



PARTE I: Introducción a REACT y Instalación

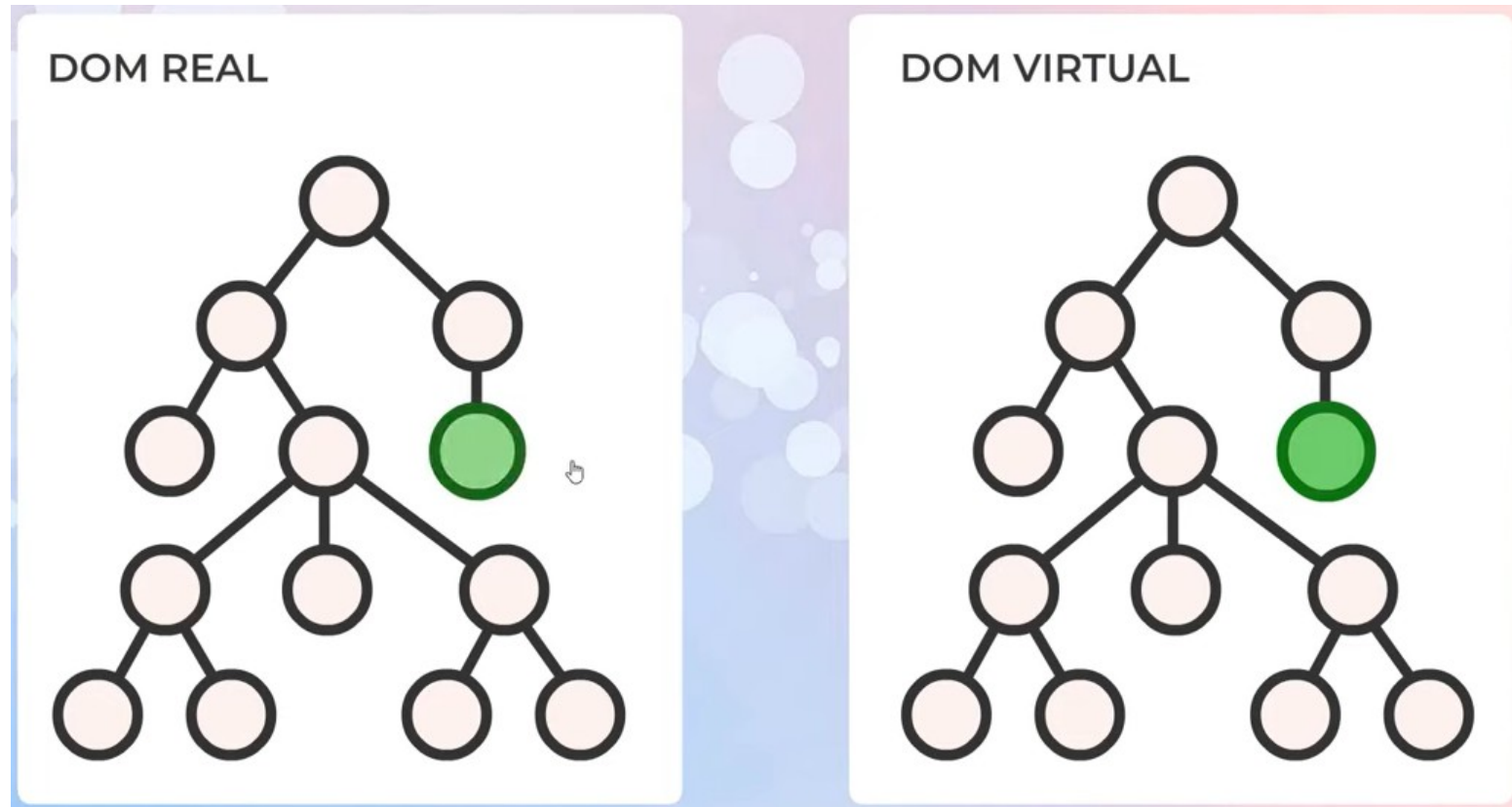
INTRODUCCION

INTRODUCCIÓN: ¿Que es REACT y para qué sirve?

