React

DEV.F:.
DESARROLLAMOS(PERSONAS);

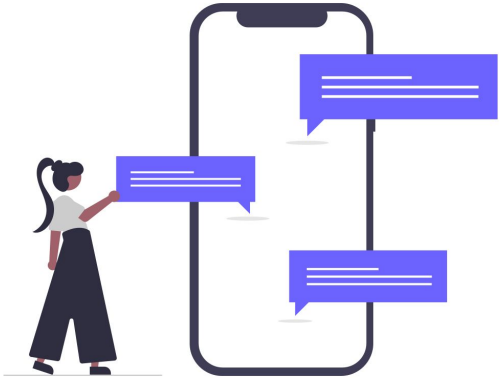
React

- create-react-app y Vite.
- Definición de un componente web.
- Partes de un componente.
- Tipos de componentes.
- Variables, funciones y props.
- Arrays de componentes y condicionales.
- Estilos css.
- Hooks 1 (useState).

DEV.F

React

- Router.
- Levantamiento de estado.
- Consumo de APIs.
- Hooks 2 (useEffect).
- Estilos 2: Frameworks para React (ReactBootstrap, MaterialUI y Tailwind).



Comandos create-react-app

DEV.F:: DESARROLLAMOS(PERSONAS);

CLI de React

Comando	Descripción
npx create-react-app my-project	*Crear un proyecto de react.*
npm run start	*Levantar el proyecto.*

Comandos Vite

DEV.F:: DESARROLLAMOS(PERSONAS);

Vite commands

Comando	Descripción
npm create vite@latest	*Generar un scaffold con la última versión de vite.*
npm build	*Construir el build para desplegarlo.*

Scaffold recomendado

DEV.F::
DESARROLLAMOS(PERSONAS);

Scaffold

Es la estructura de archivos y carpetas de un proyecto, también se le conoce como arquitectura del proyecto.

```
├── src  (todo el codigo fuente)
│   ├── assets (archivos multimedia)
│   │   ├── imgs
│   │   ├── media
│   ├── components (elementos reutilizables)
│   │   ├── Modal
│   │   ├── Table
│   │   ├── Tabs
│   ├── constants (valores que se ocupan en todo el proyecto)
│   ├── pages (las pantallas del sistema, se conforma de varios componentes)
│   ├── hooks (a futuro para los custom hooks)
├   ├── normalize (limpiado de entrada y salida de datos hacia la API)
├   ├── service (clientes de servicios agrupados por entidad)
├   ├── utils (funciones utileria que se repiten)
├   ├── App.css (estilos del App)
├   ├── App.jsx
├   ├── index.css (estilos de branding / genericos)
├   ├── main.jsx
├── dist
├── node_modules
├── package.json
├── package-lock.json
└── .gitignore
└── index.html
└── index.css
```

Partes de un componente

DEV.F:
DESARROLLAMOS(PERSONAS);

Partes de un componente

1. Importación de React (opcional si no usa React.Fragment).
2. Una función nombrada en PascalCase.
3. Exportar esa función.
4. La función debe tener un método return.

```jsx
import React from "react";

function Header() {
    const text = 'Pokemon News';
    return <h1>{text}</h1>;

}

export { Header };
```

```jsx
import React from "react";

function Card() {

  const nameComponent = 'patito';

  return ( /** JSX */
    <React.Fragment>
      <h1>{nameComponent} component works!</h1>
      <span>aqui iria el html de la tarjetita</span>
    </React.Fragment>
  )
}

export { Card };
```

Tipos de componentes

- **Componentes Funcionales:** Son funciones de JavaScript que devuelven elementos de React. Antes de la introducción de los hooks en React, estos componentes eran conocidos como "componentes sin estado".
- **Componentes de Clase:** Estos son componentes definidos como clases de JavaScript que extienden la clase React.Component o React.PureComponent. Antes de la llegada de los hooks, los componentes de clase eran la forma principal de crear componentes en React y podían manejar el estado interno mediante el uso del método setState() y tenian un ciclo de vida.

Tipos de componentes

- **Higher-Order Components (HOC):** Son componentes que toman otro componente como argumento y devuelven un nuevo componente. Se utilizan comúnmente para reutilizar la lógica entre componentes.
- **Pages:** Agrupan un conjunto de componentes para representar una página de un sitio web.
- **Componentes Presentacionales y Contenedores:** Esta es más una distinción de patrón que de tipo. Los componentes presentacionales se centran en la visualización y normalmente no tienen lógica de estado o lógica de negocios.

