

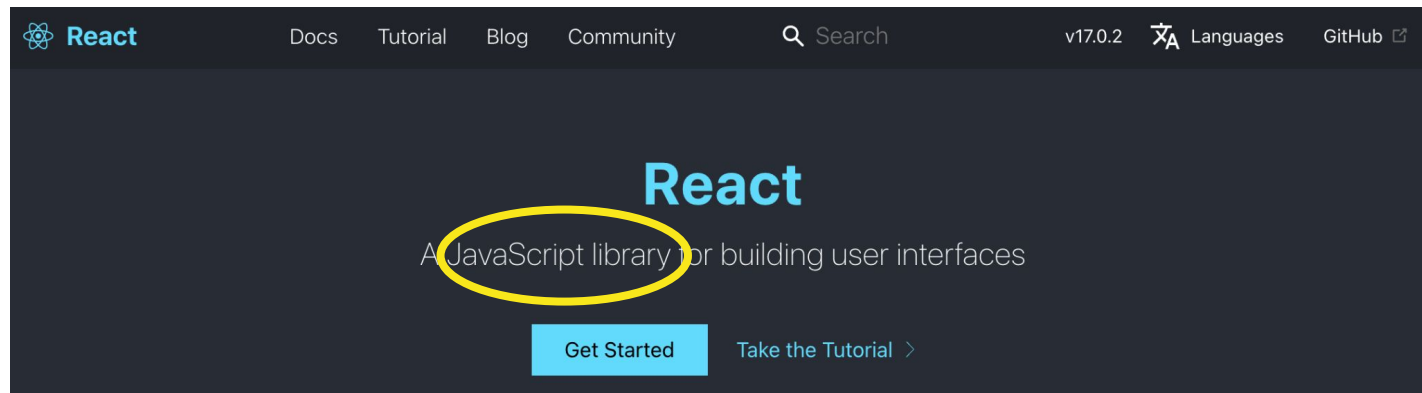
React Introduction

A brief introduction

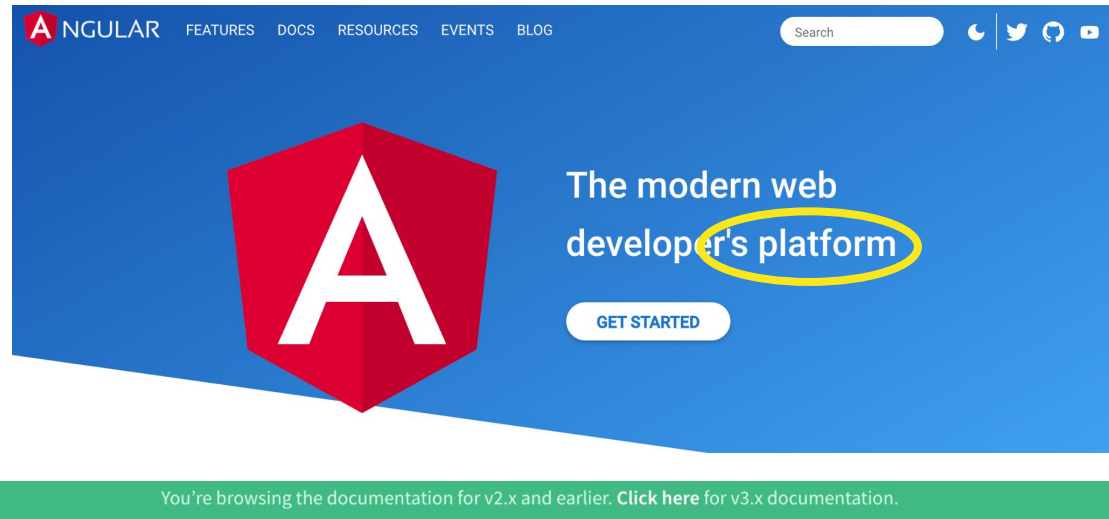
SOC

Servei d'Ocupació
de Catalunya

library



platform

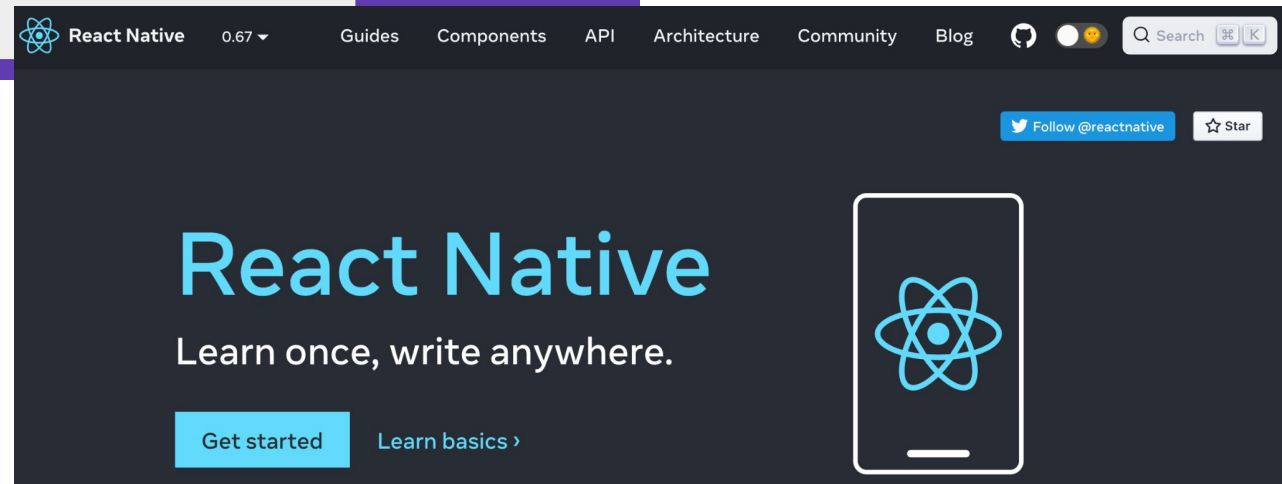


framework



The three competitors

React “Compatiblish”



Downloads Angular

@angular/core 

13.2.1 • [Public](#) • Published 15 hours ago

 [Readme](#)

 [Explore](#) 

 1 Dependency



Tip: Click on a version number to view a previous version's package page

Current Tags

Version	Downloads (Last 7 Days)	Tag
8.2.14	214,669	v8-lts
11.2.14	169,820	v11-lts
9.1.13	124,459	v9-lts
12.2.16	82,998	v12-lts
10.2.5	70,506	v10-lts
6.1.10	44,323	v6-lts
5.2.11	44,225	v5-lts
7.2.15	31,411	v7-lts
4.4.7	27,432	v4-lts
13.2.1	5,925	latest
14.0.0-next.1	491	next

Downloads Vue

vue TS

2.6.14 • Public • Published 8 months ago

 [Readme](#)

 [Explore](#) BETA

 [0 Dependencies](#)



Tip: Click on a version number to view a previous version's package page

Current Tags

Version	Downloads (Last 7 Days)	Tag
2.6.14	1,252,011	latest
3.2.29	171,394	next

Downloads React

react 

17.0.2 • Public • Published 10 months ago

 [Readme](#)

 [Explore](#) BETA

 [2 Dependencies](#)

Tip: Click on a version number to view a previous version's package page

Current Tags

Version	Downloads (Last 7 Days)	Tag
17.0.2	7,540,835	latest
18.0.0-rc.0	15,696	rc
18.0.0-beta-24dd07bd2-20211208	2,982	beta
18.0.0-rc.0-next-3a4462129-20220201	166	next

Angular Hello World

```
<!--app.component.html-->
```

```
<h1>Hello World!</h1>
```

```
// app.component.ts
```

```
import { Component } from "@angular/core";
```

```
@Component({
```

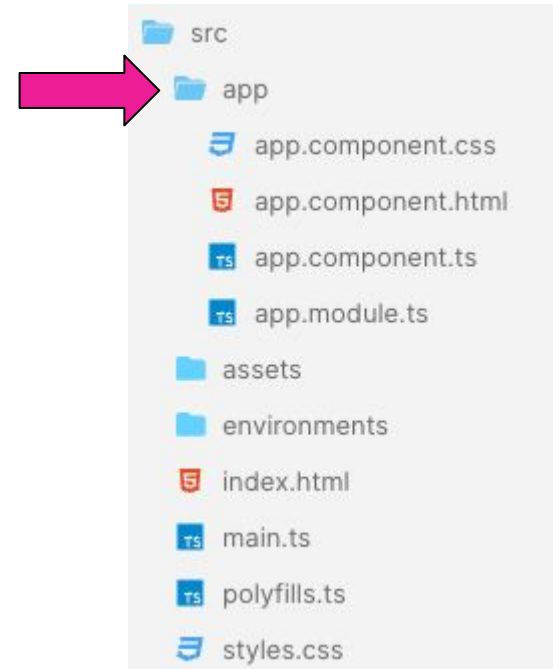
```
  selector: "app-root",
```

```
  templateUrl: "../app.component.html",
```

```
  styleUrls: ["../app.component.css"]
```

```
})
```

```
export class AppComponent {}
```

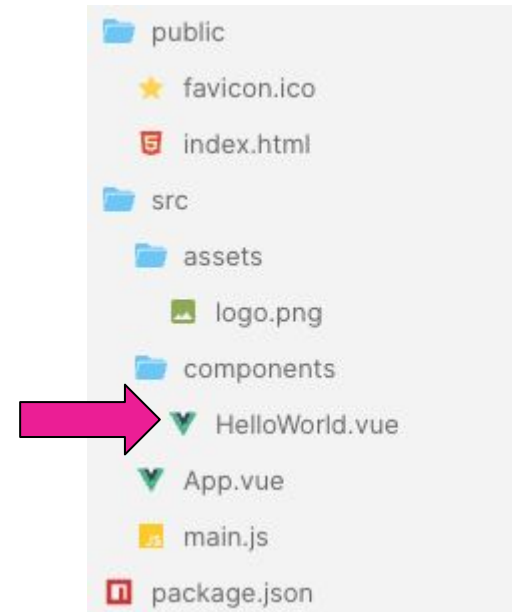


Vue Hello World

```
<template>  
  <h1>Hello World</h1>  
</template>
```

```
<script>  
  
export default {  
  name: "HelloWorld",  
};  
</script>
```

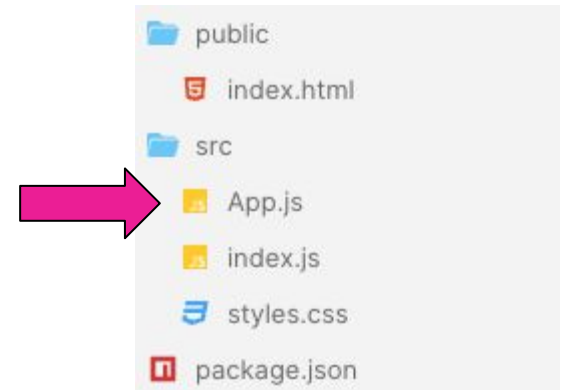
```
<style scoped>h1 { margin: 40px 0 0; }</style>
```



React Hello World

```
import './styles.css';
```

```
export default function App() {  
  return <h1>Hello CodeSandbox</h1>;  
}
```



