



Clase 08. React JS

ROUTING Y NAVEGACIÓN

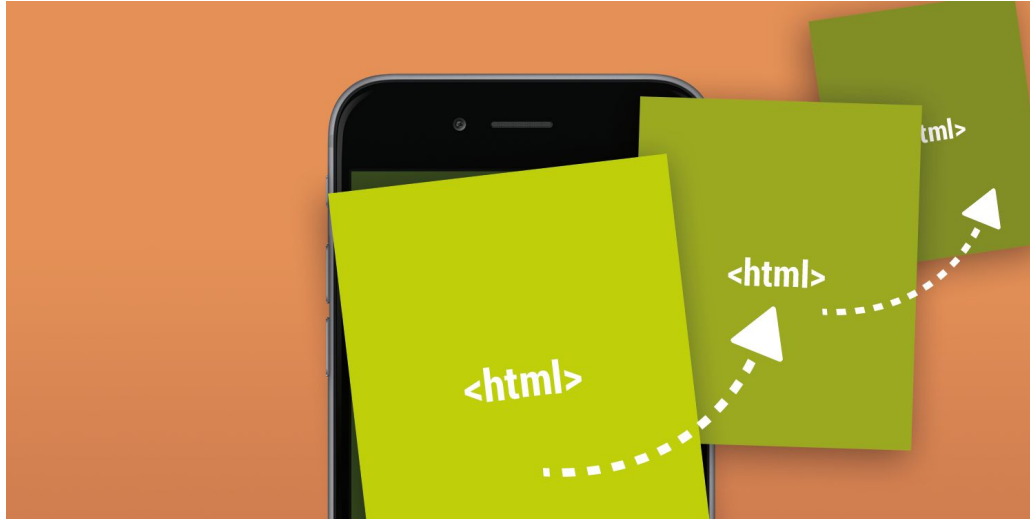


OBJETIVOS DE LA CLASE

- Organizar nuestra aplicación
- Configurar la navegabilidad entre componentes

ORGANICEMOS NUESTRA APP

La facilidad con la que nuestra aplicación permite agregar funcionalidades y navegarlas es un **factor clave** en términos de **experiencia** y **escalabilidad**





Buena navegabilidad permite a:

- **Users:** entender dónde están parados y guardar favs/marcadores a secciones en las que tienen interés
- **Navegadores:** Permitir controlar las acciones de ir adelante y volver, y conocer el nav history
- **Crawlers:** Entender la estructura de la app y proveer acceso optimizado/visibilizado a las distintas secciones

Organizando la app

Preguntémonos:

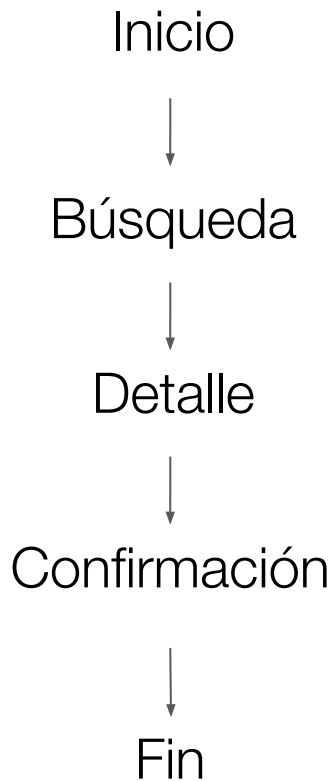
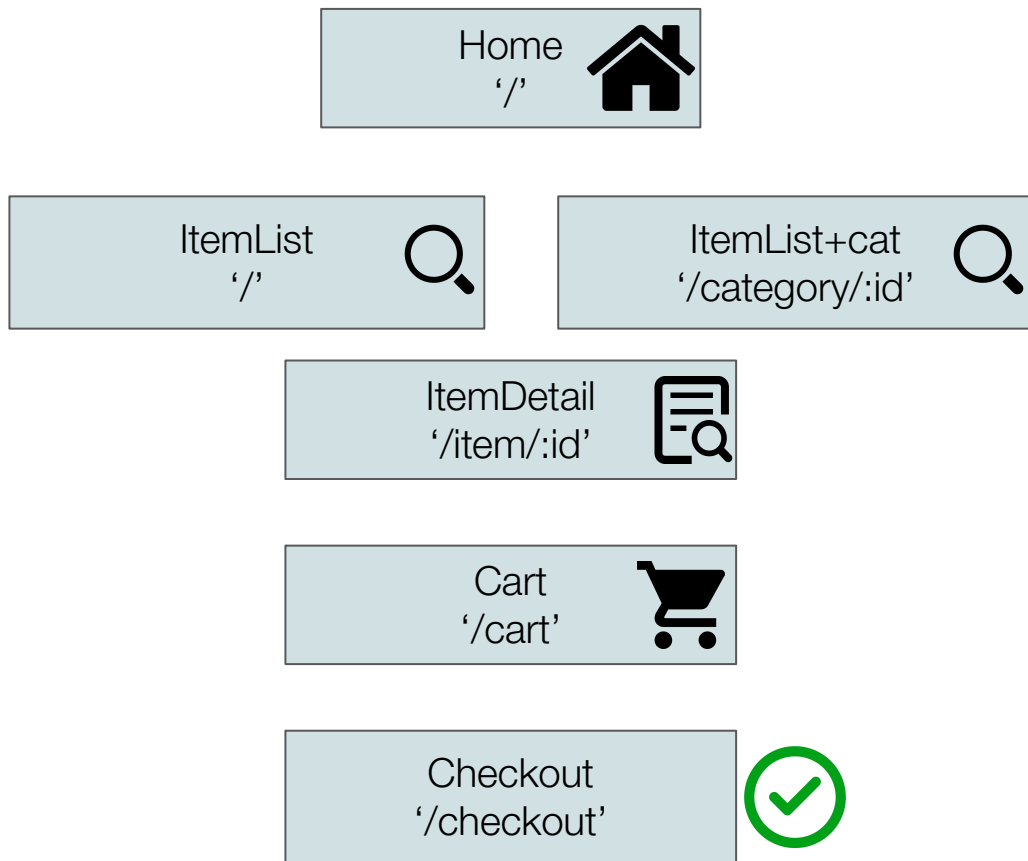
¿Cual es el punto de inicio de
nuestra app?