Características de un componente de react

DEV.F:
DESARROLLAMOS(PERSONAS);

Variables, funciones y props

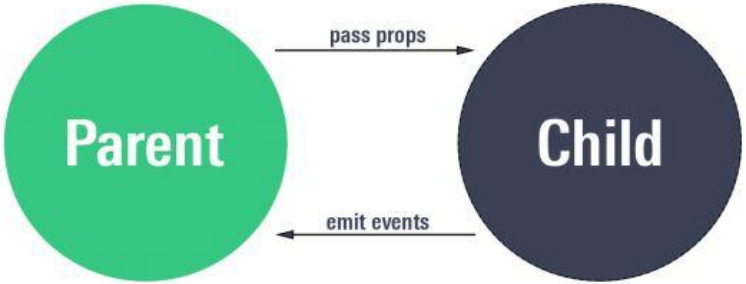
```
function Header() {

    const text = 'Pokemon News';

    const foo = () => {
        // doSomething
    }

    foo();

    return <h1>{text}</h1>;

}

export { Header };
```

```
import React from "react";

function Header() {

    const text = 'Pokemon News';

    const foo = () => {
        // doSomething
    }

    foo();

    return (
        <React.Fragment>
            <h1>{text}</h1>
            <p>Contenido del parrafo</p>
        </React.Fragment>
    );

}

export { Header };
```

Las funciones y propiedades de los componentes se definen fuera del método return.

¿Qué son los props?

- Son **como** los **atributos** de HTML pero para componentes.
- Las props son **entradas de datos** para los componentes y **son de solo lectura (inmutables).**
- Se utilizan para **pasar información de padres a hijos pero no al revés.**
- Hay una **prop especial** llamada **children** que sirve para pasar el contenido que pongamos dentro de la custom tag.



Estilos

DEV.F::
DESARROLLAMOS(PERSONAS);

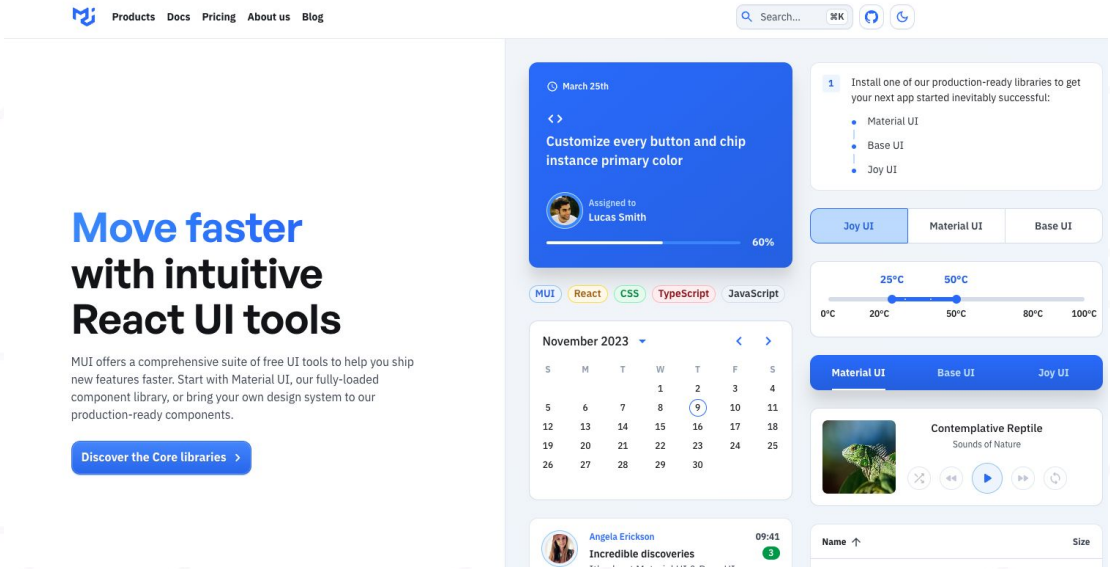
Estilos

Existen diferente formas de estilar en React:

1. **Estilos de línea (como objeto y con atributos en mayúsculas).**
2. **Importación de hoja de estilos externa y uso de className.**
3. CSS Modular.
4. CSS en el JS.
5. Preprocesadores.
6. **FrameworksUI**
    - (MaterialUI, ReactBootstrap, Tailwind).

Material UI

Es un framework enfocado a estilar componentes de React, ofreciendo un catálogo de componentes personalizables.



Arrays y condicionales

DEV.F:
DESARROLLAMOS(PERSONAS);

Arrays

- **map:** Para renderizar una lista de componentes se utiliza la función map. y
- **key:** Se debe utilizar en cada componente que se renderiza con el map para que react los pueda diferenciar.

```
{
        pokemons.map(element => (
          <Card key={element.id} name={element.name} img={element.img} />
        ))
      }
```

Condiciones

- **If:** Permite condicionar el devolver los métodos return y no el JSX interno.
- **Operador ternario (?):** Utilizado dentro del JSX para condicionar qué elementos se mostrarán si se cumple la condición y cuáles en contrario.
    - Es muy similar a if, else.
- **Logical AND Operator (&&):** Utilizado dentro del JSX para condicionar si se muestra algo o no.
    - Es muy similar a un if

NOTA: Considere que también puede encerrar su código JSX en funciones y mandarlas a llamar para tener un método return más limpio.

Hooks

DEV.F:.
DESARROLLAMOS(PERSONAS);

¿Qué es un Hook?

Son **funcionalidades extra que podemos enganchar a nuestros componentes** funcionales. Anteriormente para usar esas funcionalidades forzosamente usábamos la sintaxis de clases.

DEV.F

Hooks útiles

Surge como una solución a la necesidad del manejo de estado de los componentes funcionales.

- useState.
- useEffect.
- useContext.

useEffect y useState

DEV.F:
DESARROLLAMOS(PERSONAS);

useEffect y useState

- **useState** ofrece una propiedad get y un setter para la actualización de cualquier variable que lo requiera.
- **useEffect** se ejecuta cada vez que se se actualiza el render. Es una función que recibe un callback y una condición de ejecución.
  - Si la condición es **,** lo que esté dentro del callback se ejecuta en cada renderizado (mount, didMount).
  - Si la condición es **, []** lo que esté dentro del callback se ejecuta solo en el primer render (didMount).
  - Si la condición es **, [total]** => lo que esté dentro del callback se ejecuta la primera vez y cada que cambie algún valor dentro de las dependencias.

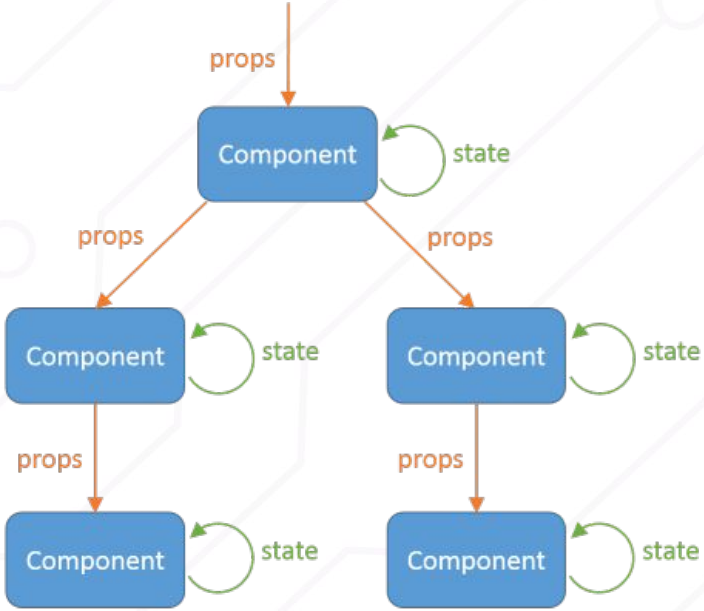
NOTA: useEffect usa Object.is para las comparaciones.

