ALCYON
IT Servicios

PARTE IV: Componentes

ALCYON
IT Servicios

INTRODUCCIÓN

En este documento veremos:

- Componentes
- Propiedades por defecto
- Uso de variables de useStage del componente padre en los componentes hijos

ALCYON
IT Servicios

INTRODUCCIÓN

Creación del proyecto

Crearemos un nuevo proyecto desde el cmd o el terminal de VSC:

create-react-app componentes

Preparar el entorno

Vamos a editar el fichero **src/App.css** y borraremos el contenido

Editamos el fichero **src/App.js** y vaciamos el contenido del return de la función App()

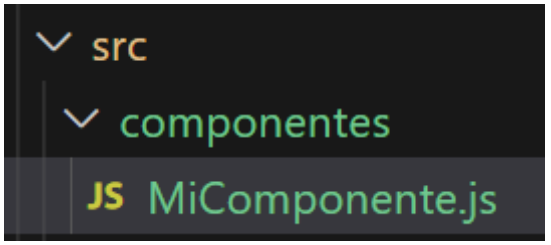
ALCYON
IT Servicios

CREAR COMPONENTES

Hasta ahora hemos trabajado siempre con el componente **App** que es el que se inyecta en el archivo **src/index.js** tal como podemos ver:

```
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Vamos a crear un nuevo componente dentro de una carpeta que llamaremos **componentes** (es buena práctica crear los componentes en una carpeta nueva dentro de la carpeta **src**)



ALCYON
IT Servicios

CREAR COMPONENTES

Editamos el nuevo componente y, si hemos instalado la extensión comentada en el documento anterior, podemos utilizar la extensión rfce para crear el esqueleto del mismo

```
import React from 'react'

function MiComponente() {
  return (
    <div>

    </div>
  )
}

export default MiComponente
```

Para probarlo vamos a añadir dentro de la caja un hola mundo:

<h1>Hola Mundo</h1>

ALCYON
IT Servicios

CREAR COMPONENTES

Ahora vamos a ‘inyectar’ el componente dentro del componente padre que será el **App**:

- Primero tendremos que importarlo:

  import MiComponente from './componentes/MiComponente'

- Y, posteriormente, inyectarlo dentro del return de la función App

```
function App() {
  return (
    <>
      <h1>Este título es del componente padre</h1>
      <MiComponente />
      <MiComponente />
    </>
  )
}
```

Este título es del componente padre

Hola Mundo

Hola Mundo

Fijaos como aparece dos veces el texto de MiComponente ya que lo hemos añadido dos veces en el componente App

ALCYON
IT Servicios

ENVIAR PROPIEDADES

¿Y si queremos enviar datos desde el componente padre App al componente hijo MiComponente para que, en vez de que aparezca dos veces el mismo texto, aparezca info diferente?

Por ejemplo, vamos a enviar a cada uno de los componentes información acerca de dos viejos amigos vintage: el zx spectrum y el oric 1:

```
<>
    <h1>Micros Vintage</h1>
    <MiComponente micro="ZX Spectrum" memoria="48"/>
    <MiComponente micro="Oric 1" memoria="64"/>
</>
```

NOTA 1: Fijaos que pasamos datos al componente en forma de parámetros en la propia etiqueta utilizada para inyectar el componente

NOTA 2: Si tenemos que pasar valores numéricos que tengamos que utilizar posteriormente para realizar alguna operación aritmética es preferible que se pasen de esta forma (en caso contrario serán tratados como texto):

<MiComponente micro="ZX Spectrum" **memoria={48}**/>

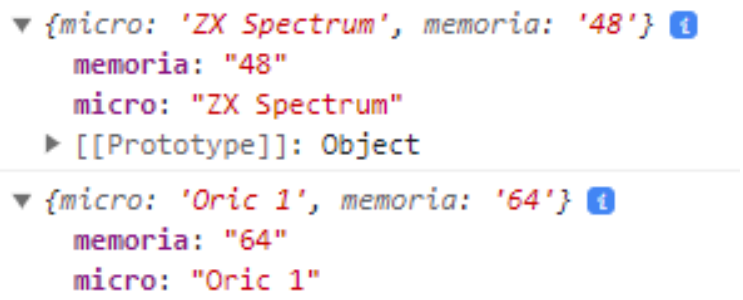
ALCYON
IT Servicios

ENVIAR PROPIEDADES

Ahora vamos editar el componente MiComponente para recoger estos datos:

function MiComponente(**prop**) { **console.log(prop)**; … }

Si miramos en la consola veremos que los datos llegan en forma de objeto:



Que podemos utilizar dentro del componente de esta forma:

<h1>El micro {prop.micro} tiene {prop.memoria} Kb de memoria RAM</h1>

El micro ZX Spectrum tiene 48 Kb de memoria RAM

El micro Oric 1 tiene 64 Kb de memoria RAM

ALCYON
IT Servicios

ENVIAR PROPIEDADES

Otra forma de enviar valores desde el componente padre al hijo es utilizando la desestructuración (utilidad de JS que nos permite convertir los atributos de un objeto en variables):

```
function MiComponente({micro, memoria}) { … }

{/*<h1>El micro {prop.micro} tiene {prop.memoria} Kb de memoria RAM</h1>*/}
<h1>El micro {micro} tiene {memoria} Kb de memoria RAM</h1>
```

Este sistema nos permite, por ejemplo, asignar un valor por defecto a alguna de las dos variables si ésta no llega informada

```
function MiComponente({micro, memoria = 0}) { … }
```

Y en App.js: `<MiComponente micro="Oric 1" />`

El micro Oric 1 tiene 0 Kb de memoria RAM

ALCYON
IT Servicios

ENVIAR PROPIEDADES

Pero tenemos más alternativas para enviar datos desde App al componente MiComponente:

- Podemos enviar los datos en formato Array:

  En App:

```
<MiComponente micro={["ZX Spectrum",48]}/>
<MiComponente micro={["Oric 1"]}/>
```

  En MiComponente:

```
function MiComponente({micro}) { … }

<h1>El micro {micro[0]} tiene {micro[1]} Kb de memoria RAM</h1>
```

El micro ZX Spectrum tiene 48 Kb de memoria RAM

El micro Oric 1 tiene Kb de memoria RAM

ALCYON
IT Servicios

ENVIAR PROPIEDADES

- O también en formato objeto:

  En App:

```
<MiComponente micro={{nombre: "ZX Spectrum", memoria: 48}}/>
<MiComponente micro={{nombre: "Oric 1", memoria: 64}}/>
```

  En MiComponente:

```
function MiComponente({micro}) { … }

<h1>El micro {micro.nombre} tiene {micro.memoria} Kb de memoria RAM</h1>
```

El micro ZX Spectrum tiene 48 Kb de memoria RAM

El micro Oric 1 tiene 64 Kb de memoria RAM

ALCYON
IT Servicios

ACTIVIDAD 4

ACTIVIDAD

Para consolidar lo que hemos visto vamos a realizar una completa actividad que dividiremos en varias partes

Actividad 4.1

A partir de un array de objetos con datos de viejos y queridos micros vintage de los 80, vamos a crear una serie de fichas con la imagen de cada uno de ellos, el nombre del modelo y un precio para los coleccionistas

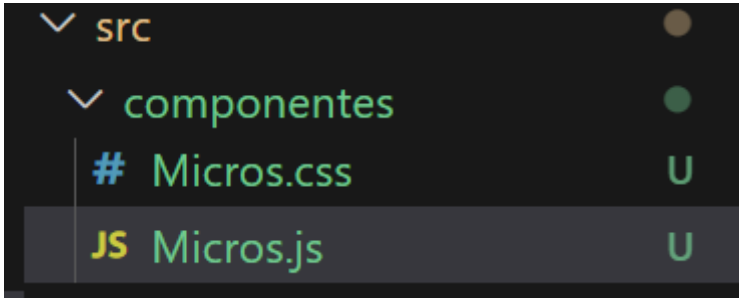
ALCYON
IT Servicios

ACTIVIDAD

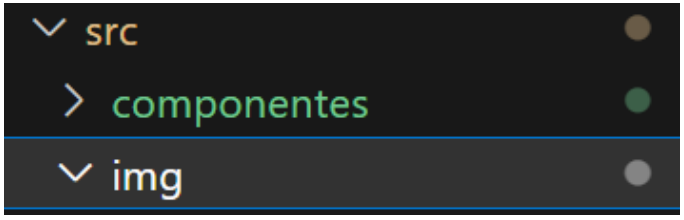
Actividad 4.1

Cada una de las fichas será un componente que llamaremos **Micros** y que crearemos dentro de la carpeta **src/componentes**

Crearemos también un fichero **Micros.css** para este componente:



Crearemos una carpeta **src/img** con todas las imagenes png que utilizaremos en el ejercicio



ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componente principal **App.js**

Vamos a modificar el componente principal para añadir 9 componentes **Micros**:

```
import Micros from './componentes/Micros';

<div className='app'>
        <h1>Micros Vintage</h1>
        <h2>Total: €</h2>
        <div className='fichas'>
            <Micros />
            ...8 componentes más...
        </div>
</div>
```

ACTIVIDAD

Actividad 4.1

Componente principal **App.js**

Añadiremos también el array con los datos de los micros:

```
const micros = [
    {
      micro: 'Sinclair ZX Spectrum',
      imagen: 'sinclair_zx_spectrum.png',
      precio: 250
    },
    {
      micro: 'Sinclair ZX 81',
      imagen: 'Sinclair_ZX81.png',
      precio: 300
    },
    {
      micro: 'Atari 800',
      imagen: 'atari_800.png',
      precio: 250
    },
    {
      micro: 'Commodore 64Kb',
      imagen: 'commodore_64.png',
      precio: 200
    },
    {
      micro: 'Commodore Amiga 500',
      imagen: 'commodore_amiga.png',
      precio: 500
    },
    {
      micro: 'Dragon 32',
      imagen: 'dragon_32.png',
      precio: 220
    },
    {
      micro: 'Tandy New Brain',
      imagen: 'new_brain.png',
      precio: 400
    },
    {
      micro: 'Oric Atmos',
      imagen: 'oric_atmos.png',
      precio: 300
    },
    {
      micro: 'Toshiba MSX HitBit',
      imagen: 'toshiba_msx.png',
      precio: 350
    },
]
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componente principal **App.css**

Y añadiremos un poco de css para que cada uno de los componentes hijos aparezcan unos al lado de otros utilizando display flex:

```
.app {width: 50%; margin:auto;}
.fichas {display: flex; flex-wrap: wrap; justify-content: space-between;}
h1 {text-align: center;}
```

Recordad importar el fichero de css desde **App.js** :

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

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componentes **Micros.js**

Crearemos el código JSX para la ficha donde mostraremos la imagen y los datos de cada micro vintage:

```
<div className='ficha'>
      <img src=” alt=”/>
      <div className='datos'>
        <h4></h4>
        <p >Precio:  €</p>
        <button >Comprar</button>
      </div>
</div>
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componentes **Micros.css**

Y añadiremos el css necesario para que la apariencia visual sea la adecuada

```
.ficha {
    width: 30%;
    border: 2px solid lightskyblue;
    border-radius:10px;
    overflow: hidden;
    background-image: url('../img/fondo_vintage.jpg');
}
.datos {padding: 20px;}
.ficha img {width: 100%;}
```

Recordad importar el fichero de css desde **Micros.js** :

```
import './Micros.css';
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componente principal **App.js**

Volvemos al componente principal y enviaremos los datos de cada micro que tenemos en el array de objetos micros a cada uno de los componentes:

```
<Micros datos={micros[0]}/>
<Micros datos={micros[1]}/>
<Micros datos={micros[2]}/>
… resto de componentes ...
```

NOTA: Fijaos que estamos repitiendo el código del componente 9 veces. En el siguiente documento veremos como utilizar las funciones de JavaScript para iterar sobre arrays de objetos para simplificar mucho más este código

ACTIVIDAD

Actividad 4.1

Componente **Micros.js**

Y volvemos a editar el componente **micros** para recoger los datos que enviamos desde **App** para poder mostrarlos en cada una de las fichas

```
function Micros({datos}) { … }

<div className='ficha'>
    <img src=" alt={datos.micro}/>
    <div className='datos'>
      <h4>{datos.micro}</h4>
      <p>Precio: {datos.precio} €</p>
      <button>Comprar</button>
    </div>
</div>
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.1

Componente **Micros.js**

¿Y como hacemos para mostrar la imagen?

Hemos visto hasta ahora como importar imágenes estáticas, es decir, imágenes que siempre se muestran en el componente principal.

Pero ahora tenemos que mostrar una imagen distinta en cada uno de los 9 componentes **Micros.js** y lo que no podemos hacer es importarlas todas en el componente ya que cada uno de ellos corresponde a una ficha diferente.

ACTIVIDAD

Actividad 4.1

Componente **Micros.js**

¿Y como hacemos para mostrar la imagen?

La solución pasa por utilizar la función JS require():

```
function Micros({datos}) {
  const imagen = require(`../img/${datos.imagen}`);
  return (
    <div className='ficha'>
      <img src={imagen} alt={datos.micro}/>
      .../…
    </div>
  )
}
```

De esta forma cada componente tiene su propia imagen

ACTIVIDAD

Actividad 4.2

Vamos a activar ahora el botón de compra de micros de forma que, vayamos acumulando el importe de cada uno de los micros comprados y que este importe se muestre en el componente principal:

Micros Vintage

Total: 450€

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.2

Componente principal **App.js**

Necesitamos una variable en el componente principal para acumular el total del importe de cada uno de los micros comprados. Esta variable obligatoriamente tiene que estar en este componente ya que recogerá el importe de cada una de las fichas seleccionadas (las fichas sobre las que el usuario pulse el botón de *comprar*).

Para ello necesitamos el hook useState para poder trabajar con variables que se informan desde la vista (el documento JSX)

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

const [total, setTotal] = useState(0); //inicializamos la variable a cero
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.2

Componente principal **App.js**

Incorporamos la variable total en el documento JSX

```
<h2>Total: {total}€</h2>
```

Y, cuando inyectamos los componentes la tenemos que enviar a cada uno de ellos ya que se informará en éstos cuando el usuario pulse el botón de comprar:

```
<Micros datos={micros[0]} total={setTotal}/>
<Micros datos={micros[1]} total={setTotal}/>
<Micros datos={micros[2]} total={setTotal}/>
...
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.2

Componente **Micros.js**

Modificaremos el componente para recuperar el nuevo dato total que enviamos desde el componente padre **App**:

```
function Micros({datos, total}) { … }
```

Asociamos el evento al botón de compra:

```
<button onClick={comprar}>Comprar</button>
```

Y confeccionamos la función comprar:

```
const comprar = () => { ... }
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.2

Componente **Micros.js**

En la función comprar necesitamos sumar el precio del micro seleccionado al total acumulado y esto lo hacemos de la siguiente forma:

```
total((t) => t + datos.precio)
```

Donde:

total	→ corresponde a la variable que recogemos como argumento de la función
t	→ es el contenido de la variable en cada momento (contendrá los precios anteriormente añadidos)
datos.precio	→ es el precio del micro que obtenemos del objeto que recibimos en los argumentos de la función

ACTIVIDAD

Actividad 4.3

Si probamos el ejercicio vemos que efectivamente se va sumando el total con el importe del micro seleccionado y, puesto que la correspondiente ficha continua visible en pantalla, podemos pulsar repetidas veces sobre ella e ir acumulando sucesivas compras del mismo micro

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.3

Vamos a introducir una mejora consistente en eliminar la ficha del micro comprado de forma que no podamos volver a pulsar el botón 'comprar' de esta ficha

Componente **Micros.js**

En la función comprar recogeremos el objeto event, que ya hemos visto en otros ejercicios, para obtener el botón correspondiente a la ficha sobre la que lo hemos pulsado

```
const comprar = (e) => { … }
```

A partir del botón (etiqueta <button>) tenemos que llegar a la caja que corresponde a la ficha que lo contiene (la caja). Para ello utilizaremos los métodos de nodos de JS

```
let ficha = e.target.closest('.ficha')
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 4.3

Componente **Micros.js**

Una vez nos hemos situado en la caja correspondiente a la ficha donde se encuentra el botón, tenemos dos opciones:

1. Eliminar el nodo correspondiente a la ficha desde el elemento padre (al que accedemos con parentNode):

   ficha.parentNode.removeChild(ficha)

2. Ocultamos el nodo correspondiente a la ficha utilizando la propiedad display none

   ficha.style.display = 'none'

Si ahora probamos veremos como se elimina la ficha una vez pulsamos el botón de comprar

Aviso Legal

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