- React permite desarrollar páginas web de una manera sencilla gracias a sus componentes reutilizables. Estos hacen posible usar un mismo elemento en varias partes del sitio o en otros sitios sin necesidad de volver a escribir todo el código.
- La finalidad de React es darnos herramientas que nos simplifiquen el trabajo a lo largo de todo el ciclo de vida de las aplicaciones web SPA (Single Page Application).
- Una aplicación SPA consta de una sola página, en la cual la navegación entre páginas, así como la carga de datos, se realiza de manera dinámica, casi instantánea, asíncronamente, haciendo llamadas AJAX al servidor, sin refrescar la página en ningún momento.

INTRODUCCIÓN: Virtual DOM

React fue desarrollado por Facebook para corregir los problemas de rendimiento de sus aplicaciones. Para ello desarrollaron una interesantísima funcionalidad llamada Virtual DOM

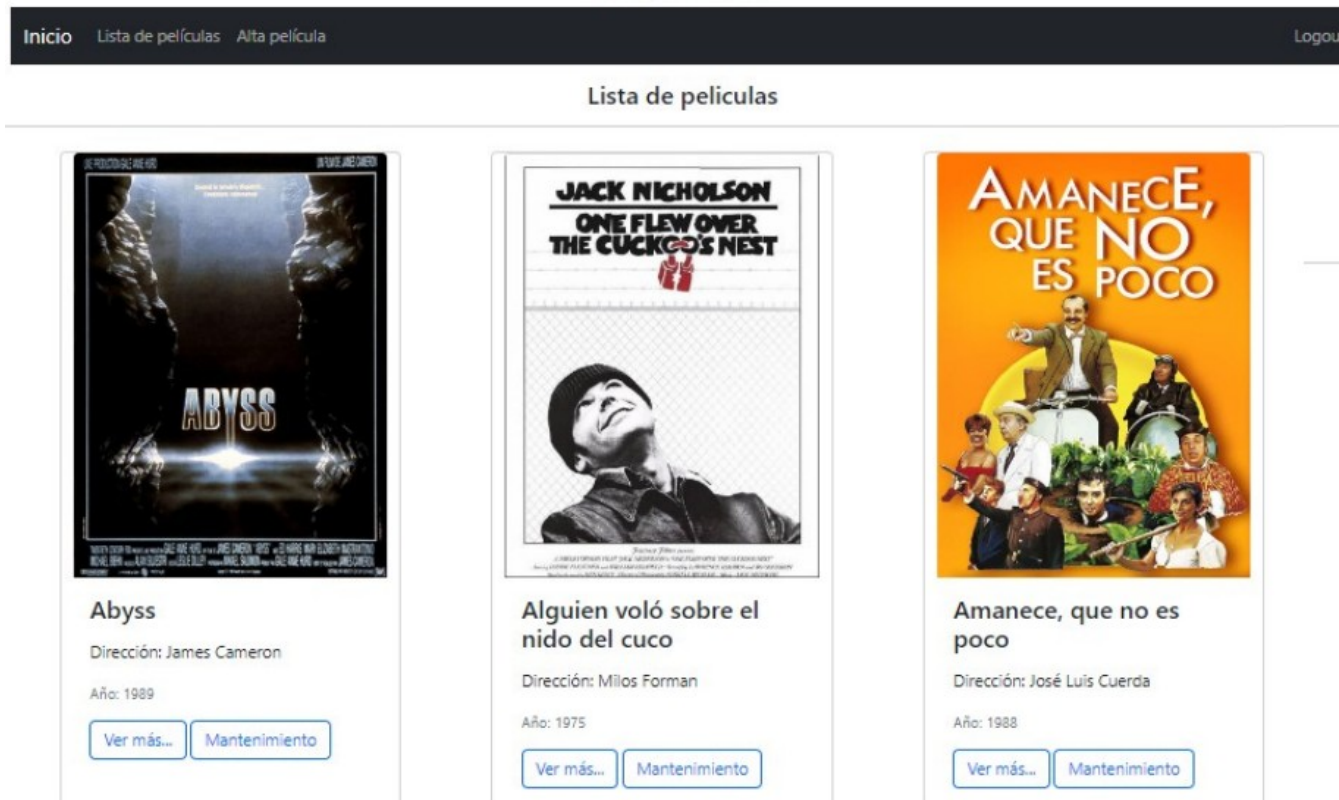


INTRODUCCIÓN: Virtual DOM

El Virtual DOM es una copia exacta del DOM real pero almacenada en memoria de forma que, siempre que haya que realizar un cambio en el DOM, este se realizará primero en el Virtual DOM y, posteriormente se trasladará al DOM real de una forma muy rápida (el renderizado de elementos del DOM suele ser muy costoso de forma que si solo se refresca el elemento que ha cambiado en el Virtual DOM, la carga de este elemento es muy eficiente)

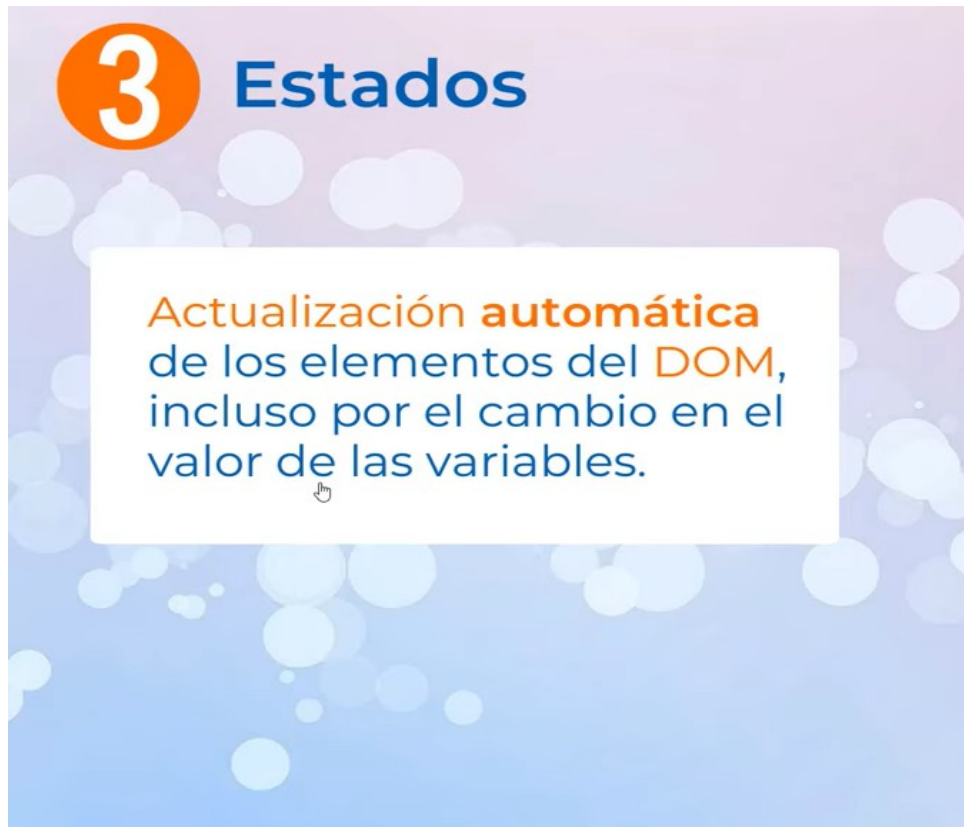
INTRODUCCIÓN: Componentes

Otra característica de React es que trabaja con componentes de forma que, cada elemento de un documento HTML (títulos, párrafos, botones, imágenes, etc, ...) es un componente que podemos reutilizar en múltiples secciones, de forma individual o agrupado con otros componentes



