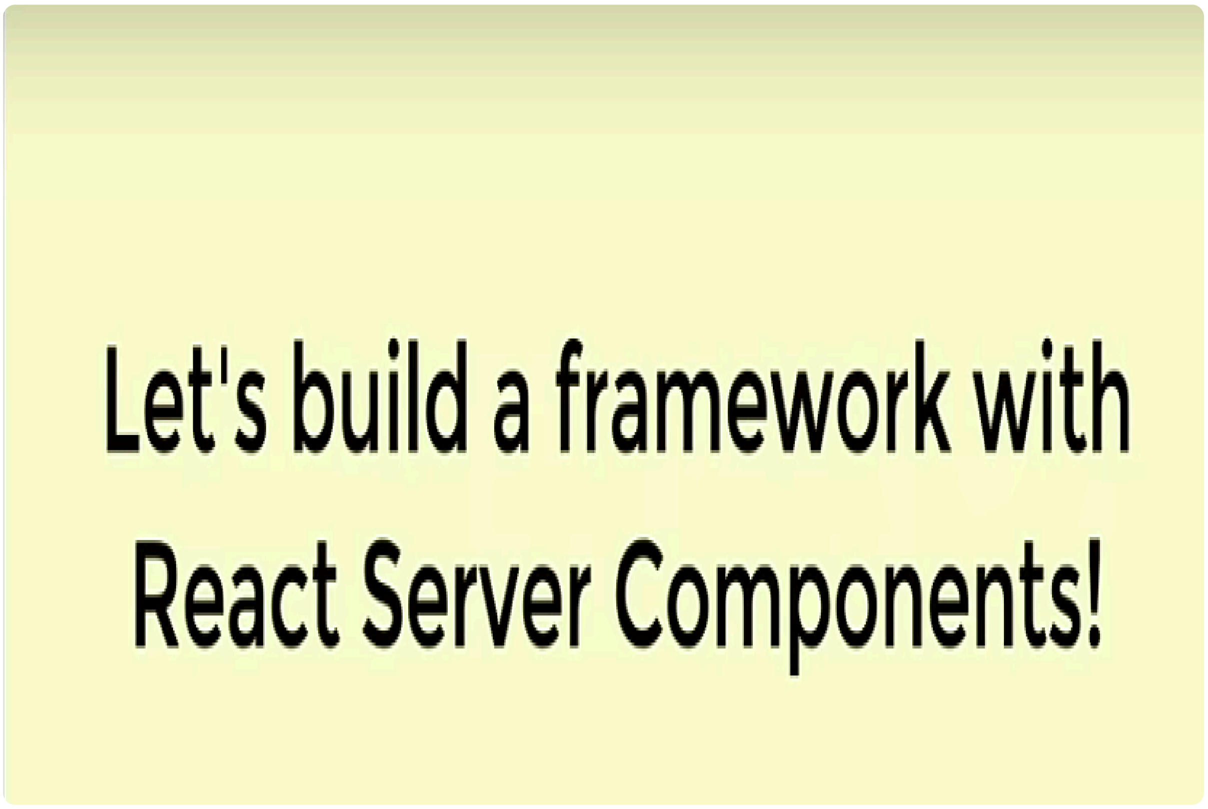
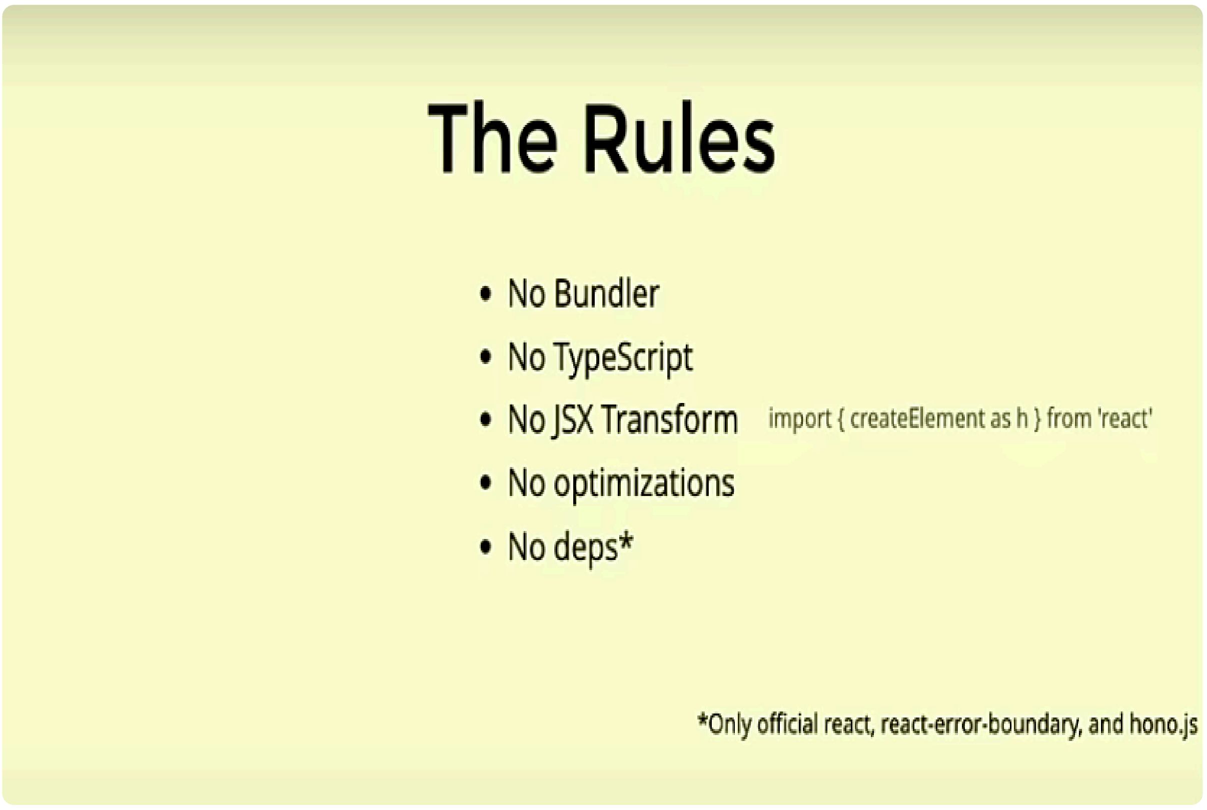
📑 And Now You Understand RSC



