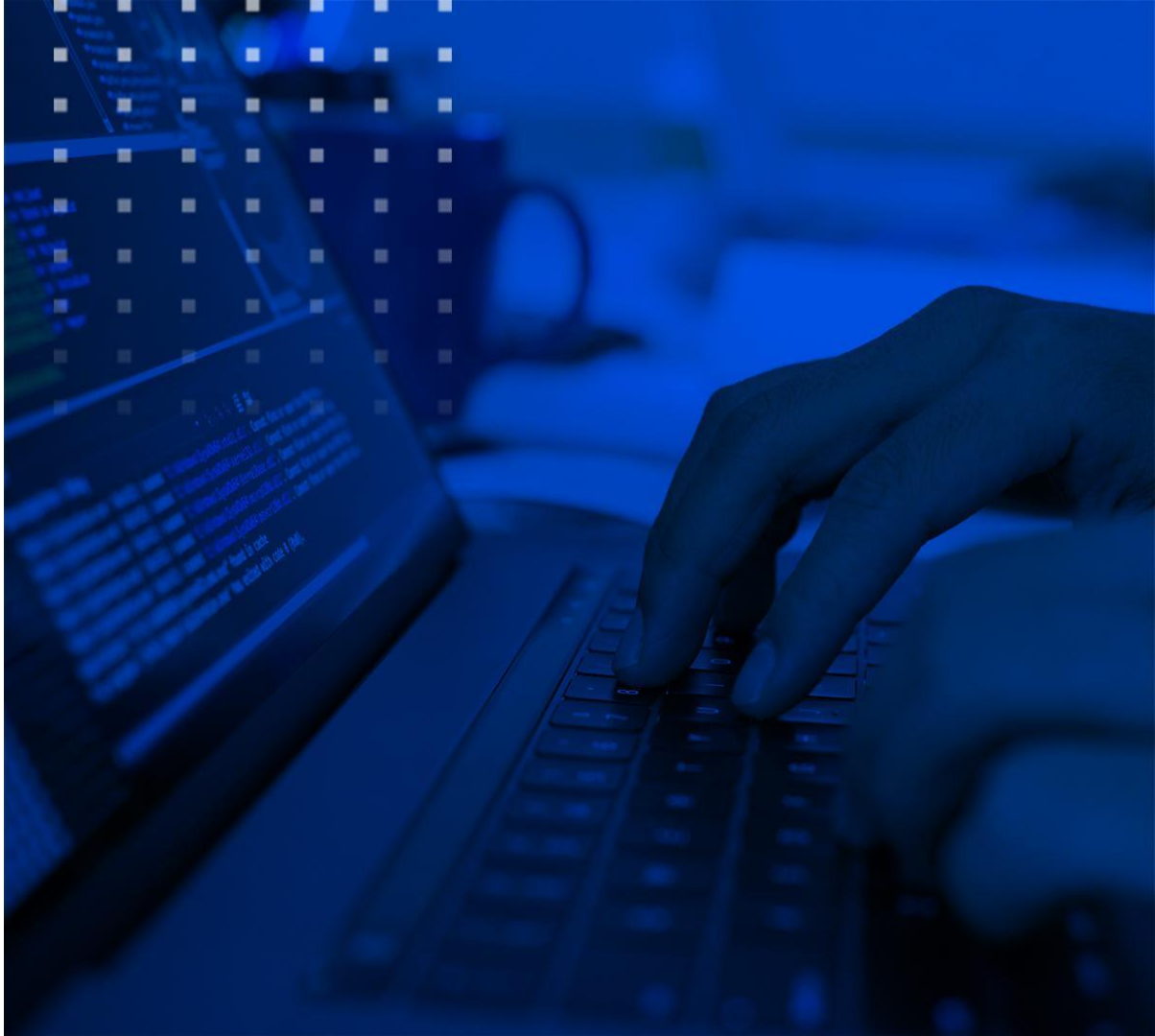
incluyeme.com

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

CURSO

FULL STACK DEVELOPER
NIVEL INICIAL



UNIDAD 20
JavaScript
Routing y estados

ESTRUCTURA DE LA APP

ESTRUCTURA

En la clase anterior definimos todos los componentes básicos que va a tener nuestra app...