React is part of an Ecosystem

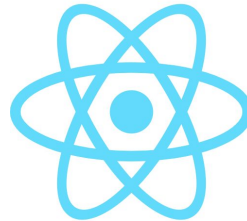


Lerna

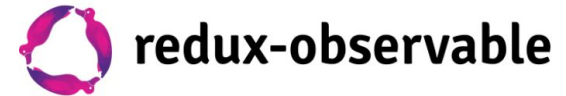
husky



BABEL



Create React App



Reselect

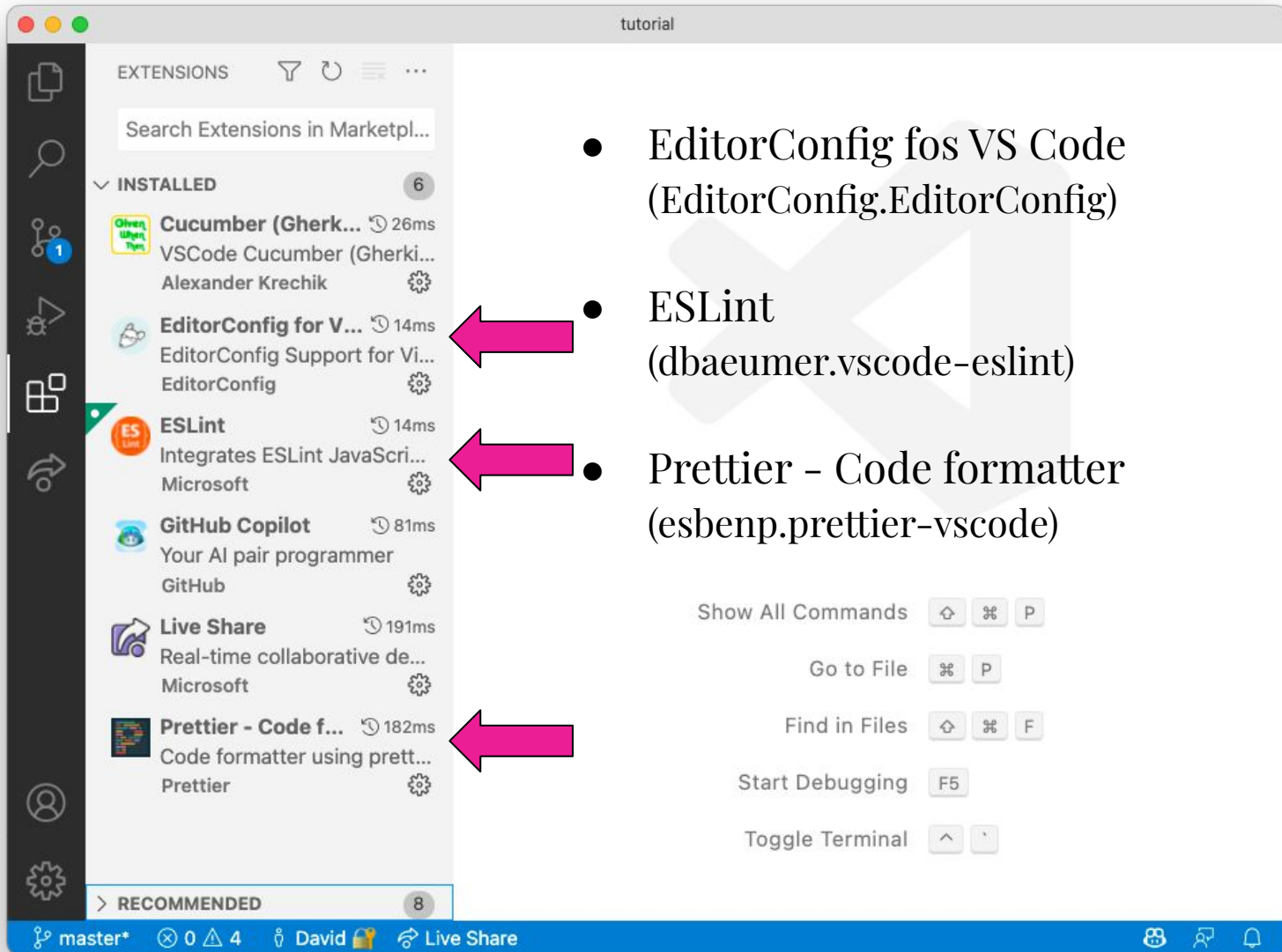


Testing Library



Setup VS Code

VS Code Setup: Add three Extensions



The screenshot shows the VS Code interface with the Extensions view open. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Extensions, and Settings. A pink arrow points to the Extensions icon. The Extensions view displays a list of installed and recommended extensions. Three extensions are highlighted with pink arrows pointing to them:

- EditorConfig for VS Code (EditorConfig.EditorConfig)
- ESLint (dbaeumer.vscode-eslint)
- Prettier - Code formatter (esbenp.prettier-vscode)

The bottom status bar shows the current file is named 'tutorial', the workspace is 'master*', and the user is 'David'.

VS Code Setup: Configure Prettier

settings.json — tutorial

settings.json

Users > drodenas > Library > Application Support > Code > User > settings.json

```
1 {  
2   "editor.defaultFormatter": "esbenp.prettier-vscode",  
3   "editor.formatOnSave": true,  
4 }  
5
```

"editor.defaultFormatter":

"esbenp.prettier-vscode",

"editor.formatOnSave": true,

Command Palette...

Settings [⌘,]

Online Services Settings

Extensions

Keyboard Shortcuts [⌘K ⌘S]

Migrate Keyboard Shortcuts from...

User Snippets

Live Share

UTF-8

LF

{ } JSON with Comments

⌘

Prettier

⌘

Your First App

Create-React-App



Create React App

Docs

Help 

GitHub 



 Search  



Create React App

Set up a modern web app by running one command.

[Get Started](#)

<https://create-react-app.dev/docs/getting-started>

Crear la App

```
$ npx create-react-app hello-world
```

```
$ cd hello-world
```

```
$ yarn test
```

```
# prem (a) all
```

```
# prem (q) quit
```

```
$ yarn start
```

```
PASS src/App.test.js  
✓ renders learn react link (18 ms)
```

```
Test Suites: 1 passed, 1 total  
Tests: 1 passed, 1 total  
Snapshots: 0 total  
Time: 0.669 s, estimated 1 s  
Ran all test suites.
```

```
Watch Usage: Press w to show more.
```

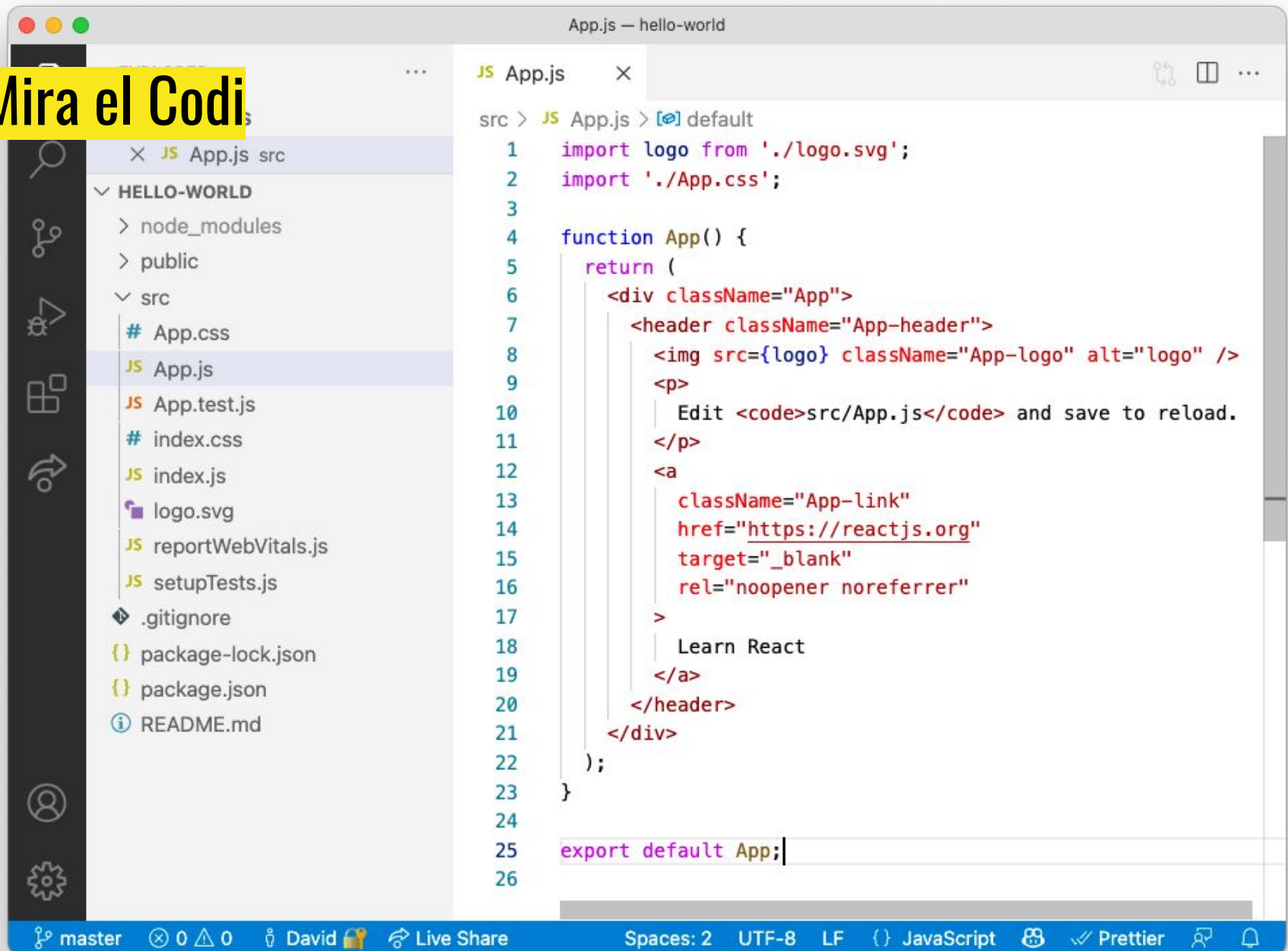
<http://localhost:3000>



Edit `src/App.js` and save to reload.

[Learn React](#)

Mira el Codi



Import i Export

Import: Agafa coses d'un altre fitxer

Export: Senyala el que un altre agafarà

```
// App.js  
import logo from './logo.svg';  
import './App.css';
```

```
function App() {  
  // ...  
}  
  
export default App;
```

```
// index.js  
  
import App from './App';
```

Hello World

```
export default function App() {  
  return <h1>Hello World</h1>;  
}
```

Hello World: Large

```
export default function App() {  
  return (  
    <div>  
      <h1>Hello World</h1>  
      <p>  
        This is my first <br />  
        page  
      </p>  
    </div>  
  ) ;  
}
```

Hello World: CSS

```
import './App.css';

export default function App() {
  return <div className="App">
    <h1>Hello World</h1>
    <p>
      This is my first <br />
      page
    </p>
  </div>;
}
```

Components

```
function HelloWorld() {  
  return <h1>Hello World</h1>;  
}
```

```
export default function App() {  
  return <div>  
    <HelloWorld />  
    <p>  
      This is my first <br />  
      page  
    </p>  
  </div>;  
}
```

Components i Fitxers

```
// HelloWorld.js
export default function HelloWorld() {
  return <h1>Hello World</h1>;
}
```

```
// App.js
import HelloWorld from "../HelloWorld.js"
export default function App() {
  return <div>
    <HelloWorld />
    <p>...
```

Props

```
// Hello.js
```

```
export default function Hello({ nom }) {  
  return <h1>Hello {nom}</h1>;  
}
```

```
// App.js
```

```
import Hello from "../Hello.js"  
export default function App() {  
  let john = "John";  
  return <div>  
    <Hello nom="Pere" />  
    <Hello nom={john} />  
    <Hello nom={<em>Paul</em>} />  
    <p>...  
  </div>  
}
```


Children

```
// Box.js
import './Box.css';
export default function Box({ children }) {
  return <div className="box">{children}</div>;
}

// App.js
import Box from './Box.js'
export default function App() {
  return <Box><h1>Hello</h1></Box>;
}
```

Javascript return i ;

Prova les següents 4 funcions en node i mira que retornen:

```
function mateixaLinia() {  
    return 1 + 2  
}  
  
function liniaPartida() {  
    return 1  
    + 2  
}  
  
function diferentLinia() {  
    return  
        1 + 2  
}  
  
function parentesis() {  
    return (  
        1 + 2  
    )  
}
```

JSX son objectes

> En el navegador, mira la web i la consola, que surt a cada lloc?

```
export default function HelloWorld() {  
  let helloWorld = <h1>Hello World</h1>;  
  console.log(helloWorld);  
  return helloWorld;  
}
```

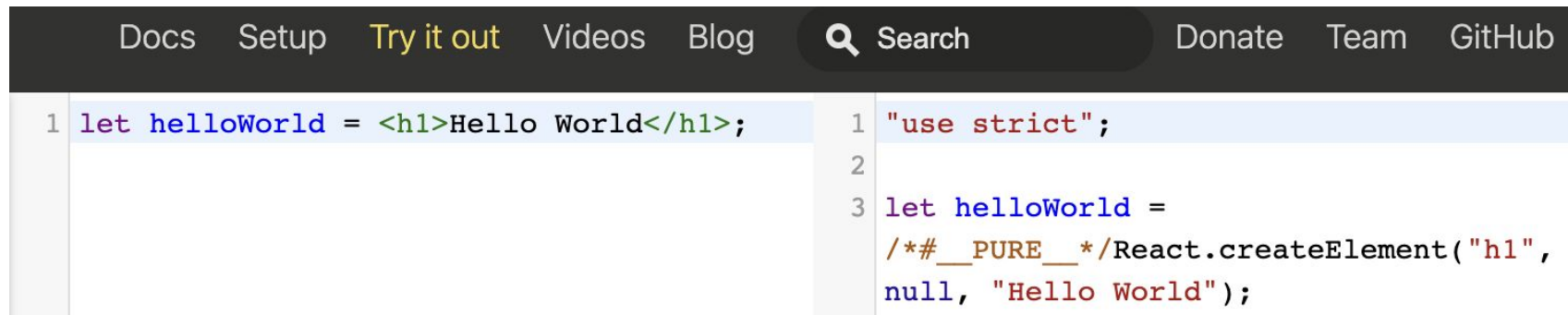
JSX no és Javascript

> Funciona això a la consola del browser?

```
let helloWorld = <h1>Hello World</h1>;
```

> Funciona a node?

