

## Tasca 1. Assenyala quin error surt per consola i corregeix el codi perquè desaparegui l'error.

Este error indica que el sistema no puede encontrar el archivo de mapa de fuentes (source map) para el módulo @jridgewell/set-array. Estos archivos de mapas de fuentes son utilizados para la depuración, pero no son esenciales para la ejecución del código.

```
Could not read source map for file:///C:/Users/ethan/Documents/e-commerce/node_modules/%40jridgewell/sourcemap-codec/dist/sourcemap-codec.umd.js: ENOENT: no such file or directory, open 'c:\Users\ethan\Documents\ecommerce\node_modules\@jridgewell\sourcemap-codec\dist\sourcemap-codec.umd.js.map'
```

Esto significa que cuando el navegador o las herramientas de desarrollo intentan cargar el mapa de fuentes para esos archivos, no lo encuentran. **No es crítico** para el funcionamiento de la aplicación. Los **source maps** solo son útiles durante el desarrollo y depuración.

Para solucionarlo primero he instalado las dependencias necesarias para trabajar con Webpack, React y CSS:

```
npm install --save react react-dom
```

```
npm install --save-dev webpack webpack-cli webpack-dev-server style-loader css-loader  
babel-loader @babel/core @babel/preset-env @babel/preset-react html-webpack-plugin
```

He usado el archivo index.html como plantilla para que Webpack lo copie y lo modifique, incluyendo el bundle.js generado automáticamente.

```
<!DOCTYPE html>  
<html Lang="es">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <title>Mi Proyecto</title>  
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>  
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>  
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>  
    <link rel="stylesheet" href="styles.css">  
  </head>  
  <body>  
    <div id="app"></div>  
  </body>  
</html>
```

Configuracion de mi web.config.js:

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  mode: "development",
  entry: "./tasca1.js",
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
    clean: true,
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/, // archivos .js y .jsx
        exclude: /node_modules/,
        use: {
          loader: "babel-loader",
          options: {
            presets: ["@babel/preset-env", "@babel/preset-react"],
          },
        },
      },
    ],
  },
  resolve: {
    extensions: [".js", ".jsx"],
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: "./index.html",
      filename: "index.html",
      inject: "body",
    }),
  ],
  devServer: {
    static: path.join(__dirname, "dist"),
    compress: true,
    port: 9000,
  },
}
```

En caso de no querer utilizar los source maps los podremos evitar añadiendo la siguiente línea:

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  devtool: false,
  mode: "development",
  entry: "./tasca1.js",
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
  },
}
```

Para arrancar el servidor lo haremos con el comando `npx webpack serve`.

**Tasca 2. El component Gallery conté un HTML molt similar per a dos perfils. Extreu el component Profile per reduir la duplicació. Haureu d'escollir quins props li passareu.**

He creado el componente Profile que recibe datos mediante props. Tambien he simplificado Gallery, reutilizando Profile para cada científico.

```
function Profile({ name, image, profession, awards, discovery }) {
  return (
    <section className="profile">
      <h2>{name}</h2>
      <img className="avatar" src={image} alt={name} width={70} height={70} />
      <ul>
        <li><b>Profession:</b> {profession}</li>
        <li><b>Awards:</b> {awards}</li>
        <li><b>Discovered:</b> {discovery}</li>
      </ul>
    </section>
  );
}



function Gallery() {
  return (
    <div>
      <h1>Notable Scientists</h1>
      <Profile
        name="Maria Skłodowska-Curie"
        image="https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Marie_Curie_%281900%29.jpg/220px-Marie_Curie_%281900%29.jpg"
        profession="Physicist and Chemist"
        awards="4 (Nobel Prize in Physics, Nobel Prize in Chemistry, Davy Medal, Matteucci Medal)"
        discovery="Polonium (chemical element)"
      />
      <Profile
        name="Katsuko Saruhashi"
        image="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQZ6_EyUQ4_4967ox6xqp1FARawJYwMSIagFg&s"
        profession="Geochemist"
        awards="2 (Miyake Prize for Geochemistry, Tanaka Prize)"
        discovery="A method for measuring carbon dioxide in seawater"
      />
    </div>
  );
}
```

**Tasca 3. Extreu el component Card del component Profile i feu servir el prop children per passar-li diferent JSX a cada card.**

He extraído un nuevo componente Card para evitar la duplicación en Profile. Antes, el código repetía la misma estructura para mostrar una imagen y un texto en tarjetas separadas. Ahora, Card usa `props.children`, permitiendo que su contenido sea flexible y reutilizable. Esto mejora la claridad, reduce la redundancia y facilita el mantenimiento del código.

```
// Nuevo componente Card para evitar la duplicación de código
function Card({ children }) {
  return (
    <div className="card">
      <div className="card-content">
        {children}
      </div>
    </div>
  );
}
```