State

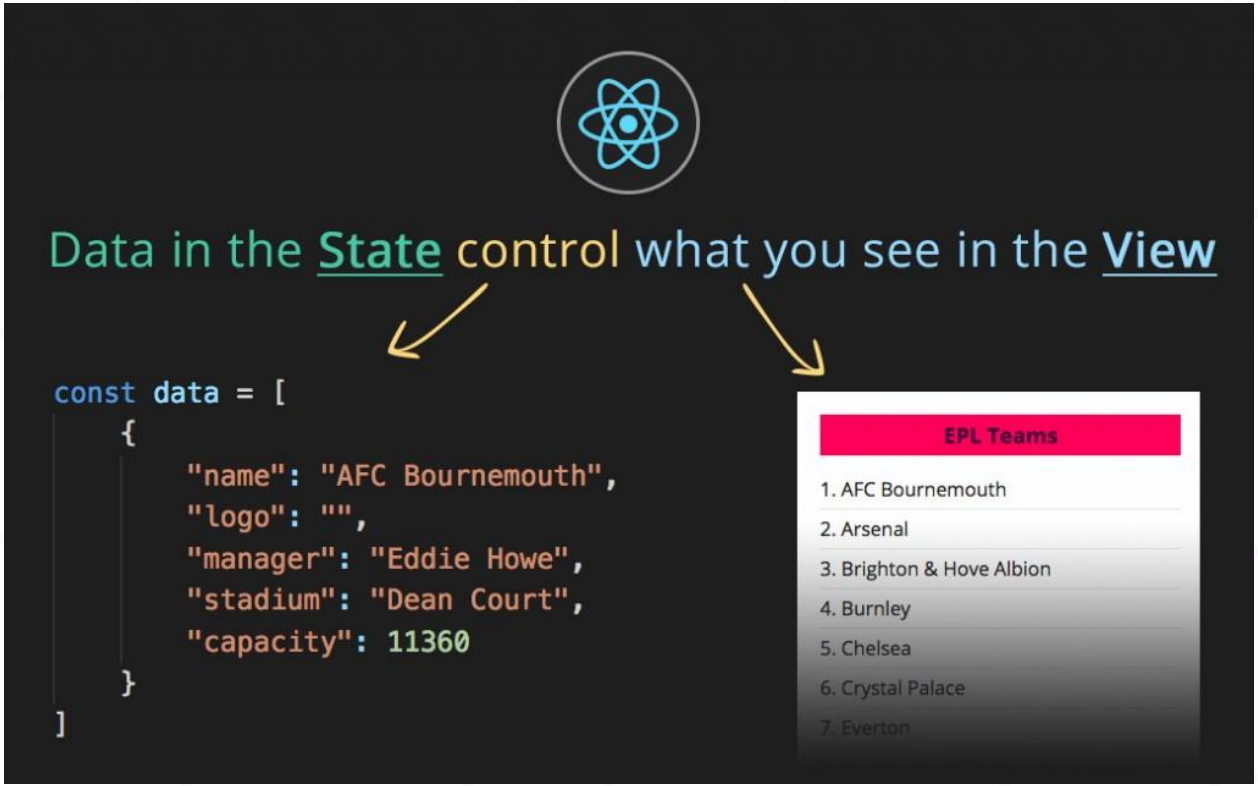
DEV.F
DESARROLLAMOS(PERSONAS);

State

El estado es el **medio que utiliza react para guardar los valores en tiempo de ejecución**. A diferencia de las props, el estado **es actualizable**.



State


Data in the State control what you see in the View
const data = [
{
"name": "AFC Bournemouth",
"logo": "",
"manager": "Eddie Howe",
"stadium": "Dean Court",
"capacity": 11360
}
]
EPL Teams
1. AFC Bournemouth
2. Arsenal
3. Brighton & Hove Albion
4. Burnley
5. Chelsea
6. Crystal Palace
7. Everton

Formularios

DEV.F::
DESARROLLAMOS(PERSONAS);

Manejo de formularios

Eventos

DEV.F::
DESARROLLAMOS(PERSONAS);

Listado de eventos

- El nombrado en camelCase.
- Considere la WEB_API y su clases Event (e, e.target, etc.) y formData.
- Listado de eventos:
  - Html.
  - JS.
  - React.

Router

DEV.F::
DESARROLLAMOS(PERSONAS);

Router

Crearlo por medio de la librería react router. Considere que se puede implementar como componentes o con useRoutes.

Artículo de ayuda.

Levantamiento del estado

DEV.F::
DESARROLLAMOS(PERSONAS);

Levantamiento de estado

El levantamiento de estado es una técnica de React que **pone el estado en una localización donde se pueda pasar como props a los componentes**.

Lo ideal es **poner el estado en el lugar más cercano a todos los componentes que quieren compartir esa información**, así todos nuestros componentes tendrán el mismo estado y cuando este cambie sólo re-renderizará lo necesario.

Estado compartido

Consiste en **pasar las funciones setter como props desde padres a través de los componentes hijos que requieren actualizarlo**. Con ello se brinda la posibilidad de modificar valores desde otros componentes.

Consumo de API’s

DEV.F::
DESARROLLAMOS(PERSONAS);

Consumo de API’s

En react podemos consumir apis mediante cualquier librería, por ejemplo axios o bien de forma nativa com fetch.