R19

Next.js do 0

Ou;

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

O que é o `React 19`?

- Novos hooks e APIs
- React Server Components
- React Server Actions
- Compilador experimental 🤯

Diferenças de `Server` / `Client` Components?

	Renderiza no servidor?	Renderiza no cliente?
`Server Components`	✓	✗
`Client Components`	✓*	✓

O que são `Server Actions`?

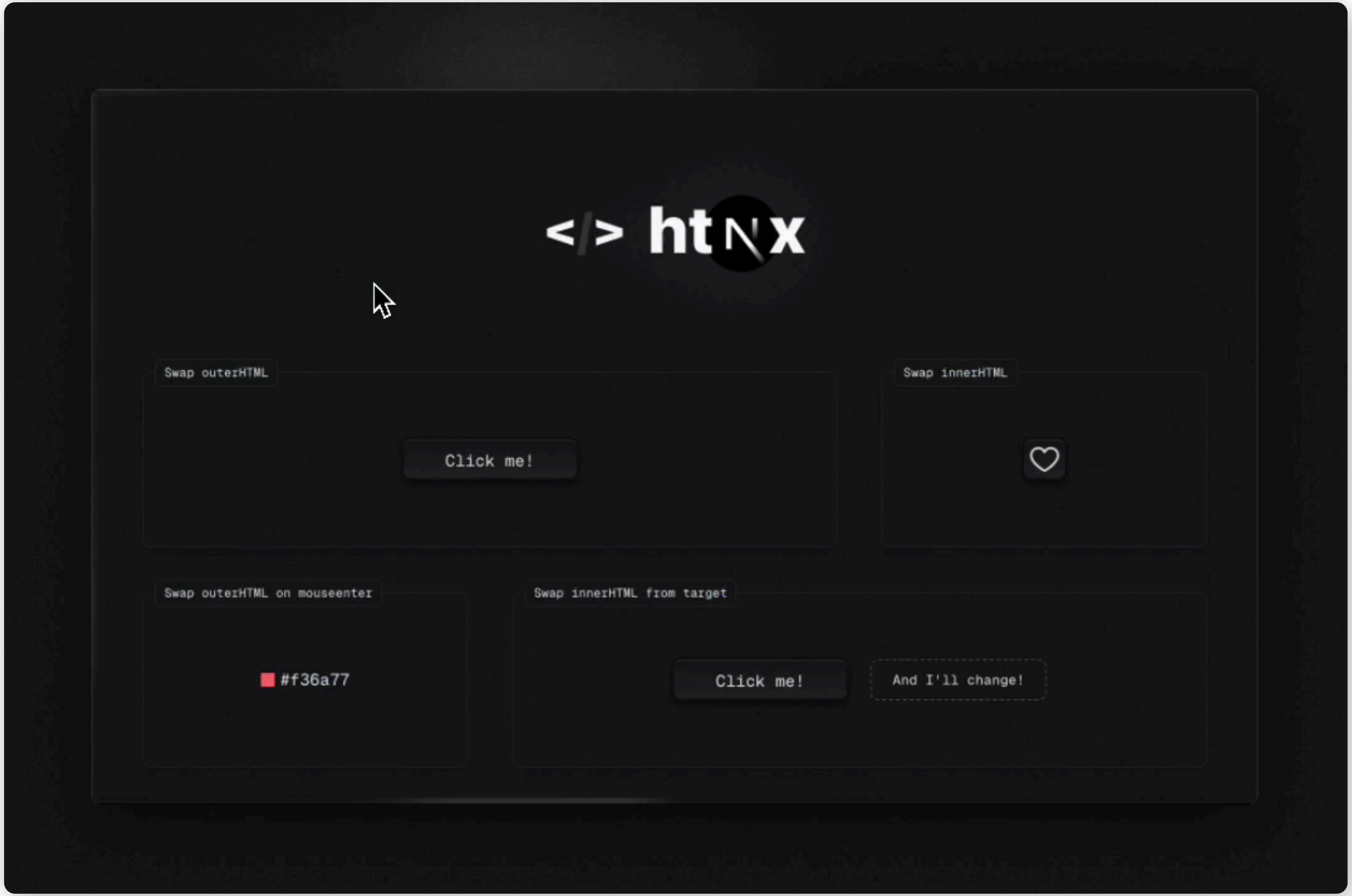
Funções que rodam no servidor e retornam algo para o cliente.

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 ⇒ {
    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</button>
    </form>
  )
}
```

Um pouco de contexto;



</> htmx
Swap outerHTML
Click me!
Swap innerHTML
Swap outerHTML on mouseenter
#f36a77
Swap innerHTML from target
Click me!
And I'll change!