Let's build a framework with React Server Components!



The Rules

- No Bundler
- No TypeScript
- No JSX Transform `import { createElement as h } from 'react'`
- No optimizations
- No deps*

*Only official react, react-error-boundary, and hono.js

What we're building

- **API:** to fetch and render Server Components
- **Bundler:** to bundle and render Client Components*
- **Router:** to route on the client and fetch updated Server Components

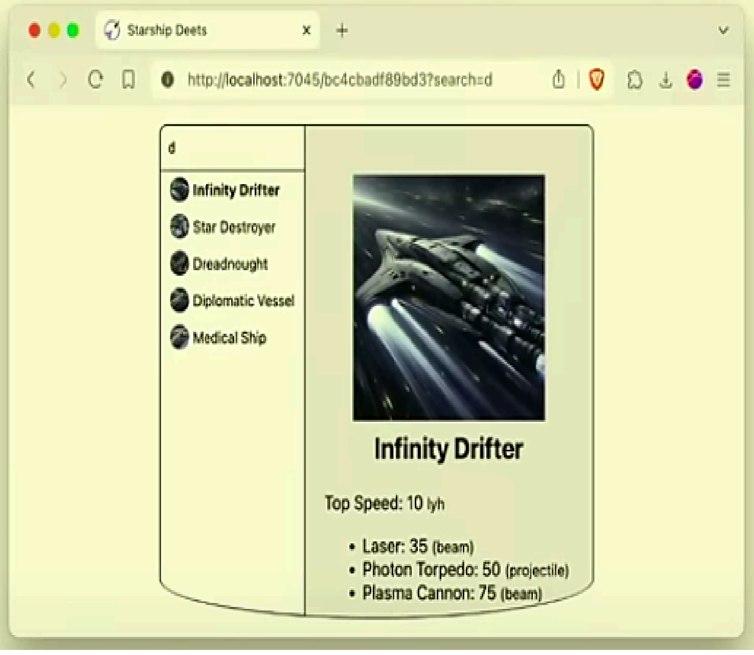
*It's not really a bundler, but a node loader (runtime module transformer)