ESTRUCTURA

En la clase anterior definimos todos los componentes básicos que va a tener nuestra app...

¡Ahora hay que **añadirle el contenido**!

Trabajando sobre el componente home vamos a armar nuestra estructura JSX (HTML) y CSS para empezar a darle forma.

ROUTING

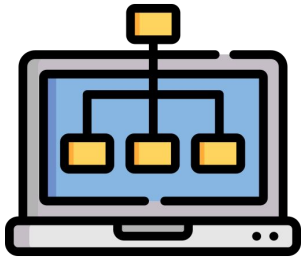
ROUTING

Tenemos que empezar a marcar un camino en nuestra app...

ROUTING

Es importante mantener

una buena **navegabilidad**

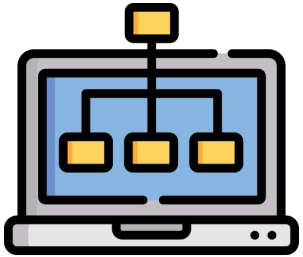


ROUTING

Por ello tenemos que definir

cómo se va a navegar por

nuestra app...



ROUTING

Por defecto React **no viene con un mecanismo integrado de navegación**.

Esto es para mantener sus **dependencias al mínimo** y dado que no todo proyecto necesita routing, se maneja como una dependencia aparte.

ROUTING

Por eso es que vamos a utilizar el módulo NPM **react-router-dom**

Instalar el paquete React Router utilizando con el comando **npm install react-router-dom**

```
npm install  react-router-dom
```

Luego importar los módulos correspondientes en nuestra app.js: **import { BrowserRouter, Route, Switch } from "react-router-dom";**

```
import { BrowserRouter, Route, Link, Switch } from "react-router-dom";
```

Envolver toda la app con el **BrowserRouter**

```
<BrowserRouter>
  <Switch>

    <Route exact path="/">
      <Intro text="Hola :)" />
    </Route>

  </Switch>
</BrowserRouter>
```

Luego, a partir de donde empiezan las vistas (home, contacto) añadir un **Switch**

```
<BrowserRouter>
  <Switch>

    <Route exact path="/">
      <Home />
    </Route>

    <Route exact path="/">
      <Contacto />
    </Route>

  </Switch>
</BrowserRouter>
```

Por último crear los **Route** de cada vista añadiendo el **path** correspondiente

```
<BrowserRouter>
  <Switch>

    <Route exact path="/">
      <Home />
    </Route>

    <Route exact path="/contacto">
      <Contacto />
    </Route>

  </Switch>
</BrowserRouter>
```

NAVEGACIÓN

NAVEGACIÓN

Para navegar a una ruta o “hacer links” no vamos a utilizar los tags **<a>** con el atributo **href** como venimos haciendo.

NAVEGACIÓN

Para navegar a una ruta o “hacer links” no vamos a utilizar los tags **<a>** con el atributo **href** como venimos haciendo.

Vamos a utilizar el elemento **Link** que instalamos junto con los demás en el React Router.

NAVEGACIÓN

```
<Link to="/contacto">Contacto</Link>
```

Al clickear en este elemento, el **BrowserRouter** se encarga de llevarnos al **componente** correspondiente según lo que le indicamos en el **switch**,

¡A Practicar!

1. Vamos a ir a React Bootrstrap > Navbars y en nuestro componente ya creado Navbar vamos a añadir este componente de Bootstrap y añadir los links que sean necesarios para navegar en nuestra app.