Organización



Routing



REACT ROUTER

Instalación

Instalación

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.

Hay varias soluciones, pero hoy instalaremos:

react-router-dom

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



Versiones react-router

- Si bien instalaremos **react-router-dom** publicado en NPM, veremos varias versiones
 - **react-router** => librería core (no instalar)
 - **react-router-dom** => para routing en el browser
 - **react-router-native** => para routing en react-native
 - Y algunas otras...

Configurar nuestra app con el Router

Importar

Una vez instalado, importaremos el módulo desde
react-router-dom

```
JS App.js > ...  
import React from 'react';  
import { BrowserRouter, Switch, Route } from 'react-router-dom';
```


Agregar la funcionalidad a toda la aplicación

Una vez realizado el import necesitamos configurar dos cosas:

1. Wrappear (envolver) la aplicación en un **BrowserRouter**
2. Crear un **Switch** (donde proyectaremos las vistas navegadas)
3. Crear los **Route's** de las distintas navegaciones con sus componentes asociados

```
function App() {  
  return (  
    <BrowserRouter>  
      <Navbar categories={categories} />  
      <Switch>  
        <Route exact path="/">  
          <Home />  
        </Route>  
        <Route path="/cart">  
          <Cart />  
        </Route>  
      </Switch>  
      <Footer />  
    </BrowserRouter>  
  );  
}  
  
export default App;
```



Especificidad de match

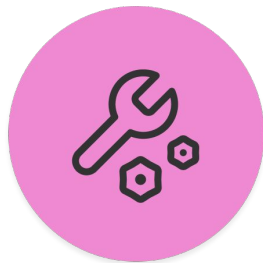
- Por defecto se matchean (coinciden) **únicamente partes** de la url, por lo tanto '/' va a matchear '/cart' o '/checkout', a no ser que le digamos que use la propiedad **exact**

```
<Navbar categories={categories} />  
<Switch>  
  <Route exact path="/">  
    <Home />  
  </Route>  
  <Route path="/cart">
```

VAMOS AL CÓDIGO



CODER HOUSE



Agrega un router a tu App

En tu aplicación, instala **react-router-dom**, agrégala al root de tu app y configura tus rutas apuntando a tu Home.

(Si tenes otro nombre o la organizaste distinta no hay problema)

Tiempo: 15 minutos

¿PREGUNTAS?



BREAK



CODER HOUSE

Navegar a una ruta

Navegar a una ruta

Ahora que tenemos todo configurado podemos importar un **link** perteneciente a react-router-dom en **cualquier componente** del sub-árbol del **<BrowserRouter>**

y usarlo para que al clicar, el **BrowserRouter** renderice ese **Route** que habíamos declarado dentro del **Switch**

```
src > components > Navbar > JS Navbar.js > ...  
1   import React from 'react';  
2   import { Link } from 'react-router-dom';
```

```
<li><Link to={`/${cart}`}>Carrito</Link></li>
```


Navegar a una ruta (con parámetros)

Navegar a una ruta - Con parámetros

Si hacemos la ruta
dinámica
(con parámetros)

```
<Route path="/categories/:categoryId">  
  <Home />  
</Route>
```

Podremos navegarla idénticamente pero de manera
dinámica

Navegar a una ruta - NavLink

¿Notaron que cambiamos el **Link** por un **NavLink**?

```
t => <li><NavLink to={` /categories/${cat.id}`} activeClassName="currentCategory" className="text-white">{cat.name}</NavLink>
```

Un NavLink es un **link con un estilo**, está siempre detectando la ruta actual, y si coincide con la suya nos activa la clase que le demos para que el user sepa **qué item de la lista** corresponde con la **vista actual**

***Recibir parámetros por
ruta***

Recibir parámetros - por ruta

React router provee integración con Hooks.

useParams:

- Lo podemos utilizar para leer en js los parámetros de la ruta
- En combinación con un `useEffect` nos sirve para obtener actualizaciones sobre los parámetros

```
function ComponentWithRouteParams() {  
  const { userId } = useParams();  
  useEffect(() => {  
    console.log('Received userId to: ', userId);  
    return () => {  
      console.log('Will change userId:', userId);  
    }  
  }, [userId]);  
}
```

VAMOS AL CÓDIGO



CODER HOUSE



Configurar routing

Configurar el routing para que el '/' nos lleve a Home.js, y se pueda ver el listado de productos. Posteriormente clickeando un producto se debe poder navegar a su detalle '/item/:id' y desde cualquiera de ambas navegar al carrito (Cart.js) clickeando en el CartIcon.js

Formato de entrega: GIF mostrando la navegabilidad

Tiempo total: 30 minutos

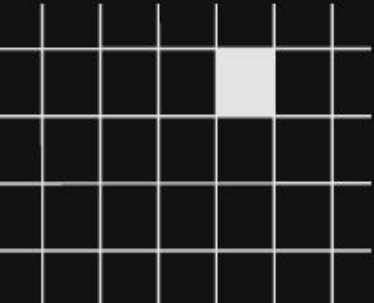
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Organización de navegabilidad
 - Routing estático
 - react-router-dom
- 



OPINÁ Y VALORÁ ESTA CLASE