INTRODUCCIÓN: Estados

React conoce en todo momento el estado de la aplicación, es decir, el estado de cada una de las variables y arrays de forma que, el cambio de cualquiera de ellos provoca una actualización automática del DOM



INTRODUCCIÓN: SPA

React está especialmente indicado para trabajar con aplicaciones de tipo SPA basadas en componentes en donde solo existe una única página y, en ésta, se van cargando los diferentes componentes que correspondan a peticiones o acciones del usuario. Esto nos permitirá, aunque no es el objetivo de este curso, adaptar una aplicación a móvil utilizando React Native



4 SPA (Single Page Application)

Además, nos abre la puerta a otros entornos, como la **creación de aplicaciones totalmente nativas** para iOS y Android con un único desarrollo.



React Native

INSTALACIÓN

INSTALACION

El primer paso es instalarnos NodeJS, no porque vayamos a necesitar desarrollar servicios backend, sino porque Node viene con un gestor de dependencias llamado npm que necesitaremos para instalarnos las librerías y utilidades de React

Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

Descargar para Windows (x64)

18.17.1 LTS

Recomendado para la mayoría

20.5.1 Actual

Últimas características

[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

O eche un vistazo al [Programa de soporte a largo plazo \(LTS\)](#).

NOTA: Una vez instalado podemos ejecutar desde el cmd del sistema el comando **node -v** para comprobar que se ha instalado correctamente

INSTALACION

Es muy recomendable, una vez instalado nodeJs, que actualicemos **npm** a su última versión. Para ello, desde la línea de comandos tecleamos

npm install -g npm@latest

```
C:\Users\david>npm install -g npm@latest
added 1 package, and audited 250 packages in 4s
28 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

Y también limpiaremos la cache para actualizar y descargar las últimas versiones de las librerías que necesitamos

npm cache clean --force

```
C:\Users\david>npm cache clean --force
npm WARN using --force Recommended protections disabled.
```

INSTALACION

El siguiente paso es instalar REACT utilizando, desde el cmd del sistema:

`npm install -g create-react-app`

NOTA: el parámetro `-g` permitirá que react sea accesible desde cualquier carpeta

```
C:\Users\david>npm install -g create-react-app
npm WARN deprecated tar@2.2.2: This version of tar
  upgrade asap.
added 67 packages in 5s
```

INSTALACION

Una vez instalado React vamos a crear una carpeta (donde queramos) donde crearemos el primer proyecto

Nos situamos dentro de la carpeta desde el cmd del sistema y crearemos nuestro primer proyecto (que llamaremos **intoreact** - todo en minúsculas, react no acepta mayúsculas en los nombres de la aplicación a desarrollar-):

`create-react-app intoreact`