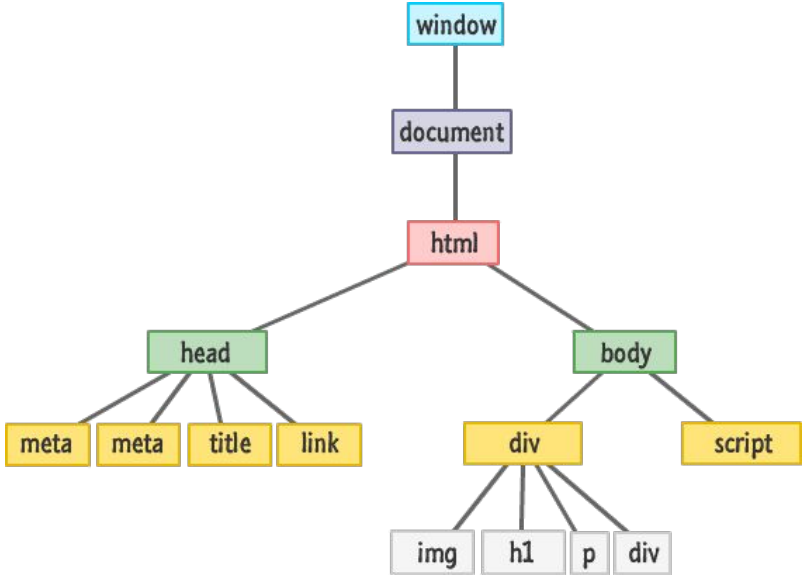
VIRTUAL DOM

VIRTUAL DOM

El DOM es la **representación de la interfaz gráfica** de nuestra aplicación. Por tanto, cada vez que el estado de la aplicación cambia, lo "esperable" es que también lo haga dicha interfaz para adaptarse a las modificaciones introducidas.

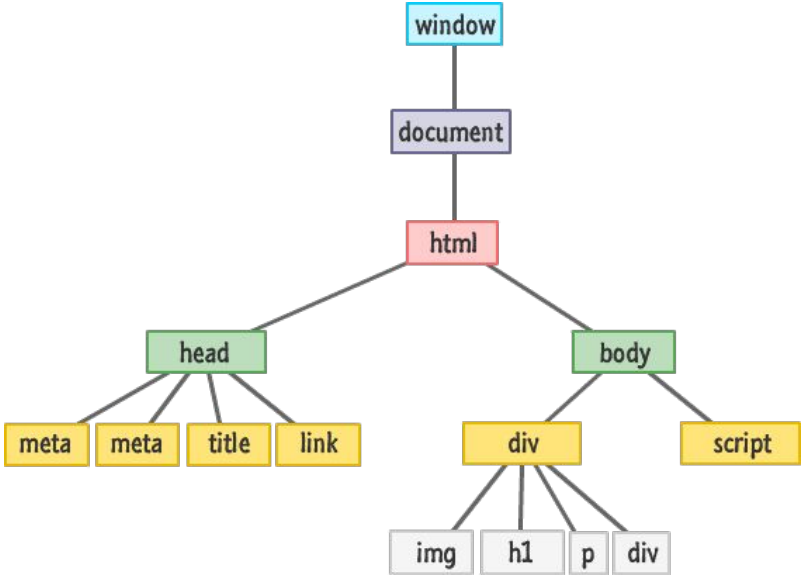
VIRTUAL DOM

Sin embargo, **actualizar el DOM es una tarea costosa** en cuanto a rendimiento se refiere, por lo que cuantos más cambios de estado sea necesario reflejar en él, más lento irá nuestra web.



VIRTUAL DOM

EL DOM posee una estructura en **forma de árbol**. Esto provoca que cada vez que modificamos un elemento dentro de él, todos sus hijos tengan que ser pintados de nuevo (hayan o no cambiado).



VIRTUAL DOM

El Virtual DOM es una **representación en memoria** del DOM real que actúa de **intermediario entre el estado de la aplicación y el DOM** de la interfaz gráfica que está viendo el usuario.

VIRTUAL DOM

En React, c**ada pieza de la UI es un componente** y **cada componente posee un estado interno**.
Este estado es observado por la React para **detectar sus cambios**. Así, cuando esto sucede, actualiza el árbol de su Virtual DOM y sigue el mismo proceso para trasladar los cambios resultantes a la interfaz presentada en el navegador.

VIRTUAL DOM

Gracias a esto, tiene un **rendimiento mejor** que las librerías que manipulan el DOM directamente, pues React **sólo actualiza aquellos objetos en los que ha detectado cambios** durante el proceso.

Ahora veamos, *¿qué son todos estos **estados**?*


[i]ncluyeme
.com


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES