Ajuntament de Barcelona

Cibernàrium

Introducció a NodeJs

Capacitació Tecnològica
per a Professionals i Empreses

Formador:
Oriol Tinoco Marco
Formador &
CTO en Worktocloud, S.L.

Barcelonactiva

Barcelona Activa: Qui som?

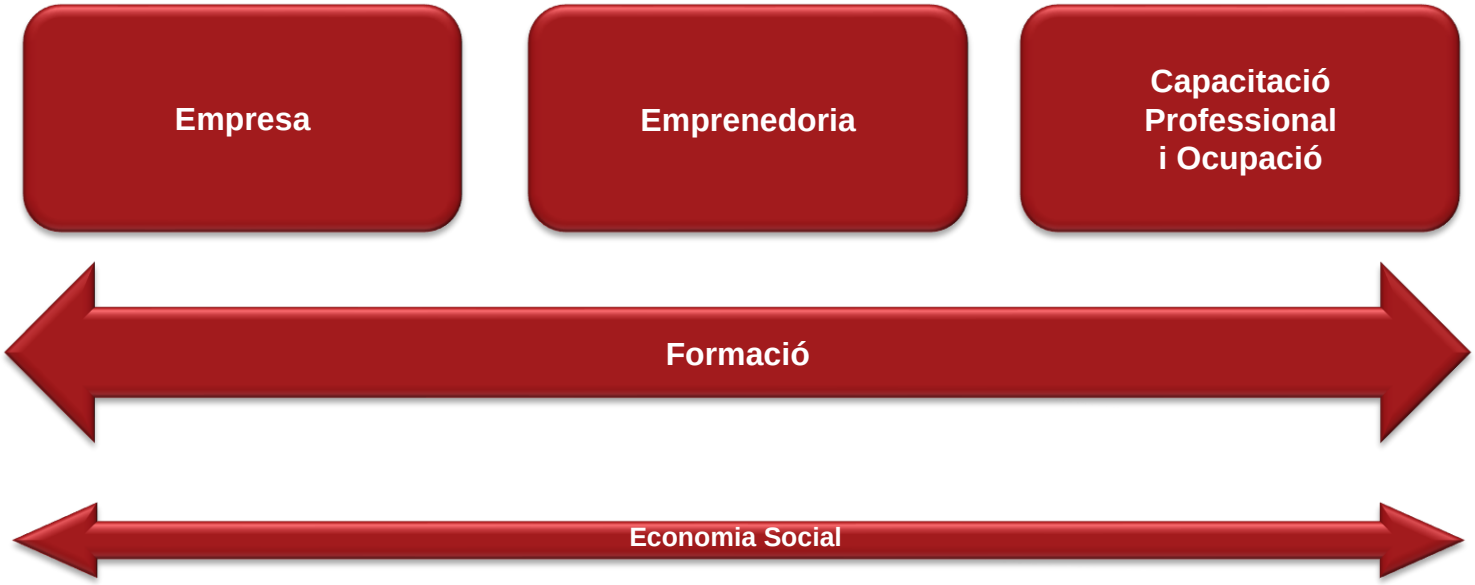
Barcelona Activa, integrada en l'àrea d'Economia, Empresa i Ocupació, és l'organització executora de les polítiques de promoció econòmica de l'Ajuntament de Barcelona.

Des de fa 25 anys impulsa el creixement econòmic de Barcelona i el seu àmbit d'influència donant suport a les empreses, la iniciativa emprenedora i l'ocupació, alhora que promociona la ciutat internacionalment i els seus sectors estratègics; en clau de proximitat al territori.

Barcelona Activa va ser guanyadora del Gran Premi del Jurat 2011, atorgat per la DG d'Empresa i Indústria de la Comissió Europea en el marc dels *European Enterprise Awards*, per la iniciativa empresarial més creativa i inspiradora d'Europa.

Àrees d'activitat de Barcelona Activa

Barcelona Activa s'estructura en tres grans blocs de serveis a les **Empreses**, a l'**Emprenedoria** i a la **Ocupació**. La **Formació** és un instrument transversal present en els tres blocs, així com també tot el relacionat amb l'economia social.



Una xarxa d'Equipaments Especialitzats

Xarxa de Proximitat

13 antenes Cibernàrium a biblioteques
10 punts d'atenció en Ocupació

8 Hooks

Adéu als Components de Classe?

Els hooks (o ganxos en català) són una eina relativament nova, van aparèixer el 2018 amb la versió 16.8 de React. A partir d'aquell moment, els components funcionals van guanyar molta potència amb la possibilitat de gestionar el seu propi estat entre altres funcionalitats.

Això ha causat que els components de clase, que fins el moment tenien el monopoli d'aquest tipus de funcionalitats, perdin el seu protagonisme.

En quant el seu ús, els hooks de React no son més que funcions que estan connectades directament el cicle de vida d'un component.

Què són els cicles de vida d'un component de React?

En REACT, cada component pasa per diferents fases a la seva vida, tal com una persona pot pasar de ser un nen a una persona adulta, per després convertir-se en una persona gran. Els components de React passen per tres fases:

Mounting: És la primera fase i significa que el component està en procés d'inserir-se el contingut al DOM.

Updating: La segona, Updating, es diu quan el component està sent actualitzat. L'actualització d'un component es produeix quan hi ha un canvi a l'estat del component o dels seus props.

Unmounting: La fase següent i l'última, és Unmounting, que s'anomena quan un component ha de ser eliminat del DOM.

Què són els cicles de vida d'un component de React?

Els dos hooks que veurem en aquesta càpsula, useState i useEffect, s'activen a la fase d'Updating.

Quan actualitzem l'estat del nostre component amb useState, l'aplicació ha de tornar a pintar el contingut del component a la pantalla per tenir en compte el canvi que hem introduït.

Després d'haver actualitzat el DOM, l'aplicació procedeix a executar useEffect, si és present al component.

Com utilitzar el Hook useState?

Perquè un component funcional tingui estat propi, podem fer ús del hook useState.

```
import React, { useState } from 'react';
```

useState() és una funció que crea internament una variable on podrem emmagatzemar l'estat del nostre component. Accepta un valor inicial per a aquesta variable i torna un array amb dos elements, el valor de la variable i la funció per modificar-la. Com el valor retornat per la funció és un array, podem descomposar-ho per accedir els seus elements de forma individual.

```
const [count, setCount] = useState(0);
```

Com utilitzar el Hook useState?

Veiem un exemple:

```
import React, { useState } from 'react';

function SimpleCounter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Has hecho click {count} veces</p>
      <button onClick={() => setCount(count + 1)}>
        Hazme click!
      </button>
    </div>
  );
}

ReactDOM.render(<SimpleCounter />, document.getElementById('root'));
```

Podeu afegir tants useState() com desitgeu, cada un d'ells per una variable diferente.

L'única condició és que és cridi des d'un nivell superior de còdi, no en un bloc.

Es important saber quan cridem la funció set d'un useState(), és sobreescriu el contingut de la variable.

Com utilitzar el Hook useEffect?

```
import React, { useEffect } from 'react';
```

Aquest hook normalment és usat per a la inicialització de variables, cridades a APIs o per netejar un component abans de desmuntar-ho del DOM.

És l'equivalent funcional a componentDidMount, componentDidUpdate, i componentWillUnmount als components de classe.

La crida a useEffect accepta una funció com a argument. Aquesta funció s'executa per defecte quan el component es renderitza per primera vegada, i després cada cop que el component s'actualitzi.

Com utilitzar el Hook useEffect?

Veiem un exemple:

```
import React, { useState, useEffect } from 'react';

function CompoundCounter() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Has hecho click ${count} veces`;
  });

  return (
    <div>
      <p>Has hecho click {count} veces</p>
      <button onClick={() => setCount(count + 1)}>
        Hazme click!
      </button>
    </div>
  );
}

ReactDOM.render(<CompoundCounter />, document.getElementById('root'));
```

Com podem apreciar la funció s'executarà quan el component és renderitzi per primer cop i quan s'actualitzi.

Com utilitzar el Hook useEffect?

També podem especificar quan s'ha d'executar aquesta funció amb un segon argument opcional que li podem indicar.