```html
<button
  hx-trigger="click"
  hx-swap="innerHTML"
  hx-indicator="#spinner"
  hx-get="/random">
  Click me!
</button>
```

Agora para o que interessa;

Criando um framework React 19

React Server Components ~~*Server*~~

•`_´•

React Server Components *API*

```
express()
  .get("/*", async (req, res) ⇒ {
    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) ⇒ {
    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 returnValue = await action.apply(null, args)

    const root = (await import(resolve("build/app", `.${actionOrigin}/page.js`))).default(req.query)
    renderToPipeableStream({ returnValue, root }, moduleBaseURL).pipe(res)
  })
  .listen(3001)
```

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 }
      }),
      { callServer, moduleBaseURL }
    )
  ).returnValue

const data = createFromFetch(fetch(getUrl()), { callServer, moduleBaseURL })
const Shell: FC<{ data: ReactNode }> = ({ data }) ⇒ use(data)

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

Bem legal, mas...

Isso funciona?

Mas e o código?

```
export const createReference = (e: string, path: string, directive: string) => {
  const id = `/${relative(".", path).replace("src", "build").replace(/\..+$/, ".js")}#${e}`
  const mod = `${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}.$$typeof=Symbol.f
    : `${e === "default" ? "export default {" : `export const ${e} = {`}$$typeof:Symbol.for("react.client.reference"),$$i
}

// Entries
await Bun.$`rm -rf ./build/`
const entries = (await readdir(resolve("src"), { recursive: true })).reduce(
  (acc, file) => {
    const ext = file.match(/\..+$/)
    if (!ext) return acc
    const path = resolve("src", file)
    if (file.endsWith("page.tsx")) acc["pages"].push(path)
    else if (ext[0].match(/\.tsx?$/)) acc["components"].push(path)
    else acc["assets"].push(path)
    return acc
  },
  { pages: [], components: [], assets: [] } as Record<string, string[]>
)

// Building + Bundling
await Bun.build({
  target: "bun",
  entrypoints: entries["pages"],
  external: ["react", "react-dom"],
  outdir: resolve("build", "app"),
  plugins: [
    {
      name: "rsc-register",
      setup: build =>
        build.onLoad({ filter: /\.tsx?$/ }, async args => {
          const content = await Bun.file(args.path).text()
          const directives = content.match(/(?:^|\n|;)"use (client|server)";?/)
          if (!directives) return { contents: content }
          const { exports } = new Bun.Transpiler({ loader: "tsx" }).scan(content)
          if (exports.length === 0) return { contents: content }
          return { contents: exports.map(e => createReference(e, args.path, directives[1])).join("\n") }
        })
    }
  ]
})
await Bun.build({
  target: "bun",
  external: ["react", "react-dom", "@physis/react-server-dom-esm"],
  entrypoints: entries["components"],
  root: resolve("src"),
  outdir: resolve("build")
})
entries["assets"].forEach(asset => Bun.write(asset.replace("src", "build"), Bun.file(asset)))
```

```
// Server-side Rendering
express()
  .use(
    (_, res, next) => (
      res.header("Access-Control-Allow-Origin", "*").header("Access-Control-Allow-Headers", "*"), next()
    )
  )
  .get("/*", async (req, res) =>
    renderToPipeableStream(
      (await import(resolve("build/app", `.${req.path}/page.js`))).default(req.query),
      "/build/"
    ).pipe(res)
  )
  .post("/*", bodyParser.text(), async (req, res) => {
    const [filepath, name] = String(req.headers["rsa-reference"]).split("#")
    const returnValue = await (
      await import(`.${resolve(filepath)}`)
    )[name].apply(null, await decodeReply(req.body, "/build/"))
    const root = (
      await import(resolve("build/app", `.${String(req.headers["rsa-origin"])}/page.js`))
    ).default(req.query)
    renderToPipeableStream({ returnValue, root }, "/build/").pipe(res)
  })
  .listen(3001)

// RSC Server
express()
  .use("/build", express.static("build"))
  .use("/node_modules", express.static("node_modules"))
  .get(/\.(?!js).+$/, express.static("build"))
  .get("/*", async (req, res) => {
    const url = new URL(req.url, `http://${req.headers.host}`)
    return http.get(((url.port = "3001"), url), async rsc => {
      let Root = () => use(createFromNodeStream(rsc, resolve("build/") + "/", "/build/"))
      renderToPipeableStream(
        createElement((await import(resolve("build/_layout"))).default, { children: createElement(Root) }),
        {
          bootstrapModules: ["/build/_client.js"],
          importMap: {
            imports: {
              react: "https://esm.sh/react@next",
              "react-dom": "https://esm.sh/react-dom@next",
              "react-dom/": "https://esm.sh/react-dom@next/",
              "@physis/react-server-dom-esm/client":
                "/node_modules/@physis/react-server-dom-esm/esm/react-server-dom-esm-client.browser.development.js"
            }
          }
        }
      ).pipe(res)
    })
  })
  .listen(3000)
```

Ainda falta alguma coisa...

E as...

Linhas??

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                       3              6              1             99
-------------------------------------------------------------------------------
```

Obrigado!