```
Success! Created intoreact at C:\xampp\htdocs\REACT\intoreact
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd intoreact
  npm start
```

INSTALACION

Una vez creado el proyecto vamos a visualizarlo desde el navegador, para ello desde el cmd del sistema, o desde el terminal de, por ejemplo VSC, entramos en la carpeta **introreact** y tecleamos

npm start

Veremos como se abre el navegador y se ejecuta un servidor en el puerto 3000 con la página de bienvenida de React

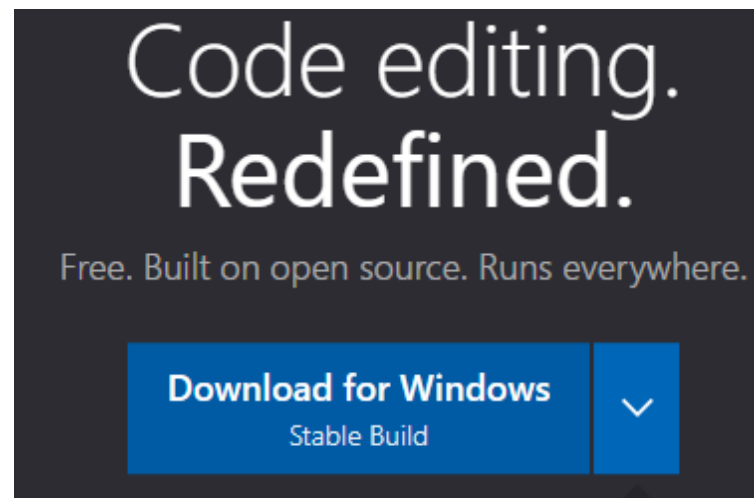


PRIMER PROYECTO

PRIMER PROYECTO

Vamos a modificar el proyecto para eliminar la página de bienvenida de React y sustituirla por código propio.

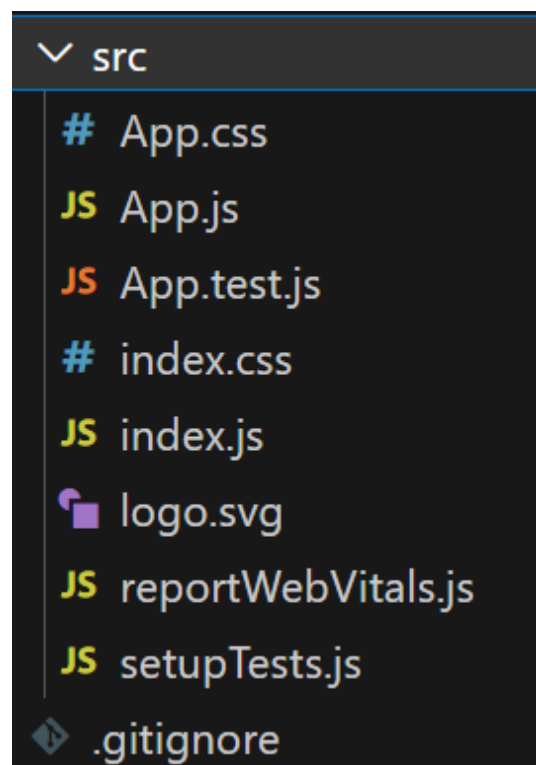
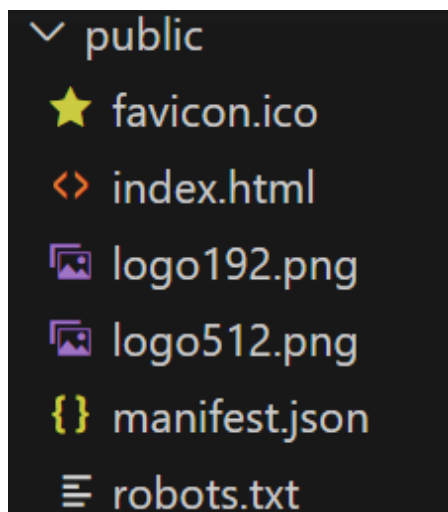
Para ello se recomienda la utilización del editor *Visual Studio Code*



<https://code.visualstudio.com/>

PRIMER PROYECTO

La estructura de carpeta que veremos será la siguiente:



PRIMER PROYECTO

public/index.html

Fichero de entrada a la aplicación. No hay que tocarlo a menos que necesitemos añadir librerías adicionales externas (por ejemplo bootstrap, jqueryui, etc...)

Veremos una caja `<div id="root"></div>` que utilizará react para inyectar todos los componentes de la aplicación

src/index.js

Archivo donde, además de importar otros archivos necesarios, que indica a react el componente de **index.html** donde debe inyectarse el resto de componentes

```
const root = ReactDOM.createRoot(document.getElementById('root'));
```

PRIMER PROYECTO

src/app.js

Fichero donde se encuentra el componente de bienvenida de React y fichero que modificaremos para añadir nuestro propio código

Todo lo que modifiquemos dentro de `<div className="App">` veremos como se visualiza inmediatamente en el navegador (siempre que tengamos el proyecpo en marcha)

`<p>Curso de REACT.</p>`

Curso de REACT.

Como podemos ver REACT utiliza una sintaxis para los componentes html muy particular ya que mezcla etiquetas html con javascript en lo que se conoce como código **JSX**

PRIMER PROYECTO

src/app.css

Fichero con las propiedades CSS que utilizará el componente que tenemos en **app.js**.

Fijaos que el fichero app.js contiene una instrucción **import** con este archivo css:

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

FICHERO APP.JS

Vamos borrar el contenido de la etiqueta `<div className="App">` y añadir nuestro propio contenido:

```
<div className="App">  
  <h1>Curso de React</h1>  
</div>
```

Todas las etiquetas llevarán un atributo específico de React llamado **className** (es como una **class** de css pero para uso de react. Podríamos decir que es el nombre de cada uno de los componentes)

Otra característica de react es que siempre tenemos que cerrar las etiquetas, incluso aquellas para las que normalmente no ponemos etiquetas de cierre

`<input type='text'></input>` o `
` en vez de `
`

FICHERO APP.JS

Si no cerreamos una etiqueta veremos como VSC ya nos avisa en rojo pero, además, veremos como en el cmd falla la compilación

```
8 | <input type='text'>
9 | </div>
```

```
ERROR in [eslint]
src\App.js
  Line 9:10:  Parsing error: Unterminated JSX contents. (9:10)

webpack compiled with 2 errors
```

E incluso en el navegador veremos

Compiled with problems:

Curso d

ERROR in ./src/App.js

Module build failed (from ./node_modules/babel-loader/lib/index.js):

SyntaxError: C:\xampp\htdocs\REACT\introreact\src\App.js: Unterminated JSX contents. (9:10)

```
7 |     <h1>Curso de React</h1>
8 |     <input type='text'>
> 9 | </div>
    |     ^
```

FICHERO APP.JS

¿Dónde colocamos el css?

Vamos a asignar un color de fondo a nuestro título. Para ello tenemos dos opciones

- Colocar el css en el fichero **src/app.css** de forma que este css solo aplicará al componente **app.js**
- Colocar el css en el fichero **src/index.css** de forma que este css aplicará a todos los componentes (normalmente aquí colocaremos el css genérico común a todos los componentes)

Lo colocaremos en app.css:

```
.App {  
  text-align: center;  
  background-color: aquamarine;  
}
```



Curso de React

FICHERO APP.JS

Podemos añadir más ficheros css dentro de la carpeta **src**. Por ejemplo podríamos crear un fichero **miCss.css** e incorporarlo dentro del fichero **App.js** con

```
import './miCss.css';
```


FICHERO APP.JS

¿Cómo integramos javascript?

Vamos a crear dentro de la función App una variable JS con nuestro nombre y la visualizaremos en el código JSX utilizando lo que se denomina data binding:

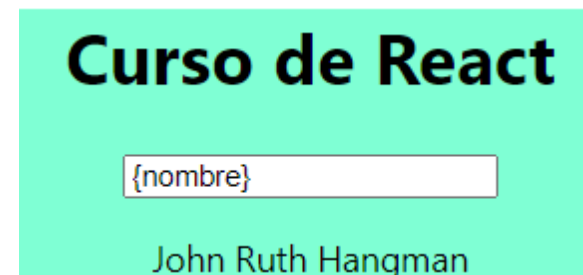
```
function App() {  
  let nombre = "Daisy Domergue";  
  return (  
    <div className="App">  
      <h1>Curso de React</h1>  
      <input type="text"></input>  
      <p>{nombre}</p>  
    </div>  
  );  
}
```



FICHERO APP.JS

También podemos inyectar nuevas etiquetas html pero utilizando una sintaxis especial sin las comillas :