Per això n'hi ha prou amb afegir un segon paràmetre a la funció, amb la llista dels elements de què depèn. Si el valor d'un d'aquests elements que hem indicat canvia, la funció s'executarà amb la següent actualització.

```
useEffect(() => {
  document.title = `Has hecho click ${count} veces`;
}, [count]);
```

DE FILL A PARE: COMPONENT PARE

Com és pot observar cada cop que el fill envia una dada nova, invoca la funció fletxa cridant la funció setData i que actualitzarà la variable data del Hook useState().

App.js M

src > App.js > App > childToParent

```
1  import './App.css';
2  import FunctionalComponent from './components/functionalComponent';
3  import {useState} from 'react'
4
5  function App() {
6
7    const [data, setData] = useState('')
8
9    const childToParent = (dataFromChild) => {
10     console.log('child to parent', dataFromChild);
11     setData(dataFromChild)
12   }
13
14   return (
15     <>
16
17     {data}
18     <FunctionalComponent childToParent = { childToParent }
19    />
20     </>
21   );
22 }
23
24 export default App;
25
```

DE FILL A PARE: COMPONENT FILL

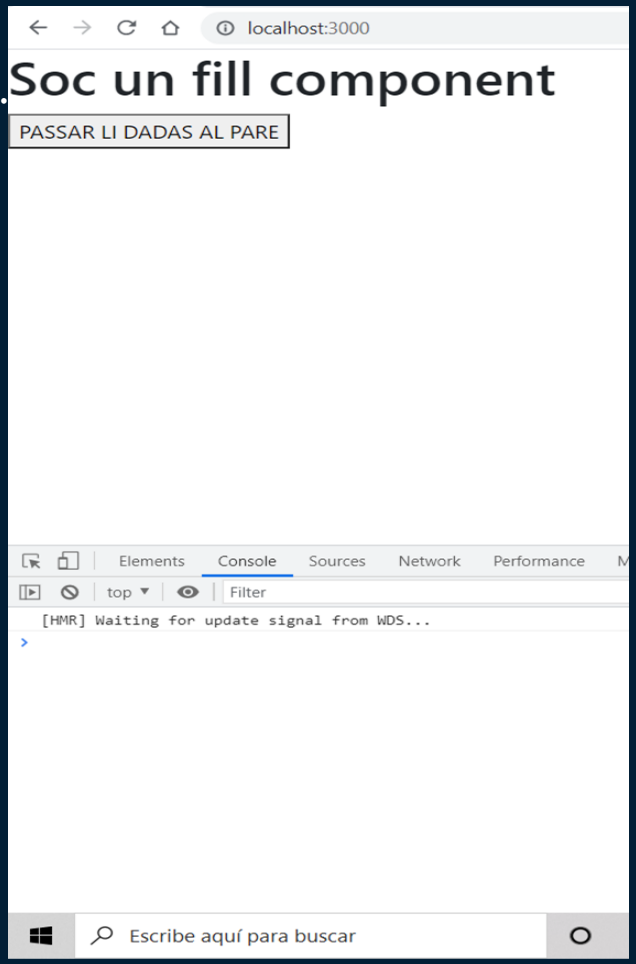
En la constant data indiquem la dada a transmetre el component pare i ho realitzarà en el moment que l'usuari cliqui el botó.

functionalComponent.js U

src > components > functionalComponent.js > FunctionalComponent

```
1   import React from 'react';
2   import './styles/functionalComp.css';
3
4   const FunctionalComponent = ({ childToParent }) => {
5
6       const data = 'Info del fill cap al pare';
7
8       return (
9         <>
10          <h1>Soc un fill component</h1>
11          <button onClick={ ()=> childToParent(data) } >PASSAR LI DADAS AL PARE </button>
12        </>
13        )
14    }
15
16    export default FunctionalComponent;
```

DE FILL A PARE: RESULTAT



Gràcies!

Formador Oriol Tinoco

Linkedin: https://es.linkedin.com/in/oriol-tinoco-marco-589b9051

Email: orioltinoco@worktocloud.es

Barcelonactiva

Ajuntament de Barcelona

bcn.cat/barcelonactiva
bcn.cat/cibernarium