query)
                                                                                                                               .listen(3001)
await Bun.build({
  target: "bun"
                                                                                                                            // RSC Server
  entrypoints: entries["pages"],
                                                                                                                            express()
  external: ["react", "react-dom"],
                                                                                                                               .use("/build". express.static("build"))
  outdir: resolve("build", "app").
                                                                                                                               .use("/node_modules", express.static("node_modules"))
  pluains: [
                                                                                                                               .get(/\.(?!is).+$/. express.static("build"))
                                                                                                                               .get("/*", async (req, res) \Rightarrow {
      name: "rsc-register".
      setup: build ⇒
       build.onLoad({ filter: /\.tsx?$/ }, async args ⇒ {
         const content = await Bun.file(args.path).text()
                                                                                                                                  renderToPipeableStream(
          const directives = content.match(/(?:^|\n|;)"use (client|server)";?/)
          if (!directives) return { contents: content }
          const { exports } = new Bun.Transpiler({ loader: "tsx" }).scan(content)
                                                                                                                                      bootstrapModules: ["/build/ client.is"].
          if (exports.length == 0) return { contents: content }
                                                                                                                                      importMap: {
          return { contents: exports.map(e ⇒ createReference(e, args.path, directives[1])).join("\n") }
                                                                                                                                        imports: +
                                                                                                                                          react: "https://esm.sh/react@next"
                                                                                                                                           "@physis/react-server-dom-esm/client"
await Bun.build({
 target: "bun"
  external: ["react", "react-dom", "@physis/react-server-dom-esm"],
  entrypoints: entries["components"].
  root: resolve("src")
                                                                                                                                   ).pipe(res)
  outdir: resolve("build")
entries["assets"].forEach(asset ⇒ Bun.write(asset.replace("src", "build"), Bun.file(asset)))
                                                                                                                               .listen(3000)
```

```
res.header("Access-Control-Allow-Origin", "*").header("Access-Control-Allow-Headers", "*"), next()
  (await import(resolve("build/app", `.${req.path}/page.js`))).default(req.query),
const [filepath. name] = String(reg.headers["rsa-reference"]).split("#")
)[name].apply(null, await decodeReply(reg.body, "/build/"))
  await import(resolve("build/app", `.${String(req.headers["rsa-origin"])}/page.js`))
renderToPipeableStream({ returnValue, root }, "/build/").pipe(res)
const url = new URL(reg.url, `http://${reg.headers.host}`)
return http.get(((url.port = "3001"), url), async rsc => -
 let Root = () ⇒ use(createFromNodeStream(rsc, resolve("build/") + "/", "/build/"))
   createElement((await import(resolve("build/_layout"))).default, { children: createElement(Root) }),
          "react-dom": "https://esm.sh/react-dom@next"
          "react-dom/": "https://esm.sh/react-dom@next/".
            "/node_modules/@physis/react-server-dom-esm/esm/react-server-dom-esm-client.browser.development.js"
```

E as...

Linhas??

TypeScript	3	6	1	99

blank

comment

code

files

Language





