3 Lazy loading & Code splitting

- ➔ Tìm hiểu thêm về Virtual DOM
- ➔ React.lazy & Suspense
- ➔ Code splitting
- ➔ Bundle size

Tìm hiểu thêm về Virtual DOM

Trước React

- DOM API là imperative
- Dev phải quan tâm về cách update DOM
- Dev phải code cả 2 thứ: quản lý state và update UI
- Không có 1 thư viện hay chuẩn nào đủ sức để hệ thống hoá vấn đề này
- Backbone, EmberJS, Angular 1, …

React

- Declarative code
- Dev không phải quan tâm về cách update DOM
- Dev chỉ code 1 thứ: quản lý app state
- Cách làm hiện tại của React là Virtual DOM:
  - 1 object thể hiện DOM tree
  - So sánh giữa các state object để biết update DOM tree ở đâu (React reconciler)

Suspense

Suspense có gì vui

- Cơ chế chờ của React
- Catch promise được throw và render fallback component
- Giống componentDidCatch mà cho promise thay vì error

```
const Component = () => {
  if (loading) {
    return <div>loading...</div>
  }

  return <OtherComponent />
}
```

```
const Component = () => (
  <Suspense
    fallback={<div>loading...</div>}
  >
    <OtherComponent />
  </Suspense>
)
```

Sử dụng Suspense

- Để quản lý loading state, bao gồm code loading và data loading
- Hiện tại thì Suspense chưa thể được sử dụng trong SSR (support trong React 18)
- Sử dụng Suspense cho data loading đang được chuẩn hoá dần (link ở cuối bài)
- Giảm dung lượng main chunk và lazy load code

React.lazy

- Tạo ra một component từ dynamic import
- Nhận vào 1 hàm trả về promise cho 1 dynamic import
- Chỉ chạy được cho default import

```
import { Suspense, lazy } from 'react'

const LazyComponent = lazy(() => import('./Component'))

const App = () => {
  return (
    <Suspense fallback={<div>Loading...</div>}>
      <LazyComponent />
    </Suspense>
  )
}
```

Code splitting

Code splitting

- Để giảm initial bundle size
- Based on Routes
- Based on Components
- Thư viện lớn chưa cần sử dụng

Route based code splitting

- Chia code theo page
- lazy load từng page
- và Suspense trên Routes

Bundle size

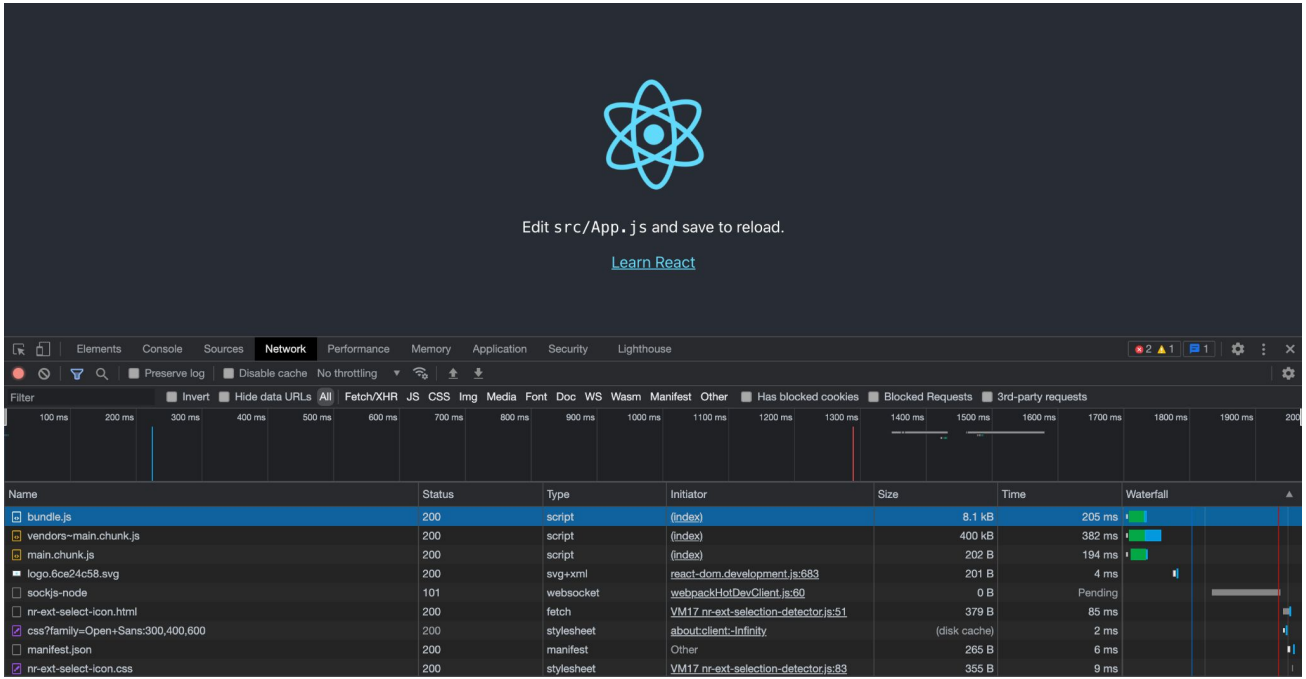
Nội dung

- Bundle Size là gì ?
- Tại sao chúng ta phải quan tâm ?
- Các công cụ hỗ trợ ?
- Làm sao để tối ưu Bundle Size ?

200Lab Education

Bundle size là gì ?

Bundle size là kích thước của những file Javascript mà Browser (Trình duyệt) tải xuống khi User truy cập một Website bất kỳ từ Browser.



Tại sao chúng ta phải quan tâm ?

Bundle size lớn sẽ khiến Browser (Trình duyệt) bị chậm vì mất thời gian để download Javascript của Website .

-> Trải nghiệm của User không được tốt

Các công cụ hỗ trợ

Sử dụng các công cụ hỗ trợ để phân tích các thành phần bên trong Bundle, sau khi phân tích các công cụ sẽ cho chúng ta cái nhìn tổng quan về kích thước của các thành phần trong Bundle. Từ đó chúng ta sẽ dễ dàng quản lý và tối ưu.

❖ Các công cụ hỗ trợ :

- Webpack-bundle-analyzer
  - Link: https://webpack.js.org/
- Source-map-explorer
  - Link: https://www.npmjs.com/package/source-map-explorer

Làm sao để tối ưu Bundle Size ?

- Code-splitting (Lazy Load, Suspense, chia nhỏ component)
- Sử dụng Import từng phần, không nên Import toàn bộ Library
- Nếu chỉ sử dụng một phần nhỏ trong Library có thể xem xét để tự code phần đó.

Bài tập

- Analyze Bundle size của 1 dự án mình đang làm => cách tối ưu hoá

Link tham khảo

Đọc vui:

- https://reactjs.org/docs/concurrent-mode-suspense.html
- https://reactjs.org/blog/2019/11/06/building-great-user-experiences-with-concurrent-mode-and-suspense.html
- https://nitayneeman.com/posts/introducing-dynamic-imports-in-ecmascript-2020/
- https://blog.jakoblind.no/code-split-vendors-with-webpack-for-faster-load-speed/