```
function App() {  
  let caja = "<div>John Ruth Hangman</div>"  
  let caja = <div>John Ruth Hangman</div>  
  return (  
    <div className="App">  
      <h1>Curso de React</h1>  
      <input type='text'></input>  
      <p>{caja}</p>  
    </div>  
  );  
}
```



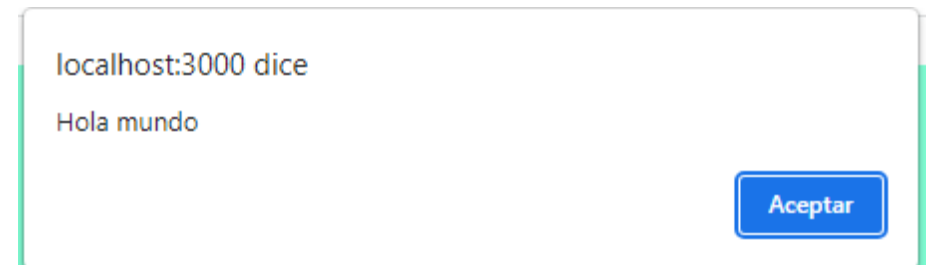
NOTA: Si utilizamos comillas veríamos:

```
<div>John Ruth Hangman</div>
```

FICHERO APP.JS: INSTRUCCIONES JS

Cualquier instrucción JS la tendremos que poner entre llaves para que se ejecute dentro del código JSX. Por ejemplo un **alert** lo pondríamos:

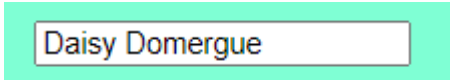
```
function App() {  
  return (  
    <div className="App">  
      {window.alert('Hola mundo')}  
      <h1>Curso de React</h1>  
      <input type='text'></input>  
    </div>  
  );  
}
```



FICHERO APP.JS

Y si necesitamos mostrar un valor dentro de un input

```
function App() {  
  let nombre = "Daisy Domergue";  
  return (  
    <div className="App">  
      <h1>Curso de React</h1>  
      <input type='text' value={nombre}></input>  
    </div>  
  );  
}
```

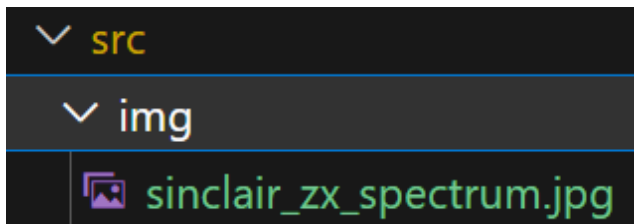


Daisy Domergue

FICHERO APP.JS

¿Cómo añadir una imagen?

Vamos a crearnos una carpeta img dentro de src para guardar una imagen cualquiera



Para poder utilizarla lo primero que tendremos que hacer es importarla en el fichero **app.js**

```
import spectrum from './img/sinclair_zx_spectrum.jpg';
```

NOTA: fijaos que podemos asignarle cualquier nombre a la imagen

E incorporarla utilizando el data binding

```
<img src={spectrum} alt='zx spectrum' />
```

FICHERO APP.JS

Y si queremos añadir estilos css podríamos utilizar un `className` y en nuestro fichero **app.css** añadir los estilos que necesitemos. Ejemplo:

```
<img src={spectrum} alt='zx spectrum' className='imagen'/>
```

```
.imagen {  
  width: 100px;  
}
```

Curso de React

Daisy Domergue

```
<div>John Ruth Hangman</div>
```



FICHERO APP.JS

¿Y podemos trabajar con arrays y objetos?

También podemos trabajar con arrays y objetos como tipos de dato. Ejemplo: creamos un objeto con los datos de una persona (siempre dentro de `function App()`)

```
let persona = {nombre: 'Daisy', apellido: 'Domergue'}
```

Y para visualizar la persona dentro de `<div className="App">`

```
<p>{persona.nombre} {persona.apellido}</p>
```

Daisy Domergue

ACTIVIDAD 1

ACTIVIDAD

Vamos a realizar la siguiente actividad consistente en crear tres fichas con la portada de tres películas y un texto debajo de cada una de ellas con el título de la misma



Kill Bill



Los bingueros



Saw

ACTIVIDAD

Paso 1:

Buscamos tres imágenes con la portada de tres películas y las guardamos en la carpeta **src/img** que hemos creado antes

Paso 2:

Editamos el fichero **src/App.js** para importar las tres imágenes que tenemos en la carpeta anterior. Ejemplo para la primera imagen:

```
import imagen0 from './img/kill-bill.jpg';
```

Vamos a crearnos también un array para guardar los títulos de las películas:

```
let titulos = ['Kill Bill', 'Los bingueros', 'Saw']
```

ACTIVIDAD

Paso 3:

Añadimos el código JSX para crear una caja contenedora para las tres fichas

```
<div className='peliculas'> ... </div>
```

Y añadimos el código JSX para crear una caja para cada ficha.
Ejemplo para la primera ficha

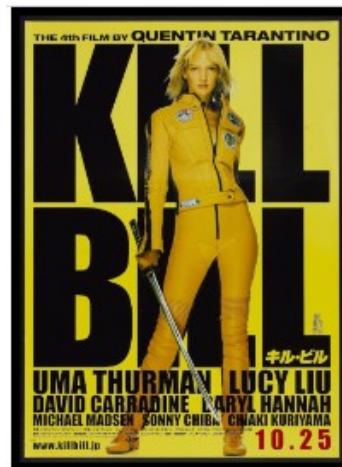
```
<div className='ficha'>  
  <img src={imagen0} alt='Kill Bill'/>  
  <h3>{titulos[0]}</h3>  
</div>
```

ACTIVIDAD

Paso 4:

Y ahora editamos el fichero **src/App.css** para conseguir que las fichas aparezcan una al lado de la otra, centradas y el texto del título también centrado. Utilizaremos el display flex

```
.peliculas {
  display: flex;
  justify-content: center;
  gap: 20px;
}
.ficha {
  width: 200px;
  background-color: lightyellow;
}
.peliculas img {
  width: 100%;
}
.peliculas h3 {
  text-align: center;
}
```



Kill Bill



Los bingueros



Saw

Aviso Legal

Los derechos de propiedad intelectual sobre el presente documento son titularidad de David Alcolea Martinez Administrador, propietario y responsable de www.alcyon-it.com El ejercicio exclusivo de los derechos de reproducción, distribución, comunicación pública y transformación pertenecen a la citada persona.

Queda totalmente prohibida la reproducción total o parcial de las presentes diapositivas fuera del ámbito privado (impresora doméstica, uso individual, sin ánimo de lucro).

La ley que ampara los derechos de autor establece que: “La introducción de una obra en una base de datos o en una página web accesible a través de Internet constituye un acto de comunicación pública y precisa la autorización expresa del autor”. El contenido de esta obra está protegido por la Ley, que establece penas de prisión y/o multa, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujesen, plagiaran, distribuyeren o comunicaren públicamente, en todo o en parte, o su transformación, interpretación o ejecución fijada en cualquier tipo de soporte o comunicada a través de cualquier medio.

El usuario que acceda a este documento no puede copiar, modificar, distribuir, transmitir, reproducir, publicar, ceder, vender los elementos anteriormente mencionados o un extracto de los mismos o crear nuevos productos o servicios derivados de la información que contiene.

Cualquier reproducción, transmisión, adaptación, traducción, modificación, comunicación al público, o cualquier otra explotación de todo o parte del contenido de este documento, efectuada de cualquier forma o por cualquier medio, electrónico, mecánico u otro, están estrictamente prohibidos salvo autorización previa por escrito de David Alcolea. El autor de la presente obra podría autorizar a que se reproduzcan sus contenidos en otro sitio web u otro soporte (libro, revista, e-book, etc.) siempre y cuando se produzcan dos condiciones:

1. Se solicite previamente por escrito mediante email o mediante correo ordinario.
2. En caso de aceptación, no se modifiquen los textos y se cite la fuente con absoluta claridad.

David Alcolea
david-alcolea@alcyon-it.com
www.alcyon-it.com