The App

SPA

No SSG

No client router yet



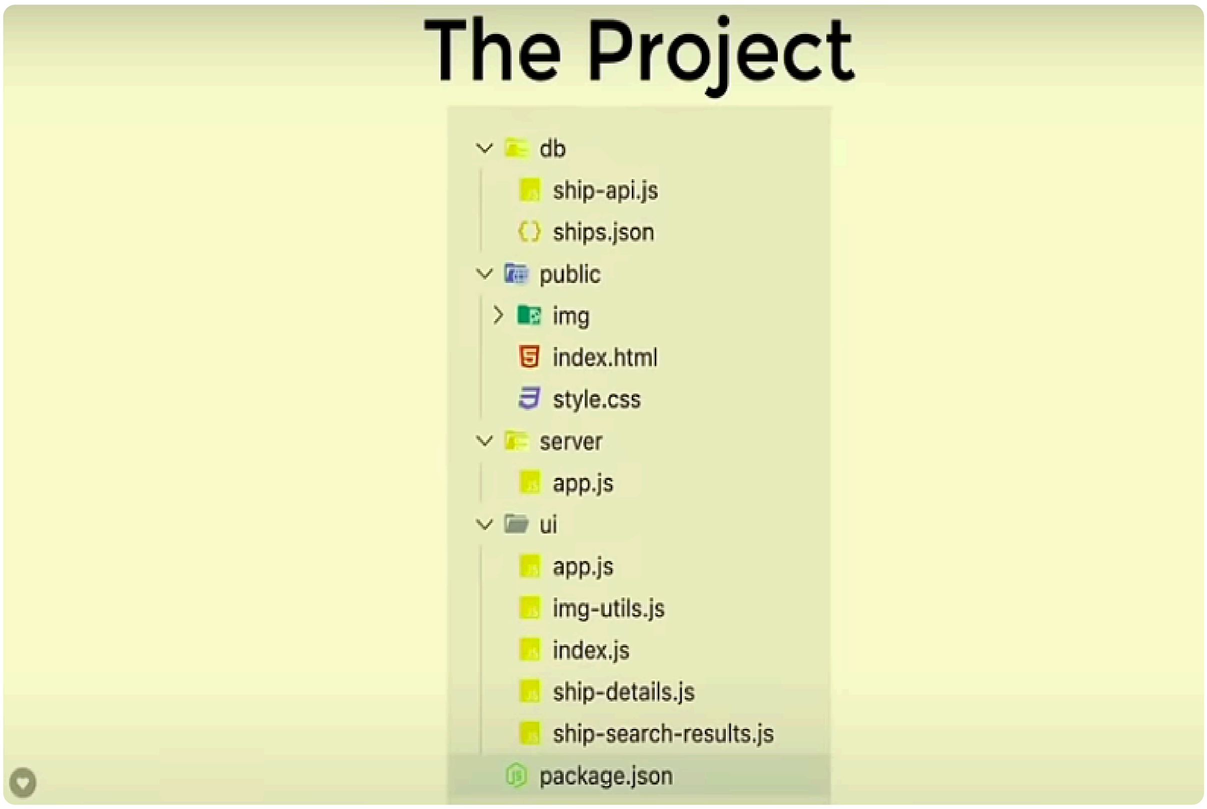
d

- Infinity Drifter
- Star Destroyer
- Dreadnought
- Diplomatic Vessel
- Medical Ship