Que fa babel quan ho enganxes a: <https://babeljs.io/repl>



JSX només pot retornar un element

> Funciona això?

```
export default function HelloWorld() {  
  return <h1>Hello</h1> <h2>World</h2>;  
}
```

És equivalent a haver fet això:

```
export default function HelloWorld() {  
  let hello = <h1>Hello</h1>;  
  let world = <h2>World</h2>;  
  return hello world;  
}
```

Fragments

Usar un fragment per retornar més d'una cosa:

```
export default function HelloWorld() {  
  return (  
    <>  
    <h1>Hello</h1>  
    <h2>World</h2>  
    </>  
  ) ;  
}
```

> Perquè el Prettier posa els parèntesis?

```
export default function Comment({author, text, date}) {  
  return (  
    <div className="Comment">  
      <div className="UserInfo">  
        <img className="Avatar"  
          src={author.avatarUrl}  
          alt={author.name}  
        />  
        <div className="UserInfo-name">  
          {author.name}  
        </div>  
      </div>  
      <div className="Comment-text">  
        {text}  
      </div>  
      <div className="Comment-date">  
        {formatDate(date)}  
      </div>  
    </div>  
  );  
}
```

Conditional Render: If & return

```
// Result.js
export default function Result({ isVictory }) {
  if (isVictory) return <h1>Victory!</h1>;

  return <del>Defeat!</del>;
}
```


Conditional Render: Hide itself

```
// Secret.js
export default function Secret({ isTrustworthy }) {
  if (!isTrustworthy) return null;

  return <div>The secret is #42</div>;
}
```

Conditional Render: Hide/Show other

```
// <Garage cars={['C3', 'Golf']} />
export default function Garage({ cars }) {
  return (
    <>
      <h1>Garage</h1>
      <p>This is the garage.</p>
      {cars.length > 0 && (
        <p>You have {cars.length} cars.</p>
      ) }
    </>
  );
}
```

Conditional Render: Ternary Operator

```
// <Garage cars={['C3', 'Golf']} />
export default function Garage({ cars }) {
  return (
    <>
      <h1>Garage</h1>
      {cars.length > 0 ? (
        <p>You have {cars.length} cars.</p>
      ) : (
        <p>The garage is empty.</p>
      )}
    </>
  );
}
```

List

```
// NumberList.js
export default function NumberList({ numbers }) {
  return <ul>
    {numbers.map((n) => (
      <li key={n}>{n}</li>
    ))}
  </ul>;
}
```

```
// App.js
export default function App() {
  return <Numbers numbers={[1, 2, 3]} />;
}
```

Events i Estats.

Events

```
// AlertButton.js
export default function AlertButton() {
  const avisam = () => alert('Alert Button Clicked!')
  return (
    <button onClick={avisam}>
      Alert Button
    </button>
  ) ;
}
```

Events

```
// AlertButton.js
export default function AlertButton() {
  return (
    <button onClick={
      () => alert('Alert Button Clicked!')
    }>
      Alert Button
    </button>
  );
}
```

State

```
import { useState } from "react";

// Counter.js
export default function Counter() {
  const [count, setCount] = useState(0);
  const increment = () => setCount(count + 1);
  return <div>
    Comptador: ${count}.<br />
    <button onClick={increment}>Increment</button>;
  </div>;
}
```


State: amb funció setter

```
import { useState } from "react";

// Counter.js
export default function Counter() {
  const [count, setCount] = useState(0);
  const increment = () => setCount((n) => n + 1);
  return <div>
    Comptador: ${count}.<br />
    <button onClick={increment}>Increment</button>;
  </div>;
}
```

Tria la Aventura

Implementa tria la aventura
amb React.

```
export default function Pagina() {  
  let [paginaActual, setPaginaActual] = useState(0);  
  let contingut = historia[paginaActual];  
  let si = () => setPaginaActual(contingut.si);  
  let no = () => setPaginaActual(contingut.no);  
  let text = contingut.text;  
  // ...  
}
```


Plantilla per el component principal

Efectes.

Efectes

```
import { useState, useEffect } from "react";

export default function Ping() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    setTimeout(() => {
      setCount((n) => n + 1);
    }, 2000);
  }, []);  // Que succeix si s'esbora el , [] ?

  return <h1>I have counted {count} times!</h1>;
}
```

Efectes: exemple Relotge

```
import { useState, useEffect } from "react";

export default function Timer() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    setInterval(() => {
      setCount((n) => n + 1);
    }, 1000);
  }, []);

  return <h1>It has passed {count} seconds.</h1>;
}
```

Neva per Nadal

Implementa neva per Nadal amb
React.

```
// Ara els flocs NO estan a l'html
// sinó en dades, com la història
function crearFlocs() {
    return [
        nouFloc(),
        nouFloc(),
        // ...
    ];
}


function nouFloc() {
    return { x: -10, y: -10 };
} // Prova primer amb x: 30, i y: 20
```

Dades


```
import { useState, useEffect } from "react";

export default function Nevada() {
  let [flocs, setFlocs] = useState(crearFlocs());
  // ...
}
```

Els flocs es guarden amb useState

```
export default function Floc({ x, y }) {  
  return (  
    <span  
      style={{ top: `${y}%`, left: `${x}%` }}  
      className="floc"  
    >  
        
    </span>  
  ) ;  
}
```

Plantilla per a un floc de neu

```
setFlocs((tots) => moureFlocs(tots));

function moureFlocs(flocs) {
  return flocs.map(floc => moureFloc(floc));
}

function moureFloc({x, y}) {
  return { x: x + 1, y: y + 1 };
}
```

Plantilla per moure flocs

Input

Llegir Opció 1: On Event (Uncontrolled)

```
import { useRef } from "react";

export default function SayHello() {
  const inputRef = useRef();
  const onClick = () => {
    alert(`Hola ${inputRef.current.value}`);
  };
  // https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Input#attr-value

  return (
    <>
      Nom: <input ref={inputRef} />
      <br />
      <button onClick={onClick}>Say Hello</button>
    </>
  );
}
```

Llegir Opció 2: Controlled Input

```
import { useState } from "react";

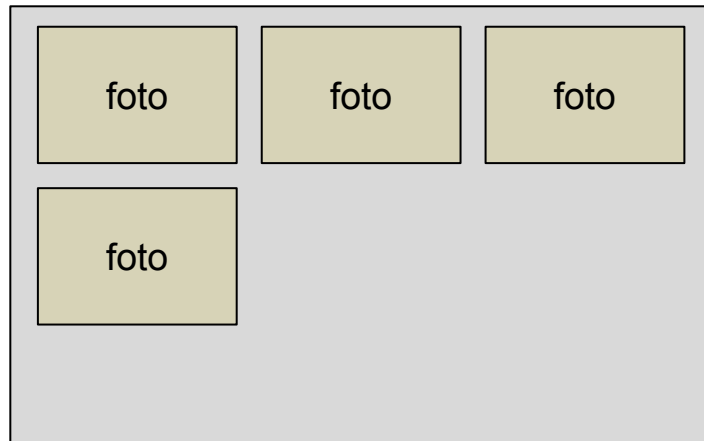
export default function SayHello() {
  const [nom, setNom] = useState("");

  return (
    <>
      Hola {nom} .<br />
      Name: { " " }
      <input
        value={nom}
        onChange={ (event) => setNom(event.target.value) }
      />
    </>
  );
}
```

Exercisis

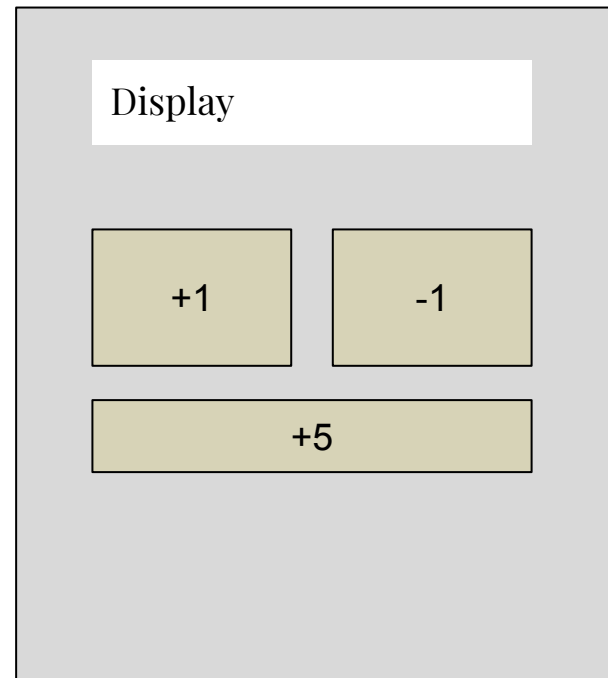
Graella de Fotos

1. Donat un array amb URL de fotos, mostrar les fotos en graella
2. En clicar un foto, mostrar-la en pantalla completa (avançat)
3. En clicar la foto ampliada, mostrar la graella un altre cop



Comptador

1. Afegir display amb comptador
2. Afegir botó d'incrementar 1
3. Afegir botó de decrementar 1
4. Limitar a zero el mínim
5. Limitar a deu el màxim
6. Afegir botó d'incrementar 5



Acordió

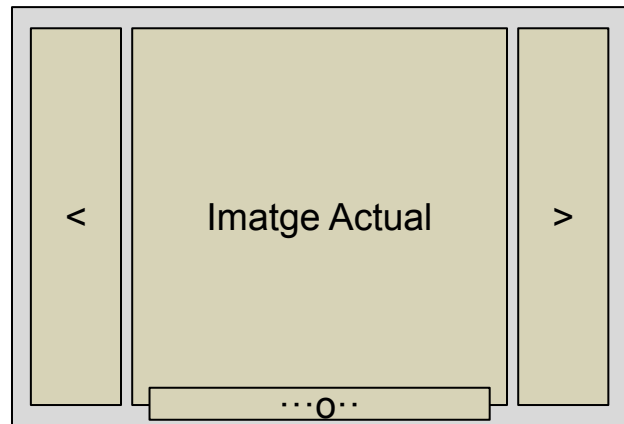
1. Mostrar un títol
2. En clicar el títol, mostra i amaga un text a sota
3. Mostrar un segon títol
4. En clicar el segon títol, mostra i amaga un altre text a sota
5. Usar un array per diversos títols/text
6. En clicar un títol, plega qualsevol altre text



Títol 1
Títol 2
Títol 3
Contingut 3
Títol 4
Títol 5

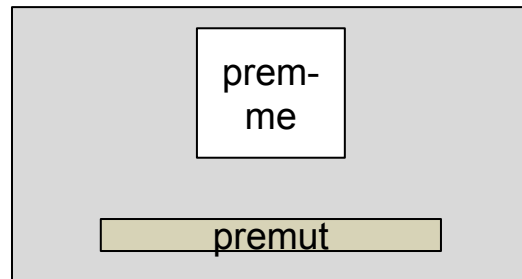
Carrusel

1. Mostrar una imatge
2. Afegir botó per canviar a una altra imatge
3. Afegir botó per tornar a l'anterior
4. Afegir més imatges darrera (només una es mostra)
5. Afegir una barra amb un punt per cada imatge
6. Ressaltar el punt de la imatge actual
7. Anar a la imatge del punt en fer clic



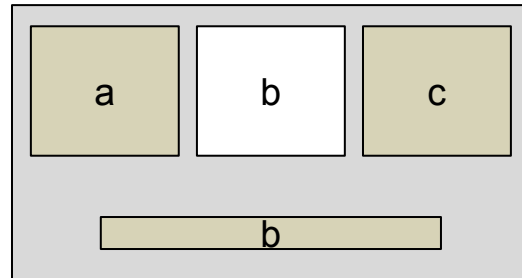
Toggle Button *17F

1. Mostrar 1 botó
2. En prémer el botó, canviar el color del botó
3. En prémer de nou el botó, treure el color del botó
4. Següents clics repliquen 2,3,2,3,2,3...
5. Mostrar un text a sota dient si està premut o no



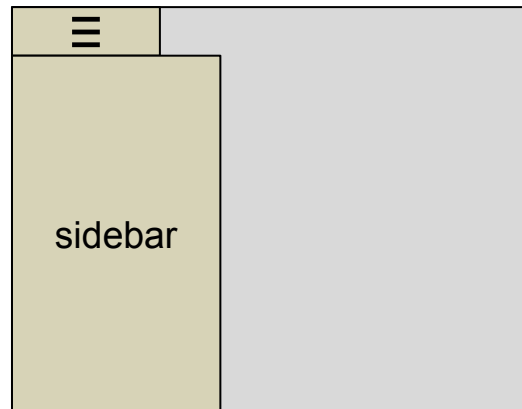
Group Button *17F

1. Mostrar 3 botons, cada un amb un text diferent
2. Mostrar un requadre a baix amb un text inicial “cap”
3. En prémer un botó, canviar el text del requadre
4. Si el text del requadre coincideix amb un botó, aquell té un color diferent



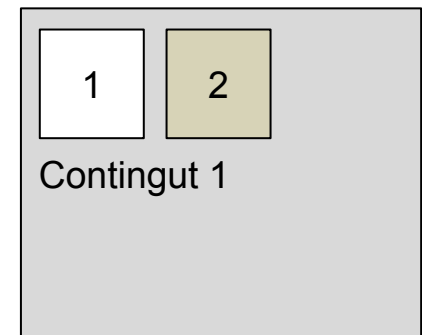
Toggleable sidebar, 17F

1. Mostrar un botón \equiv
2. Mostrar una sidebar (position: absolute, top:0, bottom:0)
3. Quan es clica el botó, ocultar la sidebar (display:none)
4. Quan es torna a clicar el botó, mostrar la sidebar



Tabs, 17F

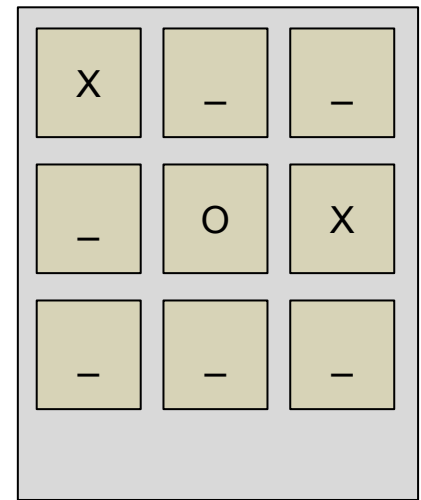
1. Mostrar 2 botons
2. Si el primer botó està pres mostrar un contingut que diu 1
3. Si el segon botó està pres, mostrar un contingut que diu 2
4. Per defecte, el primer botó està pres
5. Marcar el botó pres amb un color diferent
6. Donar aspecte de tabs als botons
7. (Avançat) Fer un component que rep un array de: [{títol, Contingut}], on títol és l'string del nom del botó a mostrar, i Contingut és el component a mostrar.



Tres en ratlla

1. Mostrar els botons en 3x3
2. En prémer qualsevol botó aquest canvia a X
3. El segon botó premut canvia a O
4. S'intercala la X i la O quan es prem
5. Detectar el fi de partida, mostrar el guanyador
6. Al final de partida, donar opció a reiniciar la partida
7. (Avançat) Usar array per l'estat dels botons
8. (Avançat) Afegir IA d'O que mai perdi

```
let array = ['_', '_', '_', /* ... */];  
let copia = [...array];  
copia[1] = 'X';  
setArray(copia);
```



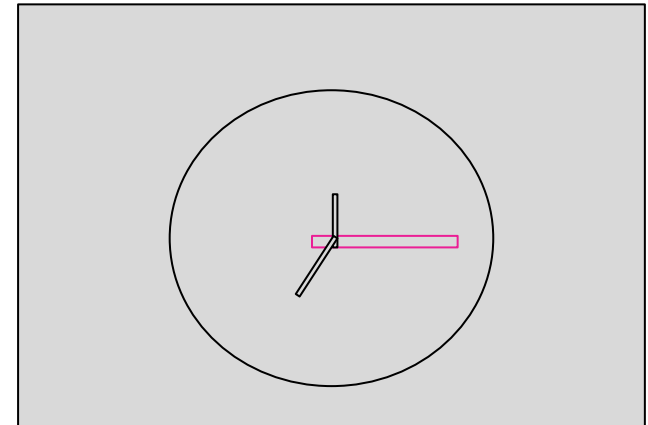
Relloj Digital

1. Mostrar l'hora actual (amb segons)
2. Actualitzar l'hora cada segon
3. Mostrar la data actual
4. Afegir un botó per ocultar la data actual
5. Afegir un botó per mostrar la data actual



Relotge Analògic (avançat)

1. Mostrar l'hora actual (amb segons) en analògic
2. Actualitzar l'hora cada segon
3. Mostrar l'esfera rodona (border-radius: 100%)
4. Afegir l'agulla de segons (div position absolute estret)
5. Moure l'agulla de segons cada segon (translate + rotate)
6. Afegir i moure l'agulla de minuts
7. Afegir i moure l'agulla d'hora
8. Afegir números 12, 3, 6, 9



Sumador de números

1. Mostrar un input per a un valor
2. Mostrar un input amb un total (inicial zero)
3. Afegir un botó que sumi el valor al total

Valor: []

Total: _____

(sumar)

Tip Calculator

1. Mostrar un input pel valor d'un dinar
2. Mostrar un input pel % de propina que es vol posar
3. Mostrar el total valor + % de propina
4. Afegir un checkbox per arrodonir a 5€ el total
(avançat: mirar a MDN checked no value)
5. Afegir un comptador de comensals
(avançat: limitar d'1 a 10, mirar a MDN min-max)
6. Mostrar el valor que haurà de pagar cada comensal

Preu: []
Propina: [] %
Total: _____
[] redondeig
Comensals: [] v^

Router

Adding the router

```
$ yarn add react-router-dom
```

App.js

```
import {BrowserRouter, Routes, Route} from "react-router-dom";
import Layout from "../pages/Layout";
import Home from "../pages/Home";
import Contact from "../pages/Contact";
import NoPage from "../pages/NoPage";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

// Sempre pinta això (sempre /)

// Aquest quan és a l'index

// Això és el que mostra

Ull, és dins

// Aquest a /contact

// Aquest quan no hi ha res a mostrar

./pages/Layout.js

```
import { Outlet, Link } from "react-router-dom";

export default function Layout() {
  return (
    <>
      <nav>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/contact">Contact</Link></li>
        </ul>
      </nav>
      <Outlet />
    </>
  );
};
```

// Reemplaça <a href>

// Pinta el que hi ha dins del Router (aquest cas el path Podria ser el <main>).

./pages/Home.js

```
export default function Home () {  
  return <h1>Home</h1>;  
}
```



// Es un component normal

Paràmetres (avançat)

// Definir la ruta amb paràmetre postId

```
<Route path="/posts/:postId" element={<Post />} />
```

// Enllaç u una ruta amb un post concret

```
<Link to={` /posts/${postId} `}></Link>
```

// Component que usa el paràmetre

```
import { useParams } from "react-router-dom";  
export default function Post() {  
  const params = useParams();  
  return <h2>Post: {params.postId}</h2>  
}
```

Exercicis

Objectes i Arrays

Exercici 1

Donat un vector d'enters,
comprovar el major, el menor i la
mitjana de tots.

Exercici 1

Donat un vector d'enters, comprovar el major, el menor, la suma de tots i la mitjana.

Ex:

63,45,58,56

Suma: 222

Mitjana: 55.5

Major: 63

Menor: 45

Exercici 2

Afegir un element.

Donat un array d'enters, que puguis afegir un nou enter tot indicant a quina posició el vols afegir.

Exercici 3

Treure els elements repetits d'un array.

Exercici 4

Treure els elements NO repetits d'un array.

Exercici 5

Concatenar dos arrays de la seguent forma:
 $A_0, B_0, A_1, B_1, A_2, B_2, \dots, A_n, B_n$

Exercici 6

Concatenar dos arrays de la seguent forma:
 $A_0, B_0, A_1, B_1, A_2, B_2, \dots, A_n, B_n$

Però amb dos arrays de diferent tamany.

Exercici 7

Donat un llistat de ids, i un objecte classificat per id, treure un array de objectes pels ids

Exemple:

Donat l'array: ['u1','u3']

I l'objecte:

{u1:{name:'Pere'},u2:{name:'Joan'},u3:{name:'Maria'}}

Resultat: [{name:'Pere'},{name:'Maria'}]

useReducer

useReducer

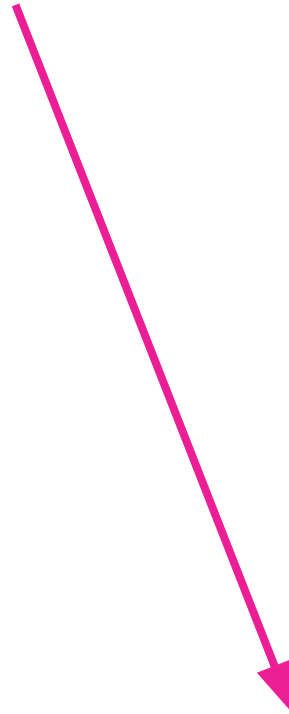
Forma part de React.

S'inspira en Redux.

És una millora de l'useState.

useReducer vs useState (llegir estat)

```
const [state, setState] = useState(valorInicial);
```



```
const [state, dispatch] = useReducer(reducer, valorInicial);
```

useReducer vs useState (canviar state)

```
const [state, setState] = useState(valorInicial);
```



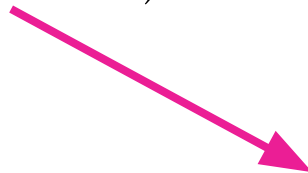
setState((stateActual) => stateSegüent)

The diagram illustrates the relationship between the `dispatch` function from `useReducer` and the `setState` function from `useState`. A horizontal magenta line is positioned below the `setState` call. Two magenta arrows originate from below this line: one points to the `stateActual` argument, and the other points to the `stateSegüent` result. The `setState` call itself is highlighted with a yellow background.

```
const [state, dispatch] = useReducer(reducer, valorInicial);
```

useReducer dispatch argument

dispatch(argument)



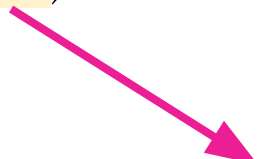
reducer = (stateAnterior, argumentDispatch) => stateSegüent

```
const [state, dispatch] = useReducer(reducer, valorInicial);
```


useReducer dispatch argument

dispatch(action)

reducer = (stateAnterior, action) => stateSegüent



```
const [state, dispatch] = useReducer(reducer, valorInicial);
```

Codi d'useReducer

```
function useReducer(reducer, initialState) {  
  const [state, setState] = useState(initialState);  
  const dispatch = (action) => setState(  
    state => reducer(state, action)  
  );  
  
  return [state, dispatch];  
}
```

Counter amb useState

```
import { useState } from "react";
export default function Counter() {
  const [count, setCount] = useState(0);

  return (
    <>
      {count}{" "}
      <button onClick={ () => setCount(count + 1) }>
        +1
      </button>
    </>
  );
}
```

Counter amb useState (bis)

```
import { useState } from "react";
export default function Counter() {
  const [count, setCount] = useState(0);

  return (<
    {count}{" "}
    <button
      onClick={() => setCount((state) => state + 1)}
    >
      +1
    </button>
  </>);
}
```

Counter amb useReducer

```
import { useReducer } from "react";
export default function Counter() {
  const [count, dispatch] = useReducer(
    (state) => state + 1
    , 0);

  return (
    <>
      {count}{" "}
      <button onClick={ () => dispatch() }>+1</button>
    </>
  );
}
```

Counter amb useReducer i decrement

```
import { useReducer } from "react";
export default function Counter() {
  const [count, dispatch] = useReducer(
    (state, amount) => state + amount
  );

  return (
    <>
      {count}{" "}
      <button onClick={() => dispatch(+1)}>+1</button>
      <button onClick={() => dispatch(-1)}>-1</button>
    </>
  );
}
```

Patrons

Els patrons són receptes.

Poden ser noves invencions, o formes convencions d'escriptura.

Ajuden a crear codi que s'adapti als canvis.

Ajuden a reconèixer el que ha escrit altres.

Són conductes de bon comportament.

Patró Action

Les accions són objectes amb un camp type, valor string, que pot contenir altres valors.

A més, els tipus d'acció s'han de desar en constants.

```
const INCREMENT = 'INCREMENT';
```

```
const action = {  
  type: INCREMENT,  
  altres: 'valors',  
}
```


Patró ActionCreator

Són funcions que retornen accions.

```
function increment(amount) {  
  return {  
    type: INCREMENT,  
    amount  
  }  
}
```

La bona conducta diu que cal cridar un actionCreator per crear una acció i no ho farem directament.

```
dispatch(increment(+1));  
dispatch({ type: INCREMENT, amount: +1 });
```

Patró Reducer

Són funcions que donat un state i una acció, retornen un nou state. Cal que usin switch en comptes d'if, i sempre cal posar default: return state;

// “Estat inicial”

```
function counter(state = 0, action) {  
  switch (action.type) {  
    case INCREMENT:  
      return state + action.amount;  
    case RESET:  
      return 0;  
    default:  
      return state;  
  }  
}
```

(incís sobre switch)

El switch equival al if, però sempre compara el mateix valor.

```
switch (action.type) {  
  case INCREMENT:  
    return state + action.amount;  
  case RESET:  
    return 0;  
  default:  
    return state;  
}
```

```
if (action.type === INCREMENT) return state + action.amount;  
else if (action.type === RESET) return 0;  
else return state;
```

Counter amb patrons

// Fitxer o carpeta amb la lògica

```
import { useReducer } from "react";
import { counter, increment, reset } from "../counter";
export default function Counter() {
  const [count, dispatch] = useReducer(counter, 0);
```

// Es el reducer

```
  return (<>
    {count}{" "}
    <button onClick={() => dispatch(increment(+1))}>
      +1
    </button>
    <button onClick={() => dispatch(increment(-1))}>
      -1
    </button>
    <button onClick={() => dispatch(reset())}>
      reset
    </button>
  </>);
```

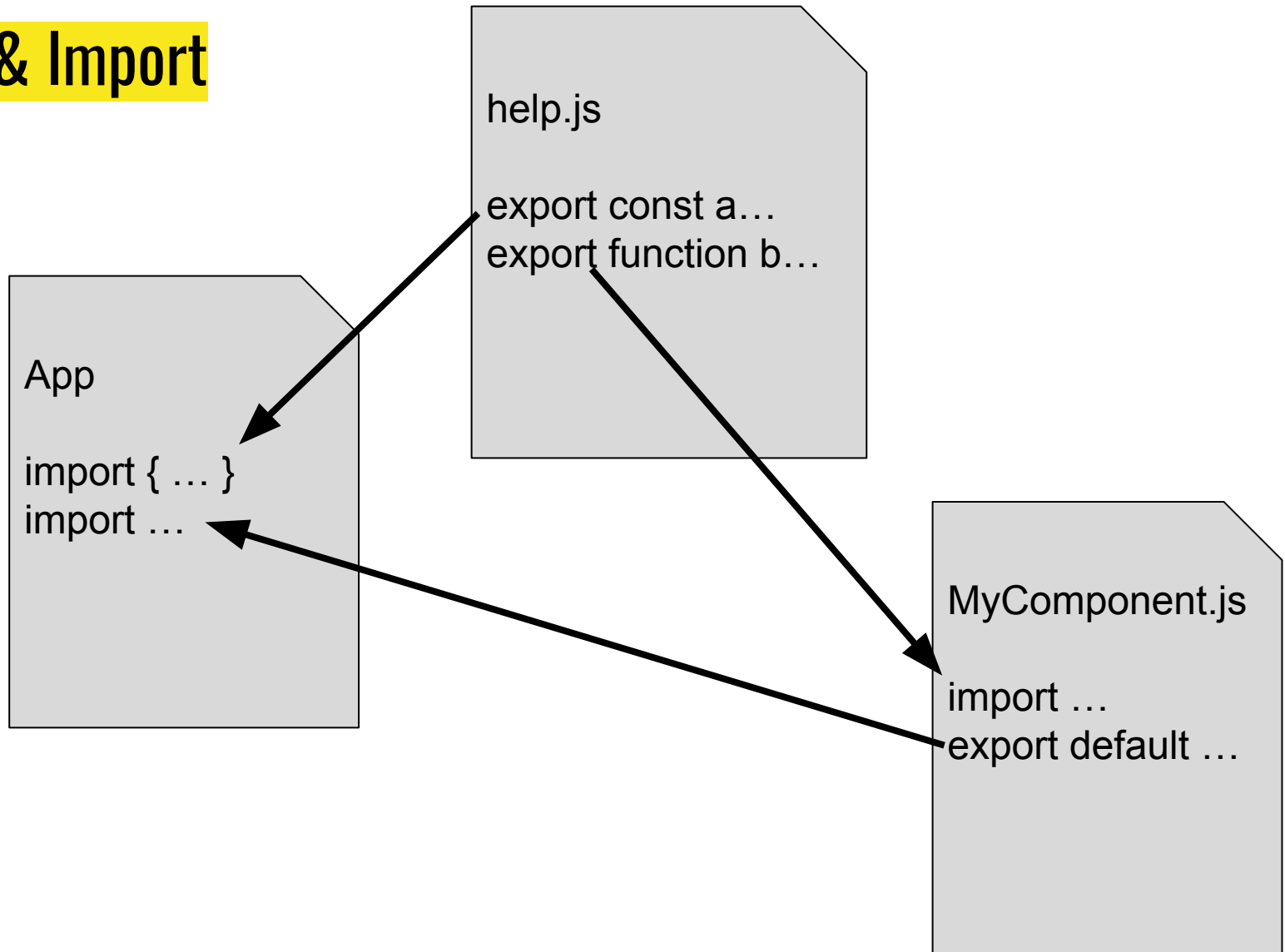
// Action creators

// Action creator

```
}
```

Imports, Exports & Barrel

Export & Import



Exports default

Es exports poden ser **default**:

```
export default function App() { /* ... */ }
```

```
const values = [ /* ... */ ];  
export default values;
```

De fet, el que fa és exportar-ho amb nom default.

Exports named

Es exports poden ser **named**:

```
export function MyComponent () { /* ... */ }
```

// No hi ha default



```
export const valors = [ /* ... */ ];
```

// No cal separar amb dues línies



Imports default

Els import dels defaults es fan sense claus.

```
import App from './App'
```

El nom del fitxer original és irrellevant, es pot canviar.

```
import NomDiferent from './App'
```

* Això vol dir que l'export/import default és perillós: es pot posar el nom malament i no adonar-se'n.

Imports named

Els import dels named es fan amb claus.

```
import { increment } from './counter';
```

Es pot importar més d'un named del mateix fitxer usant comes:

```
import { increment, reset } from './counter';
```

Imports re-named

Els import dels named es poden reanomenar si fa falta:

```
import {  
    increment as counterIncrement,  
    reset  
} from './counter';
```

Imports default is named default

El default de fet s'exporta amb nom default, però cal renombar-lo perquè default és paraula reservada.

```
import {  
    default as App,  
} from './App';
```

Cadena d'export

Es poden re-exportar coses d'un altre arxiu.

```
export { increment, reset } from './counter';
```

I es poden reexportar reanomenant.

```
export {  
    increment as counterIncrement,  
    reset  
} from './counter';  
export { default as App } from './App';
```

Fins i tot es pot exportar tot.

```
export * from './counter';
```

Import d'un directori

Si s'importa d'un directori, buscarà el index.js, i importarà tot el que exporti el index.js.

```
import { store } from './store';
```

```
// es el mateix que:
```

```
import { store } from './store/index';
```

```
// es el mateix que:
```

```
import { store } from './store/index.js';
```

Ho podem aprofitar per reemplaçar fitxers per directoris, i poder organitzar millor les coses dins dels directoris.

Barrel

És un directori amb un index.js que exporta el que hi ha dins i ens amaga com té els fitxers dins.

```
import { counters, CounterRedux } from './counters';
```

```
// counters/index.js  
export * from './counters';  
export * from './CounterRedux';  
export * from './ListCounters';
```

```
✓ counters  
  ✓ CounterRedux  
    JS CounterRedux.js  
    JS CounterReduxCount.js  
    JS CounterReduxIncrement.js  
    JS CounterReduxReset.js  
    JS index.js  
  JS counters.js  
  JS index.js  
  JS ListCounters.js
```

Redux

Redux a l'App

Redux es comporta com l'useReducer, però l'estat i el dispatch el comparteix amb tota l'aplicació i amb tots els components.

Creació de l'estat

Cal afegir redux a l'aplicació:

```
$ yarn add redux
```

L'estat es crea de la següent manera:

```
const store = createStore(reducer, optionalInitialState);
```

l l'Store té:

- `store.getState(): state`
- `store.dispatch(action)`

Exemple de Redux

```
import { createStore } from 'redux';  
import { counter, increment } from './counter';  
  
const store = createStore(counter);  
  
store.dispatch(increment(+1));  
store.dispatch(increment(+1));  
  
console.log(store.getState());
```

React + Redux

Cal la llibreria react-redux:

```
$ yarn add react-redux
```

Cal connectar react-redux amb react amb:

- `<Provider store={store}>...</Provider>`: Proveïx l'store a tots els components que té dins.
- `useSelector(selector)`: Fa un `getState` i llegeix amb una funció anomenada `selector` el contingut de l'estat.
- `useDispatch()`: Retorna la funció `dispatch`.

Exemple React/Redux: App

```
import { createStore } from "redux";
import { Provider } from "redux-redux";
import { counter, increment } from "./counter";
import CounterRedux from "./Counter";
const store = createStore(counter);

export default function App() {
  return (
    <Provider store={store}>
      <CounterRedux />
    </Provider>
  );
}
```

Exemple React/Redux: CounterRedux

```
import { useSelector, useDispatch } from "react-redux";
import { increment, reset } from "../counter";
export default function CounterRedux() {
  const count = useSelector((state) => state);
  const dispatch = useDispatch();

  return (
    <>
      {count}{" "}
      <button onClick={() => dispatch(increment(+1))}>+1</button>
      <button onClick={() => dispatch(increment(-1))}>-1</button>
      <button onClick={() => dispatch(reset())}>reset</button>
    </>
  );
}
```

Patró Selector

Són funcions que donat l'estat, retornen el valor que volem.

```
export default function getCount(state) {  
  return state;  
}
```

Això es fa perquè l'estat és de TOTA l'aplicació i pot créixer molt.

Exemple React/Redux: CounterRedux amb Patró Selector

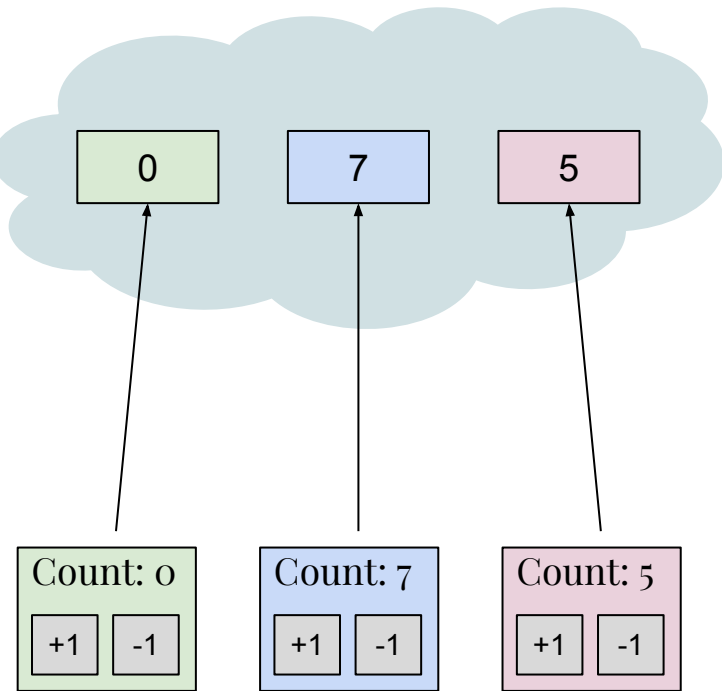
```
import { useSelector, useDispatch } from "react-redux";
import { getCount, increment, reset } from "../counter";
export default function CounterRedux() {
  const count = useSelector(getCount);
  const dispatch = useDispatch();

  return (
    <>
      {count}{" "}
      <button onClick={() => dispatch(increment(+1))}>+1</button>
      <button onClick={() => dispatch(increment(-1))}>-1</button>
      <button onClick={() => dispatch(reset())}>reset</button>
    </>
  );
}
```

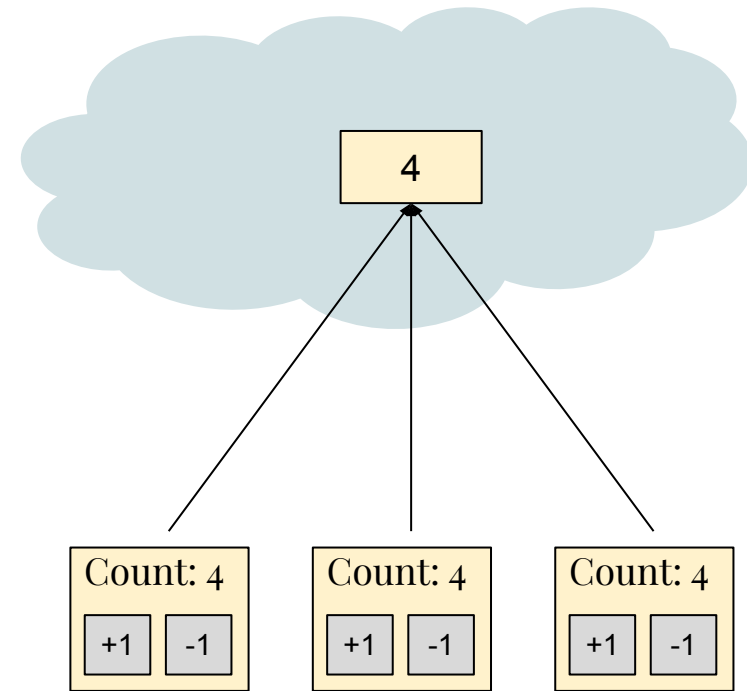

useReducer

vs

Redux

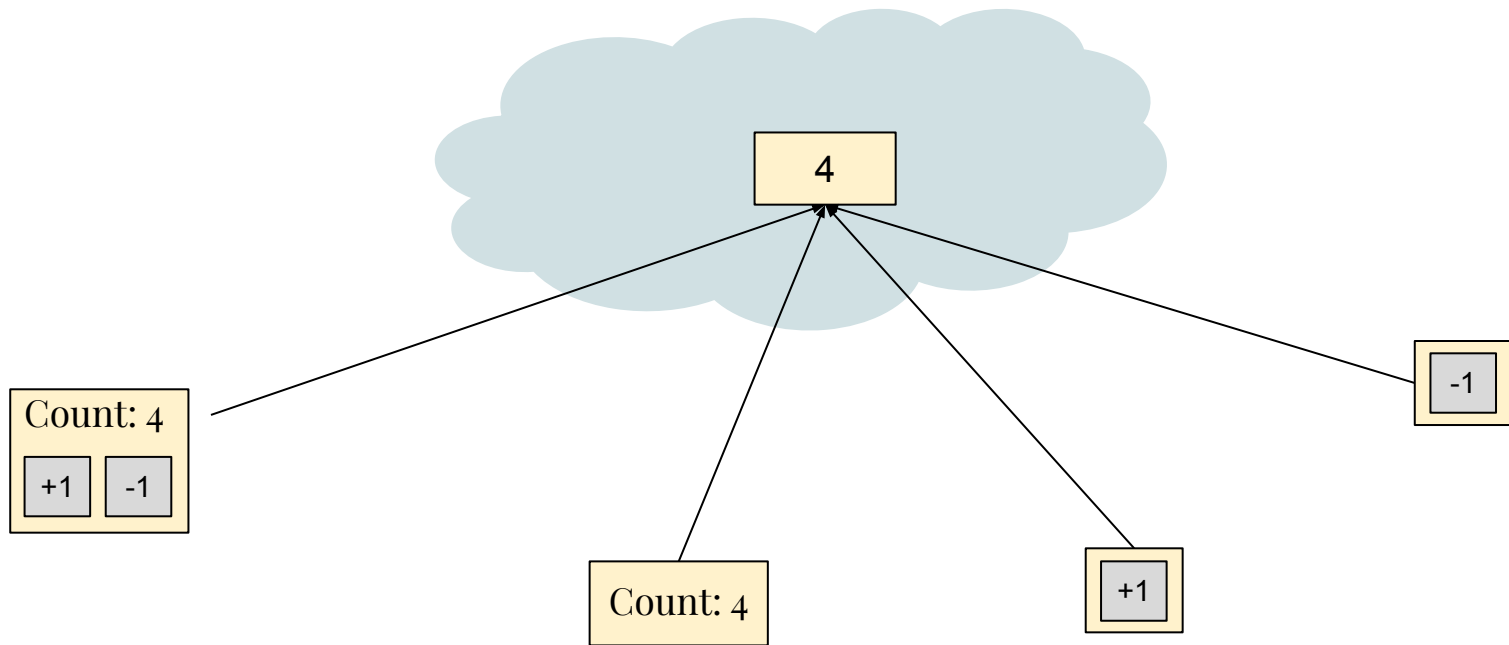


Hi ha un estat diferent per cada component.

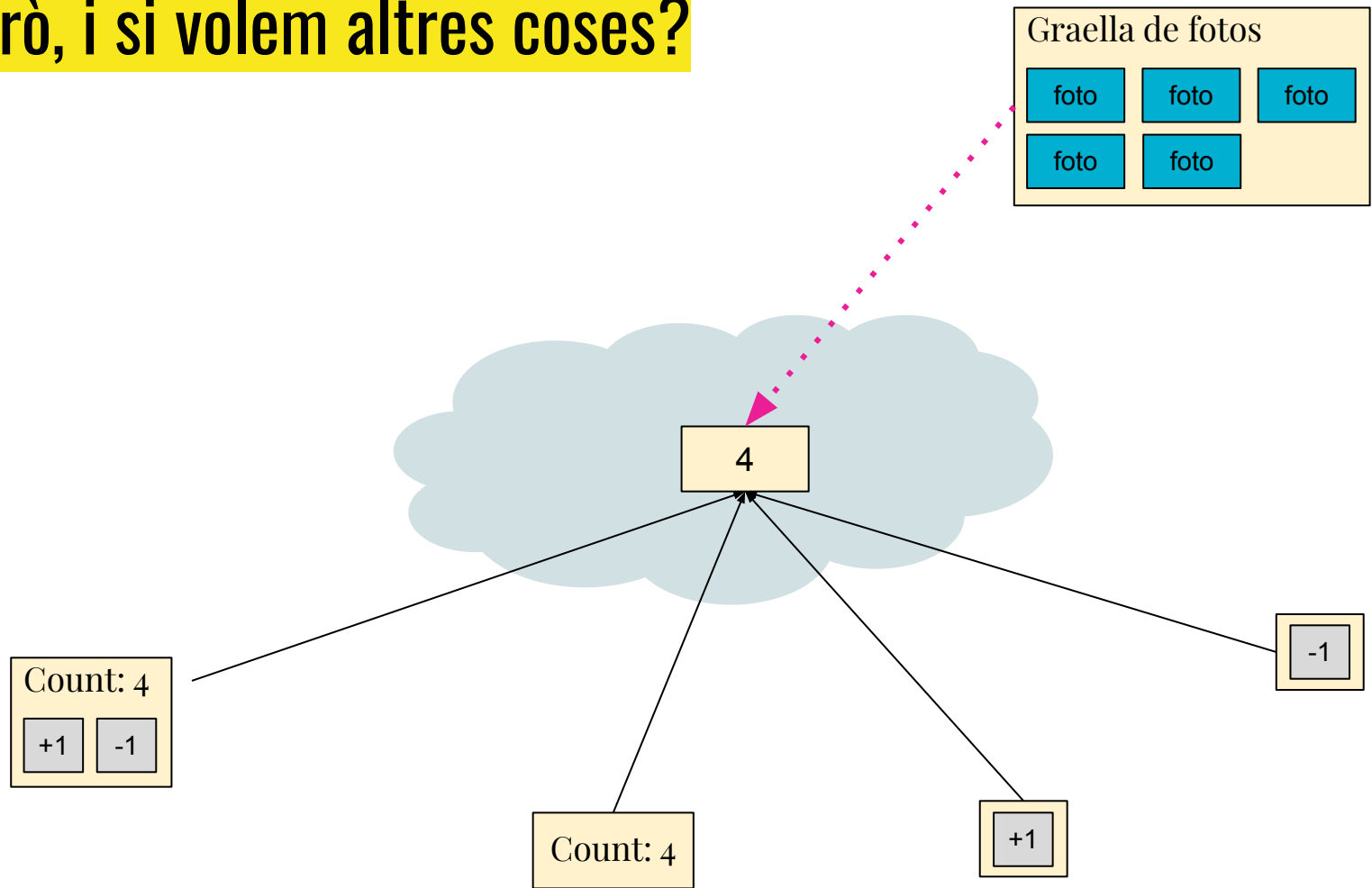


Tots els components comparteixen el mateix estat.

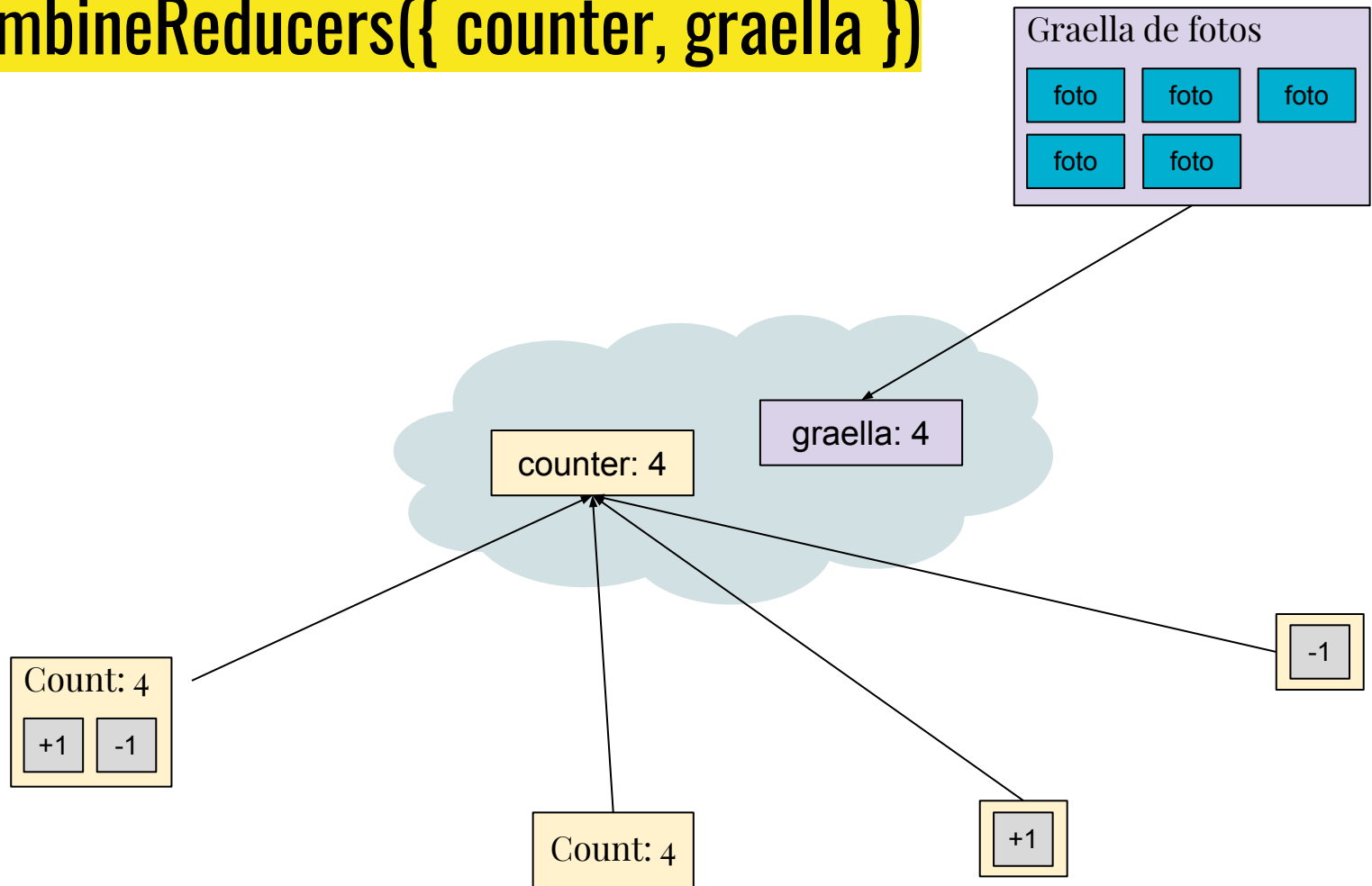
Redux permet separar components



Però, i si volem altres coses?



combineReducers({ counter, graella })



combineReducers

```
import { combineReducers, createStore } from "redux";
import { counter } from "../counter";
import { graella } from "../graella";

const reducer = combineReducers({
  counter,
  graella,
});

const store = createStore(reducer);
```

reducers

```
export function counter(state = 0, action) {  
  switch (action.type) {  
    case INCREMENT: return state + action.amount;  
    default: return state;  
  }  
}  
  
// -----  
export function graella(state = null, action) {  
  switch (action.type) {  
    case ZOOM: return action.index;  
    case UNZOOM: return null;  
    default: return state;  
  }  
}
```

Action types

// Els action type cal que incloguin el nom del reducer

```
// counter
```

```
const INCREMENT = "counter/INCREMENT";
```

```
// graella
```

```
const ZOOM = "graella/ZOOM";
```

```
const UNZOOM = "graella/UNZOOM";
```

Selectors

// Els selectors cal que diguin
de quin reducer venen

```
// counter
export function getCount(state) {
  return state.counter;
}

// graella
export function getZoomed(state) {
  return state.graella;
}

export function isZoomed(state) {
  return state.graella !== null;
}
```


Més d'un comptador?

Cal usar un array o un objecte per guardar diversos comptadors:

- Usem un array si els comptadors s'identifiquen per número.
- Usem un objecte si els comptadors s'identifiquen per nom.

```
const countersPerNumero = [  
  9, // state.counters[0]  
  5, // state.counters[1]  
  7, // state.counters[2]  
];
```


```
const countersPerNom = {  
  entrades: 9, // state.counters['entrades']  
  sortides: 5, // state.counters['sortides']  
  snaks: 7,    // state.counters['snaks']  
}
```

Counters per Nombre: Action Creator

```
const INCREMENT = "counters/INCREMENT";  
export function increment(counter, amount) {  
  return { type: INCREMENT, counter, amount };  
}
```

Counters per Nombre: Reducer

```
export function counters(state = [], action) {  
  switch (action.type) {  
    case INCREMENT: {  
      const copy = [...state];  
      if (!copy[action.counter]) copy[action.counter] = 0;  
      copy[action.counter] += action.amount;  
      return copy;  
    }  
    default:  
      return state;  
  }  
}
```

 // Abans de modificar, cal copiar

Counters per Nombre: Selector

```
export function getCounters(state) {  
  return state.counters;  
}
```

```
export function getCount(state, { counter }) {  
  return getCounters(state)[counter];  
}
```



// El patró del selector diu que
el segon argument és amb
{ objecte de props }

Counters per Nombre: <ListCounters />

```
import { useSelector } from "react-redux";
import { getCounters } from "../counters";
import Counter from "../Counter";

export default function ListCounters() {
  const counters = useSelector(getCounters);
  return (
    <>
      {counters.map((count, index) => (
        <Counter key={index} counter={index} />
      ))}
    </>
  );
}
```

Counters per Nombre: <Counter counter={...} />

```
import { useSelector, useDispatch } from "react-redux";
import { getCount, increment } from "../counters";

export default function Counter({ counter }) {
  const count = useSelector(
    (state) => getCount(state, { counter })
  ); const dispatch = useDispatch();

  return (
    <div>
      <h1>Counter {counter}</h1>
      <p>{count}</p>
      <button onClick={() => dispatch(increment(counter, 1)) }>+</button>
    </div>
  );
```

Counters per Nom: Coincideix en

Action Creator

```
const INCREMENT = "counters/INCREMENT";
export function increment(counter, amount) {
  return { type: INCREMENT, counter, amount };
}
```

<Counter counter={...} />

```
import { useSelector, useDispatch } from "react-redux";
import { getCount, increment } from "../counters";


export default function Counter({ counter }) {
  const count = useSelector((state) => getCount(state, { counter }));
  const dispatch = useDispatch();

  return (
    <div>
      <h1>Counter {counter}</h1>
      <p>{count}</p>
      <button onClick={() => dispatch(increment(counter, 1))}></button>
    </div>
  );
}
```

Selectors

```
export function getCounters(state) {
  return state.counters;
}

export function getCount(state, { counter }) {
  return getCounters(state)[counter];
}
```



// Perquè [...] és per trobar elements en un array per index, valors en un objecte per clau.

Counters per Nom: Reducer (només 2 canvis)

```
export function counters(state = {}, action) {  
  switch (action.type) {  
    case INCREMENT: {  
      const copy = { ...state };  
      if (!copy[action.counter]) copy[action.counter] = 0;  
      copy[action.counter] += action.amount;  
      return copy;  
    }  
    default:  
      return state;  
  }  
}
```


Counters per Nom: <ListCounters /> (1 canvi)

```
import { useSelector } from "react-redux";
import { getCounters } from "../counters";
import Counter from "../Counter";

export default function ListCounters() {
  const counters = useSelector(getCounters);
  return (
    <>
      {Object.keys(counters).map((count, index) => (
        <Counter key={index} counter={index} />
      ))}
    </>
  );
}
```

Nombre vs Nom

Nombre

- Creix/Decreix amb el temps
- L'ordre pot ser rellevant
- Difícil de determinar l'identificador

Nom

- Son entitats amb vida pròpia
- L'ordre pot ser irrellevant
- Tenen identificador propi

Estat Normalitzat

Nombre + Nom

Dos sub-estats: byId, allIds

Patró a aplicar quan les dades són objectes amb id.

```
{  
  counters: {  
    byId: {  
      entrades: { id: "entrades", count: 9 },  
      sortides: { id: "sortides", count: 5 },  
      snacks: { id: "snacks", count: 7 },  
    },  
    allIds: ["entrades", "sortides", "snacks"],  
  },  
  ...  
}
```

byId (nom)

El byId conté els objectes amb la clau id.

```
{  
  counters: {  
    byId: {  
      entrades: { id: "entrades", count: 9 },  
      sortides: { id: "sortides", count: 5 },  
      snacks: { id: "snacks", count: 7 },  
    },  
    allIds: ["entrades", "sortides", "snacks"],  
  },  
  ...  
}
```

allIds (nombre)

El allIds conté un array amb tots els ids.

```
{
  counters: {
    byId: {
      entrades: { id: "entrades", count: 9 },
      sortides: { id: "sortides", count: 5 },
      snacks: { id: "snacks", count: 7 },
    },
    allIds: ["entrades", "sortides", "snacks"],
  },
  ...
}
```

combineReducers({ byId, allIds })

```
// counter.js
import { combineReducers } from "redux";
export const counter = combineReducers({
  byId,
  allIds,
});
```

```
// store.js
import { counter } from "../counter";
import { combineReducers } from "redux";
const reducer = combineReducers({
  counter,
});
const store = createStore(reducer);
export default store;
```

Reducer allIds

```
function allIds(state = [], action) {  
  switch (action.type) {  
    case INCREMENT:  
      if (state.includes(action.id)) return state;  
      return [...state, action.id];  
    default:  
      return state;  
  }  
}
```


Reducer allIds (alt)

```
function allIds(state = [], action) {  
  switch (action.type) {  
    case CREATE:  
      return [...state, action.id];  
    case DELETE:  
      return state.filter((id) => id !== action.id);  
    default:  
      return state;  
  }  
}
```

Reducer byIds

```
function byId(state = {}, action) {  
  switch (action.type) {  
    case INCREMENT: {  
      const stateCopy = { ...state };  
      const counterCopy = {  
        id: action.id,  
        count: 0,  
        ...state[action.id],  
      };  
      counterCopy.count += 1;  
      stateCopy[action.id] = counterCopy;  
      return stateCopy;  
    }  
    default: return state; } }  
}
```

Reducer byIds (alt):

```
function byId(state = {}, action) {  
  switch (action.type) {  
    case CREATE: return { ...state,  
      [action.id]: { id: action.id, value: action.value } };  
    case DELETE:  
      const { [action.id]: removed, ...copy } = { ...state };  
      return copy;  
    case INCREMENT: return { ...state,  
      [action.id]: {  
        ...state[action.id],  
        value: state[action.id].value + action.amount,  
      },  
    };  
    default:  
      return state; } }
```

Actions & Action Creators

```
const INCREMENT = "counters/INCREMENT";  
export function increment(id, amount) {  
  return { type: INCREMENT, id, amount };  
}
```

Selectors

```
function getCounters(state) {  
  return state.counters;  
}  
  
export function getCounter(state, { id }) {  
  return getCounters(state).byId[id];  
}  
  
export function getCount(state, { id }) {  
  return getCounter(state, { id }).count;  
}  
  
export function getAllCounterIds(state) {  
  return getCounters(state).allIds;  
}
```

<ListCounters />

```
import { useSelector } from "react-redux";
import { getAllCounterIds } from "../counters";
import Counter from "../Counter";

export default function ListCounters() {
  const ids = useSelector(getAllCounterIds);
  return (
    <>
      {ids.map((id) => (
        <Counter key={id} counter={id} />
      ))}
    </>
  );
}
```

// És molt més eficient!

<Counter id={...} />

```
import { useSelector, useDispatch } from "react-redux";
import { getCount, increment } from "../counters";

export default function Counter({ id }) {
  const count = useSelector(
    (state) => getCount(state, { id })
  ); const dispatch = useDispatch();

  return (
    <div>
      <h1>Counter {id}</h1>
      <p>{count}</p>
      <button onClick={() => dispatch(increment(id, 1))}>+</button>
    </div> ); }
```

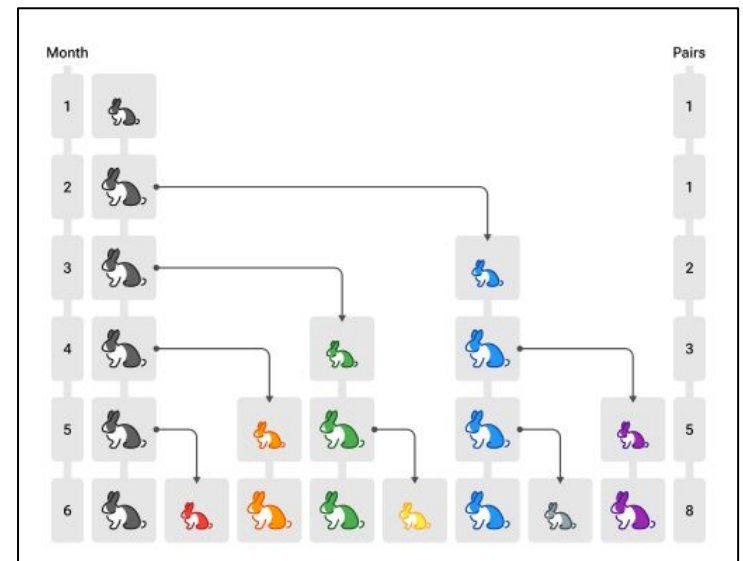
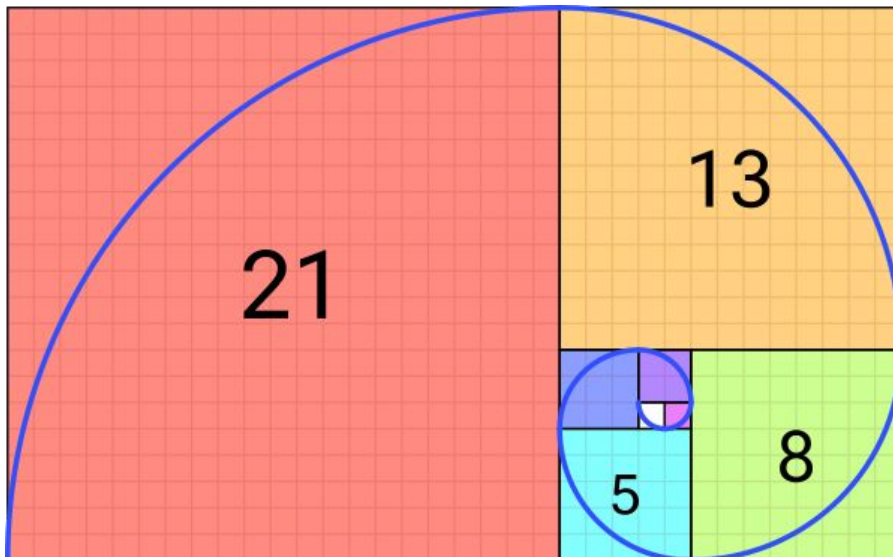
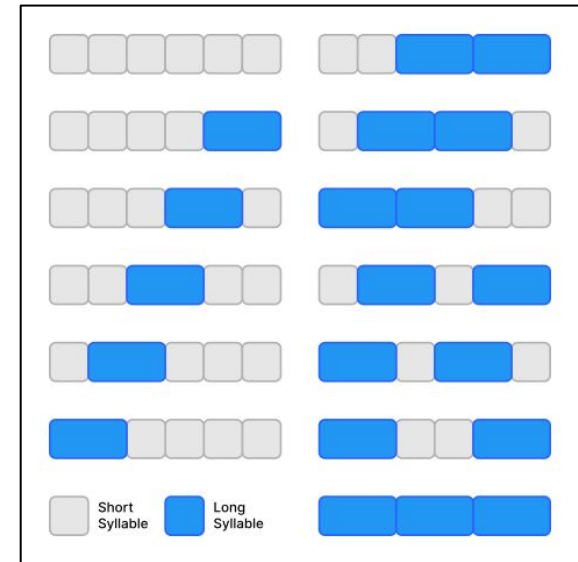
reselect:
createSelector

Fibonacci

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$



Fibonacci - Recursiu

$$F_0 = 0; F_1 = 1; F_n = F_{n-1} + F_{n-2}$$

```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

Fibonacci - Recursiu

$$F_0 = 0; F_1 = 1; F_n = F_{n-1} + F_{n-2}$$

```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

Fibonacci - Recursiu

$$F_0 = 0; F_1 = 1; F_n = F_{n-1} + F_{n-2}$$

```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

Fibonacci - Recursiu

$$F_0 = 0; F_1 = 1; F_n = F_{n-1} + F_{n-2}$$

```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

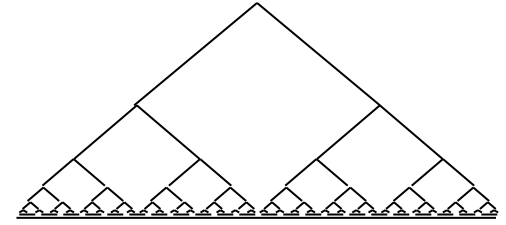
Fibonacci - Valors

```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

- > Verifiquen els fib(n) tal que n va de zero a 8, i també el fib(20)
- > Quan dona el fib(30)? I el fib(40)? I el fib(50)? I el fib(100)?

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

Fibonacci - Que ha passat?

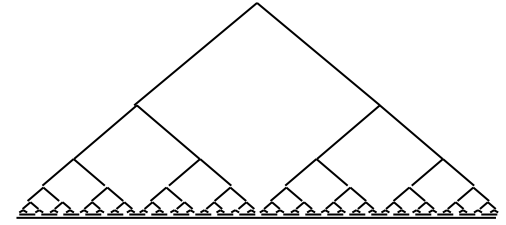


```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

Resultes que:

- fib(0) i fib(1) hi ha 1 crida
- fib(2) hi ha 2 crides
- fib(3) hi ha 3 crides
- fib(4) hi ha 5
- fib(5) hi ha 8
- fib(10) hi ha 86
- fib(20) hi ha +10.000
- fib(30) hi ha +1.000.000
- fib(50) hi ha +20.000 milions
- fib(100) hi ha ???

Fibonacci - Que ha passat?

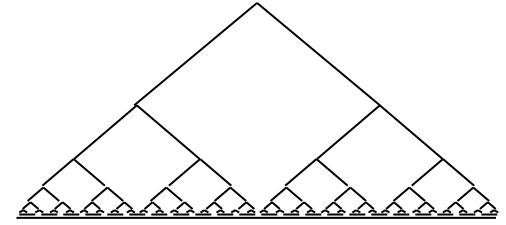


```
function fib(n) {  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  return fib(n - 1) + fib(n - 2);  
}
```

Resultes que:

- fib(0) i fib(1) hi ha 1 crida
- fib(2) hi ha 2 crides
- fib(3) hi ha 3 crides
- fib(4) hi ha 5
- fib(5) hi ha 8
- fib(10) hi ha 86
- fib(20) hi ha +10.000
- fib(30) hi ha +1.000.000
- fib(50) hi ha +20.000 milions
- fib(100) hi ha +5 x 10²⁰

Fibonacci - I si tinguessim memoria?



```
const memoria = [];  
function fib(n) {  
  if (memoria[n] !== undefined) return memoria[n];  
  if (n === 0) return 0;  
  if (n === 1) return 1;  
  memoria[n] = fib(n - 1) + fib(n - 2);  
  return memoria[n];  
}
```

Ara torna a calcular:

- fib(1)? fib(5)? fib(10)? fib(20)? fib(50)?
- fib(100)?
- fib(1000)?

3364476487643178326662161200510754331030214846068006390656476997468008144216666236815559551363373
40255820653326808361593737347904838652682630408924630564318873545443695598274916066020998841839
338646527313000888302692356736131351175792974378544137521305205043477016022647583189065278908551543
661595829872796829875106312005754287834532155151038708182989697916131278562650331954871402142875326
981879620469360978799003509623022910263681314931952756302278376284415403605844025721143349611800
230912082870460889239623288354615057765832712525460935911282039252853934346209042452489294039017
062338889910858410651831733604374707379085526317643257339937128719375877468974799263058370657428301
616374089691784263786242128352581128205163702980893320999057079200643674262023897831114700540749
9845925036063356093388383192338678305613643535189213327973290813373264265263398976392272340788292
81779535805709936910491754708089318410561463223382174656373212482263830921032977016480547262438423
748624114530938122065649140327510866433945175121615265453613311131404243685480510676584349352383695
965342807176877532834823434555736671973139274627362910821067928078471803532913117677892465908993863
545932789452377767440619224033763867400402133034329749690202832814593341882681768389307200363479
5623117103101291953169794607632737589253530772552375943788434504067715555779056450443016640119462580
9722167297586150269684431469520346149322911059706762432685159928347098912847067408620085871350162
603120719031720860940812983215810772820763531866246112782455372085323653057759564300725177443150515
39600905168603220349163222640885248852433158051534849622434848299380905070483482449327453732624
5677558790891871908036620580095947431500524025327097469953187707243768259074199396322659841474981
9360928522394503970716544315642132815768890805878318340491743455627052022356484649519611246026831
39709750693826487066132645076650746115126775227486215986425307112984411826226610571635150692600298
61704945425047491378115154139941550671256271197133252763631939606902895650288268608362241082050562
430701794976171121233066073310059947366875n

fib(10.000n)

```
const memoria = {};  
function fib(n) {  
  if (memoria[n] !== undefined) return memoria[n];  
  if (n === 0n) return 0n;  
  if (n === 1n) return 1n;  
  memoria[n] = fib(n - 2n) + fib(n - 1n);  
  return memoria[n];  
}
```

El poder de la memorització

La memorització és recordar les últimes crides i retornar un valor ja calculat anteriorment:

Donada una funció amb argument A:

- Si tinc a la memòria el resultat per A, retorno el resultat
- Si no:
 - Càlculo el resultat
 - El guardo a la memòria
 - I retorno el resultat

Memorització d'un

Per al nostre objectiu, pintar una vista, només necessitem guardar un resultat.

Donada una funció:

- Si repeteix la crida amb l'últim argument, retorno el resultat
- Si no:
 - Càlculo el resultat
 - Guardo a la memòria el resultat i l'argument
 - I retorno el resultat

Memorització d'un

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

- Si repeteix la crida amb l'últim argument, retorno el resultat

Memorització d'un

- Si no:
 - Càlculo el resultat
 - Guardo a la memòria el resultat i l'argument
 - I retorno el resultat

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

- Si repeteix la crida amb l'últim argument, retorno el resultat
- Si no:
 - Càlculo el resultat
 - Guardo a la memòria el resultat i l'argument
 - I retorno el resultat

Memorització d'un

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

- Si repeteix la crida amb l'últim argument, retorno el resultat
- Si no:
 - Càlculo el resultat
 - Guardo a la memòria el resultat i l'argument
 - I retorno el resultat

Memorització d'un

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```


- Si repeteix la crida amb l'últim argument, retorno el resultat
- Si no:
 - Càlculo el resultat
 - Guardo a la memòria el resultat i l'argument
 - I retorno el resultat

Memorització d'un

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

Memorització d'un i Selectors

I si a l'estat hi ha per exemple també el toggle, cal tornar a calcular si només canvia el toggle? Que cal mirar de fet?

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

Memorització d'un i Selectors

Però només necessitem mirar si canvia el taulell.

```
let anteriorState;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  if (state === anteriorState) return anteriorResultat;  
  const taulell = selectTaulell(state);  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorState = state;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

Memorització d'un i Selectors

Només necessitem mirar si canvia el taulell.

```
let anteriorTaulell;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  const taulell = selectTaulell(state);  
  if (taulell === anteriorTaulell) return anteriorResultat;  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorTaulell = taulell;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

Memorització d'un i Selectors

Però només una part petita és el càlcul, la resta és gestió.

```
let anteriorTaulell;  
let anteriorResultat;  
export function selectCasellesGuanyadores(state) {  
  const taulell = selectTaulell(state);  
  if (taulell === anteriorTaulell) return anteriorResultat;  
  let resultat = combinacionsGuanyadores.find(  
    ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
  );  
  anteriorTaulell = taulell;  
  anteriorResultat = resultat;  
  return resultat;  
}
```

Memorització d'un, selectors i reselect

Reselect fa la memorització per nosaltres.



(El resultat és la mateixa funció que hem vist abans)

```
import { createSelector } from "reselect";  
export const selectCasellesGuanyadores = createSelector(  
  selectTaulell,  
  (taulell) =>  
    combinacionsGuanyadores.find(  
      ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
    )  
);
```

Memorització d'un, selectors i reselect

Reselect fa la memorització per nosaltres.

(El resultat és la mateixa funció que hem vist abans)

```
import { createSelector } from "reselect";  
export const selectCasellesGuanyadores = createSelector(  
  selectTaulell,  // Crida a selectTaulell, mira si el  
  (taulell) =>  resultat canvia, i si canvia, crida a la  
    combinacionsGuanyadores.find(  
      ([a, b]) => taulell[a] && taulell[a] === taulell[b]  
    )  
);
```

Quan cal usar memorització?

- El càlcul és costós
- Quan retorna un objecte nou
- Quan retorna un array nou

Més exemples

```
import { createSelector } from "reselect";

function selectMoviesById(state) {
  return state.movies.byId;
}

export const selectAllMovies = (state) =>
  createSelector(selectMoviesById, (moviesById) =>
    Object.values(moviesById)
  );
```

Us a React sense Props

```
import { useSelector } from "react-redux";
import { selectAllMovies } from "../selectors";
import { MoviesList } from "../MoviesList";

export function AllMoviesList() {
  const movies = useSelector(selectAllMovies);
  return <MoviesList movies={movies} />;
}
```

Més exemples amb Props

```
function selectPropName(state, { name }) {  
  return name;  
}
```

```
export function makeSelectMovieByName() {  
  return createSelector(  
    listMovies,  
    selectPropName,  
    (movies, name) =>  
      movies.find((m) => m.name === name)  
  );  
}
```

Us a React amb Props

```
import { useMemo } from "react";
import { useSelector } from "react-redux";
import { makeSelectMovieByName } from "../selectors";
import { MovieDetails } from "../MovieDetails";

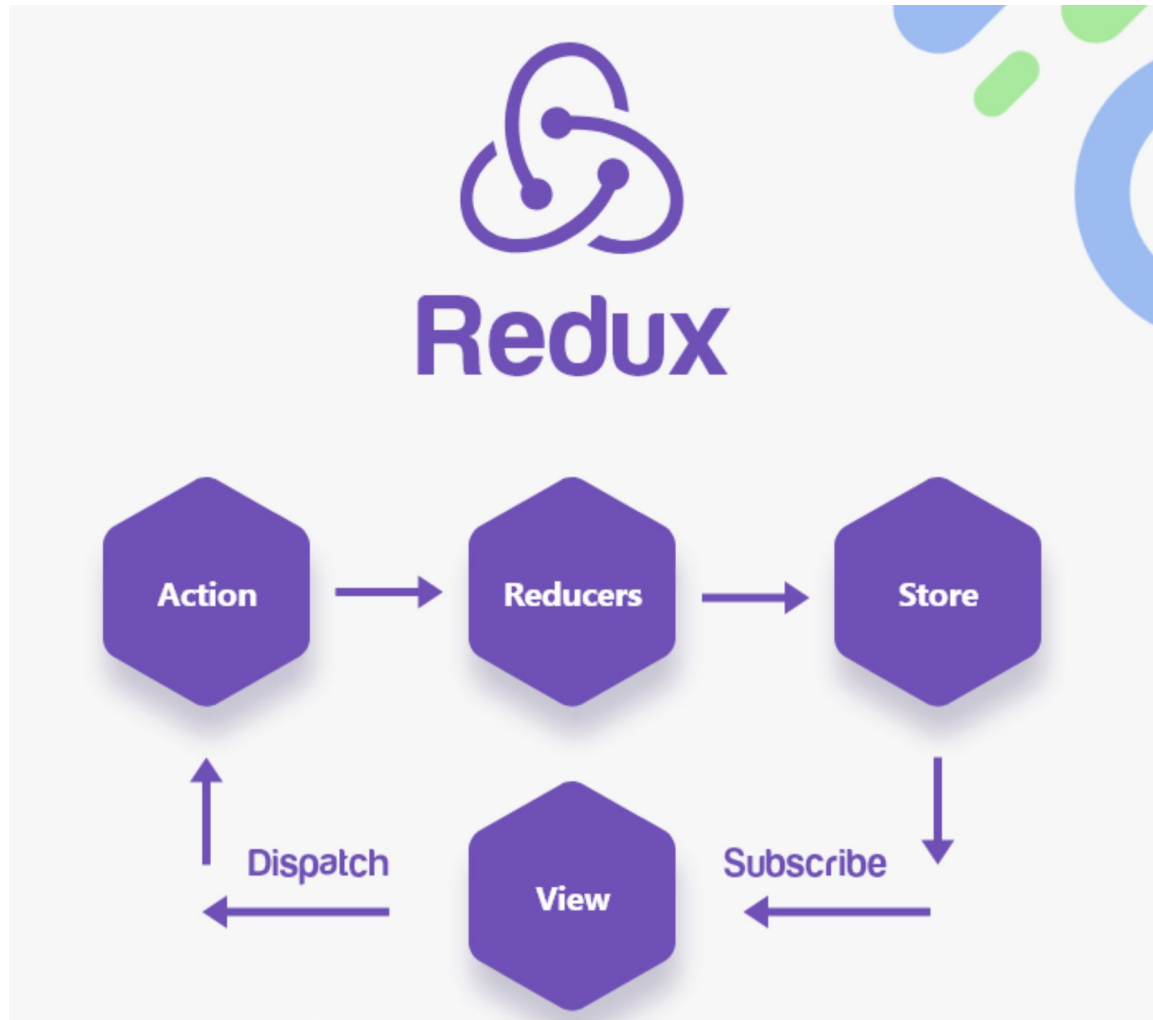
export function NamedMovieDetails({ name }) {
  const selectMovieByName = useMemo(
    makeSelectMovieByName,
    []
  );
  const movie = useSelector((state) =>
    selectMovieByName(state, { name })
  );
  return <MovieDetails movie={movie} />;
}
```

Quan cal usar els makeSelectors?

- Quan es necessita usar memorització i té una prop, o més
 - Perquè cada component cal que tingui la seva propia memorització

Pistes TodoList App

Imatge Redux



Estat inicial

Exemple:

```
const initialState = [  
  {  
    content: "Primera tasca",  
    done: true,  
  },  
  {  
    content: "Segona tasca",  
    done: false,  
  },  
];
```


Features

1. Llista de tasques
2. Afegir tasca
3. Eliminar tasca
4. Reset
5. Todo/Done
6. Filtres



Passos a seguir

1. Fer la UI
2. Pensar i afegir l'action
3. Pensar i afegir el reducer
4. Afegir el dispatch