**Tasca 4a. Fes servir l'operador condicional (cond ? a: b) per renderitzar  si isPacked no es true.**

He modificado la función Item para utilizar el operador condicional (cond ? a : b). Antes, solo se mostraba el  si isPacked era true, pero ahora también se muestra  cuando isPacked es false. Esto mejora la claridad visual y hace que el estado de cada elemento sea más evidente para el usuario.

```
function Item({ name, isPacked }) {
  return (
    <li className="item">
      {name} {isPacked ? '✔' : '✗'}
    </li>
  );
}

function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item isPacked={true} name="Space suit" />
        <Item isPacked={true} name="Helmet with a golden leaf" />
        <Item isPacked={false} name="Photo of Tam" />
      </ul>
    </section>
  );
}

const root = ReactDOM.createRoot(app);
root.render(<HomePage />);
}
```

**Tasca 5. Construeix una llista de receptes a partir de l'array recipes. Per cada recepta a l'array, mostra el seu nom en un <h2> i la seva llista d'ingredients en una <ul>.**

He actualizado el componente RecipeList para generar dinámicamente una lista de recetas a partir del array recipes. Usando .map(), ahora cada receta se muestra con un <h2> para su nombre y una lista <ul> con sus ingredientes, lo que elimina la necesidad de escribir código repetitivo. También he añadido claves (key) en las listas para optimizar la renderización en React.

```
function RecipeList() {
  return (
    <div>
      <h1>Recipes</h1>
      {recipes.map((recipe) => (
        <div key={recipe.id}>
          <h2>{recipe.name}</h2>
          <ul>
            {recipe.ingredients.map((ingredient, index) => (
              <li key={index}>{ingredient}</li>
            ))}
          </ul>
        </div>
      ))}
    </div>
  );
}

const root = ReactDOM.createRoot(app);
root.render(<HomePage />);
}
```

**Tasca 6. El component RecipeList conté dos map niuats. Per simplificar-lo, extreu el component Recipe amb els props id, name i ingredients.**

Lo que hago es sacar el componente Recipe para que se encargue de mostrar los detalles de cada receta, como el nombre e ingredientes. De esta manera, el componente RecipeList ya no tiene que hacer tanto trabajo, solo recorre el array de recetas y pasa cada una al componente Recipe. Esto hace que el código sea más limpio y fácil de entender, ya que cada componente tiene una tarea más clara: RecipeList maneja la lista y Recipe muestra la información de cada receta.

```
function renderApp() {
  const app = document.getElementById('app');

  // Componente HomePage
  function HomePage() {
    return (
      <div>
        <RecipeList />
      </div>
    );
  }

  const recipes = [{
    id: 'greek-salad',
    name: 'Greek Salad',
    ingredients: ['tomatoes', 'cucumber', 'onion', 'olives', 'feta']
  }, {
    id: 'hawaiian-pizza',
    name: 'Hawaiian Pizza',
    ingredients: ['pizza crust', 'pizza sauce', 'mozzarella', 'ham', 'pineapple']
  }, {
    id: 'hummus',
    name: 'Hummus',
    ingredients: ['chickpeas', 'olive oil', 'garlic cloves', 'lemon', 'tahini']
  }];

  function RecipeList() {
    return (
      <div>
        <h1>Recipes</h1>
      </div>
    );
  }

  const root = ReactDOM.createRoot(app);
  root.render(<HomePage />);
}
```

**Tasca 7. Dos compnents Profile es renderitzen un al costat de l'altre amb dades diferents. Premeu "Collapse" al primer perfil i després "Expand". Notareu que els dos perfils ara mostren la mateixa persona. Això és un error.**

**Trobeu la causa de l'error i solucioneu-lo.**

**Pista: el component Panel està perfectament bé, el problema es troba en els altres components**

Lo que hago es eliminar la variable global currentPerson que causaba el problema, porque estaba compartiendo los datos entre los perfiles. Ahora, paso los datos de cada perfil directamente como props a los componentes Header y Avatar. De esta forma, cada perfil maneja sus propios datos y no hay interferencia entre ellos cuando se hace "Collapse" o "Expand".

```
function Profile({ person }) {
  return (
    <Panel>
      <Header person={person} />
      <Avatar person={person} />
    </Panel>
  );
}

function Header({ person }) {
  return <h1>{person.name}</h1>;
}

function Avatar({ person }) {
  return (
    <img
      className="avatar"
      src={person.imageUrl}
      alt={person.name}
      width={50}
      height={50}
    />
  );
}

function Panel({ children }) {
  const [open, setOpen] = React.useState(true);
  return (
    <section className="panel">
      <button onClick={() => setOpen(!open)}>
        {open ? 'Collapse' : 'Expand'}
      </button>
      {open && children}
    </section>
  );
}
```