Infinity Drifter

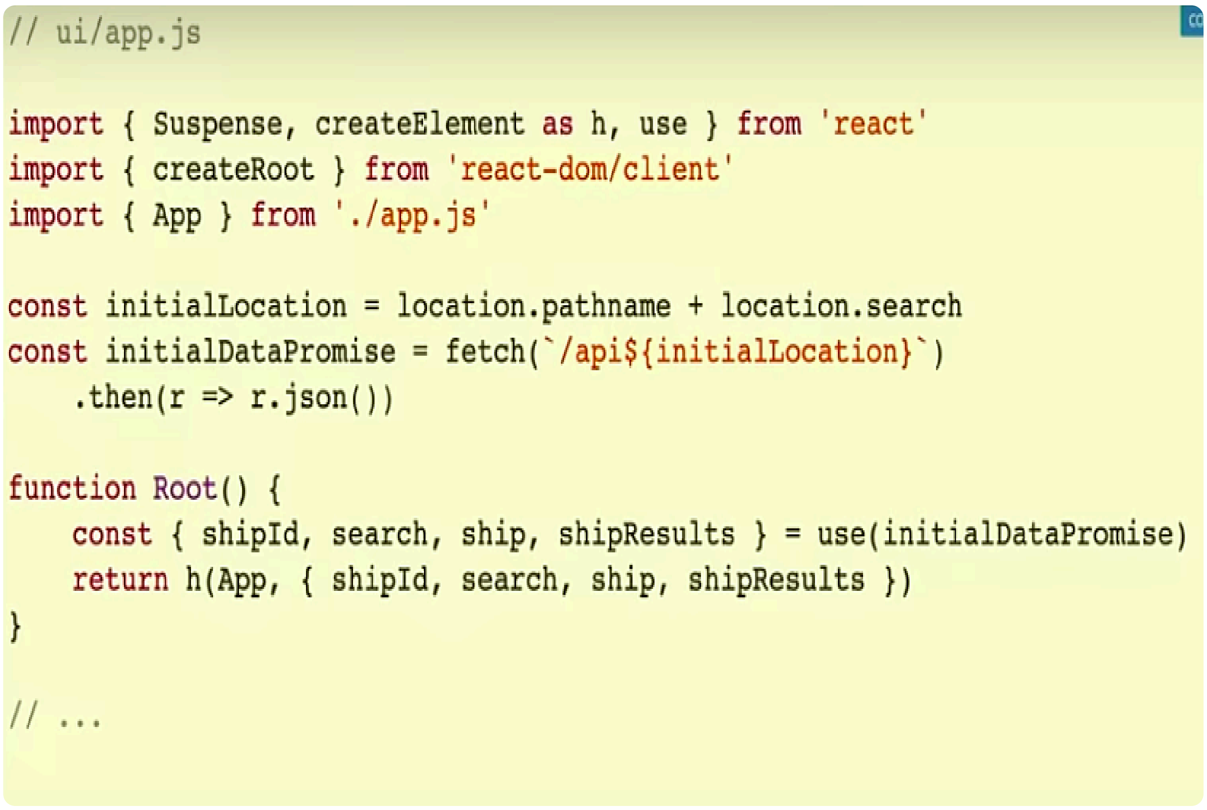
Top Speed: 10 lyh

- Laser: 35 (beam)
- Photon Torpedo: 50 (projectile)
- Plasma Cannon: 75 (beam)



The Project

```
db
  ship-api.js
  ships.json
public
  img
  index.html
  style.css
server
  app.js
ui
  app.js
  img-utils.js
  index.js
  ship-details.js
  ship-search-results.js
package.json
```



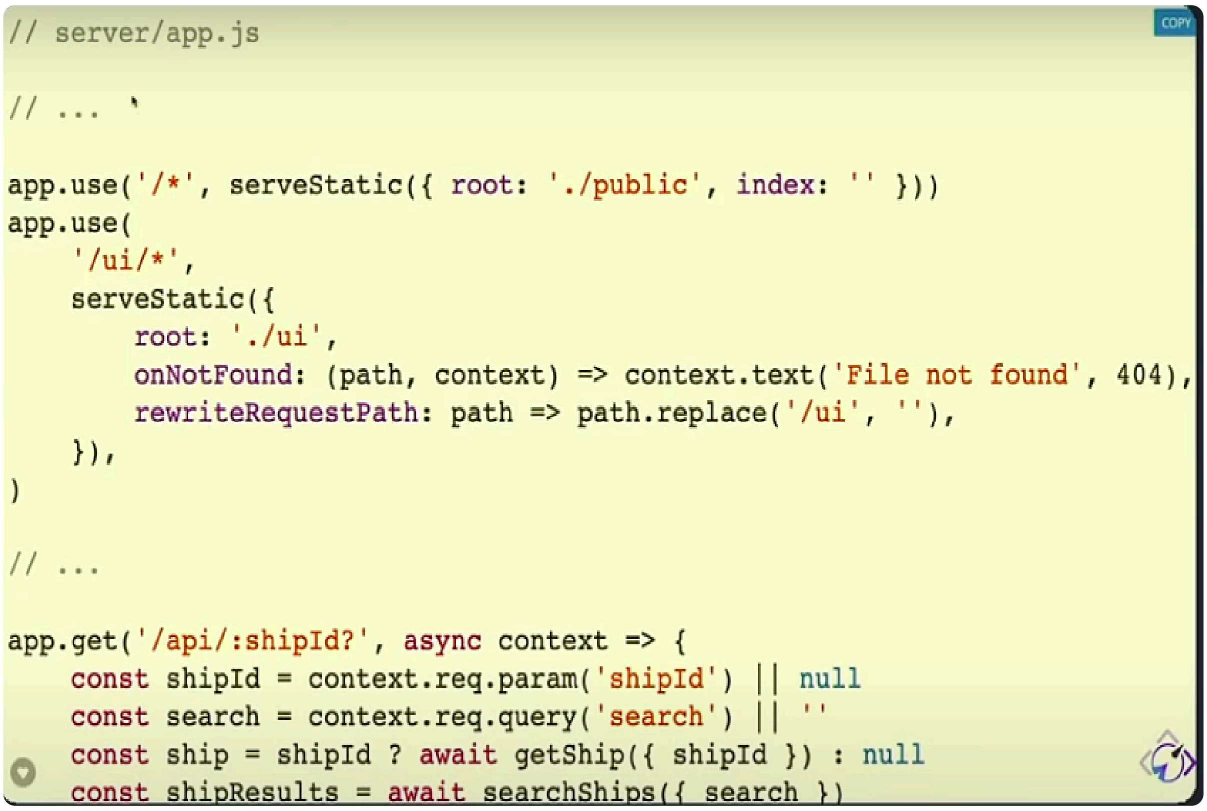
```
// ui/app.js

import { Suspense, createElement as h, use } from 'react'
import { createRoot } from 'react-dom/client'
import { App } from './app.js'

const initialLocation = location.pathname + location.search
const initialDataPromise = fetch(`/api${initialLocation}`)
    .then(r => r.json())

function Root() {
    const { shipId, search, ship, shipResults } = use(initialDataPromise)
    return h(App, { shipId, search, ship, shipResults })
}

// ...
```



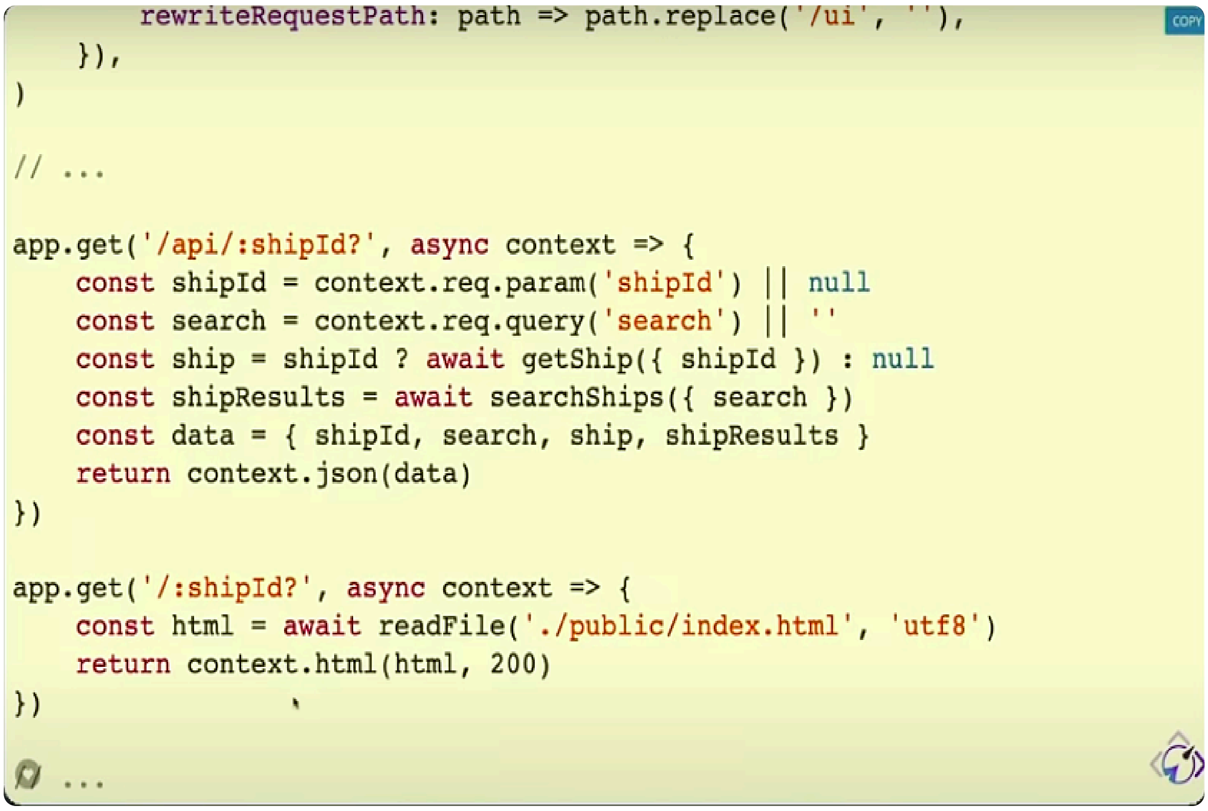
```
// server/app.js

// ...

app.use('/*', serveStatic({ root: './public', index: '' }))
app.use(
    '/ui/*',
    serveStatic({
        root: './ui',
        onNotFound: (path, context) => context.text('File not found', 404),
        rewriteRequestPath: path => path.replace('/ui', ''),
    }),
)

// ...

app.get('/api/:shipId?', async context => {
    const shipId = context.req.param('shipId') || null
    const search = context.req.query('search') || ''
    const ship = shipId ? await getShip({ shipId }) : null
    const shipResults = await searchShips({ search })
```



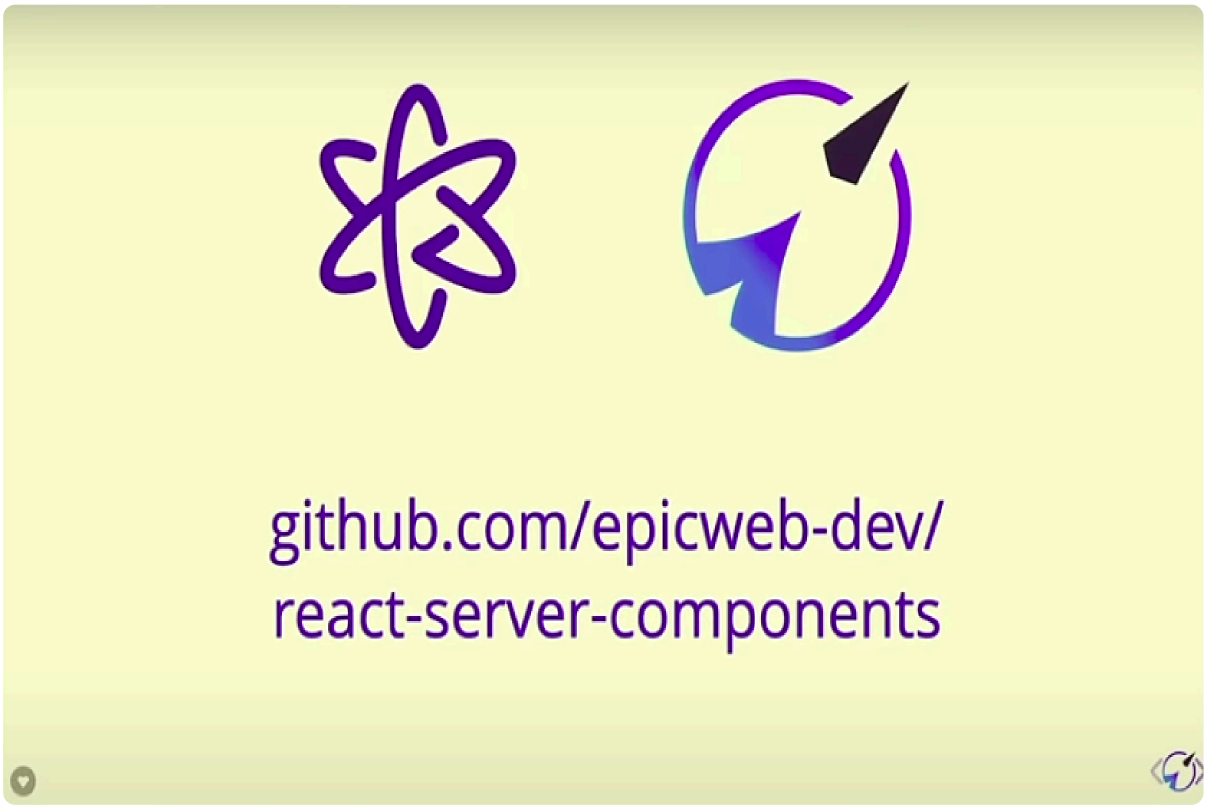
```
        rewriteRequestPath: path => path.replace('/ui', ''),
    }),
)

// ...

app.get('/api/:shipId?', async context => {
    const shipId = context.req.param('shipId') || null
    const search = context.req.query('search') || ''
    const ship = shipId ? await getShip({ shipId }) : null
    const shipResults = await searchShips({ search })
    const data = { shipId, search, ship, shipResults }
    return context.json(data)
})

app.get('/:shipId?', async context => {
    const html = await readFile('./public/index.html', 'utf8')
    return context.html(html, 200)
})

// ...
```



github.com/epicweb-dev/
react-server-components