React

DEV.F:.
DESARROLLAMOS(PERSONAS);

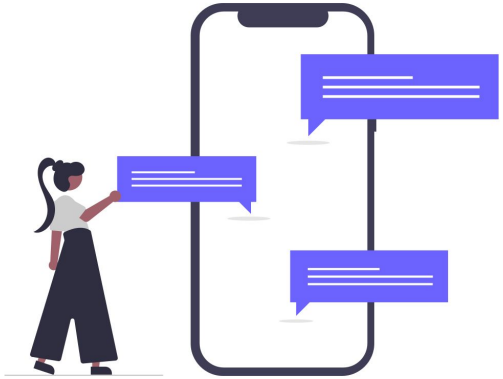
React

- Webpack, create-react-app, Vite.
- Definición de un componente web.
- Partes de un componente.
- Variables, funciones y props.
- Arrays de componentes y condicionales.
- Estilos y Material UI.
  Hooks (useState y useEffect).

DEV.F

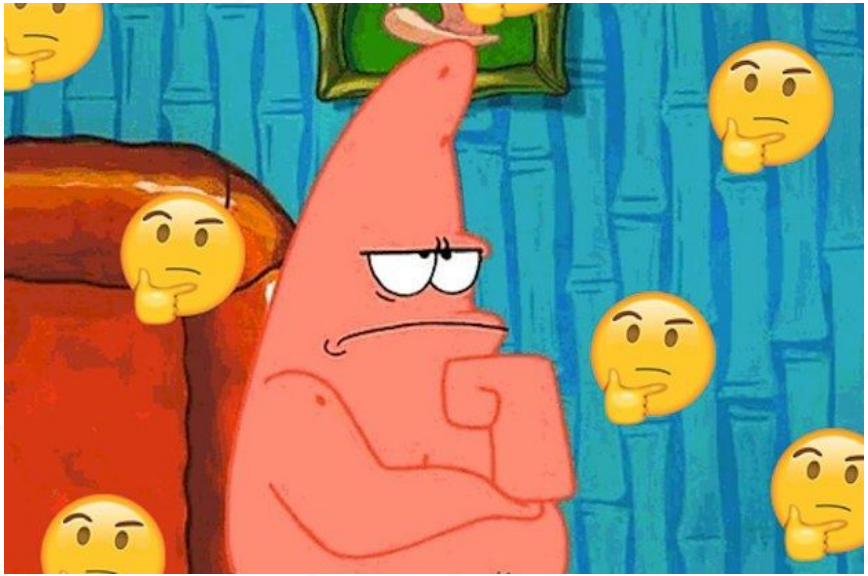
React

- State.
- Formularios y eventos.
- Router.
- Levantamiento de estado.
- Consumo de APIs.
- Context API (createContext y useContext).
- Consumo de APIs.
- Portals.
- Scaffold recomendado.
- Tipos de componentes.



¿Y cómo vamos a hacerlo?

- Teoría.
- Ejercicio práctico que resuelve problemas del mundo real.
- Investigando y resolviendo errores.



React app, vite y webpack

DEV.F::
DESARROLLAMOS(PERSONAS);

React app, vite y webpack

create-react-app

vite

webpack

Conceptos clave

DEV.F::
DESARROLLAMOS(PERSONAS);

Conceptos clave

- **Web component:** Es un segmento de nuestra página que podemos abstraer y separar en una funcionalidad específica.
- **Custom Tag:** Es el nombre con el que mandamos llamar a nuestros componentes para que se rendericen. Ejemplo: <Card/>.
- **JSX:** Es una sintaxis especial de JS para escribir React de una forma más intuitiva para el desarrollador.
- **React.fragment:** Es una etiqueta especial que no ocupa espacio en el DOM y que nos resuelve el problema de qué return tenga un solo elemento. Se puede abreviar con **<></>**.

Conceptos clave

- **Atributo** (html) vs **props** (react)

- Imports ES6 (export y export default).
- **Virtual DOM:** Es un punto medio entre JSX y HTML para que react sepa que renderizar en la pantalla.

Web Componente

DEV.F::
DESARROLLAMOS(PERSONAS);

Web Component

- Es un técnica y estándar moderno para el desarrollo web.
- Consiste en abstraer una funcionalidad (que normalmente se repite) en uno o varios scripts, esto nos permite utilizarlo en varias partes de nuestro sitio.
- Es similar a maquetar y dividir en cajar pero además de le asigna funcionalidad y distintos comportamientos.
- Se basa en 3 conceptos:
  - **Custom elements:** Crear etiquetas personalizadas.
  - **Shadow DOM:** Copia del DOM principal para encapsular funcionalidad..
  - **HTML Templates:** Escribir plantillas que combinan HTML, CSS y JS. Renderizar contenido pero tambien son dinámicas y cambian de acuerdo a su lógica.

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 CamelCase.
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 };
```

Variables, funciones y props

DEV.F:: DESARROLLAMOS(PERSONAS);

Variables, funciones y props

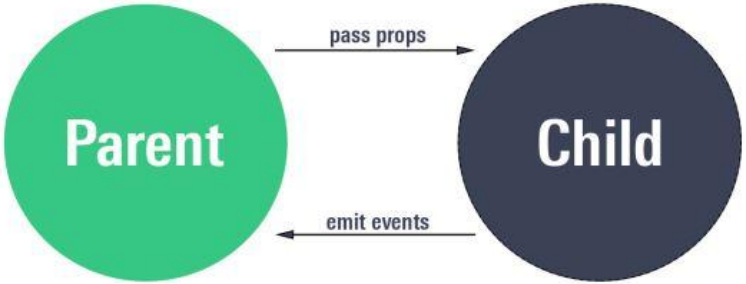
```
function Header() {

    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.

Características de las 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.



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.

Estilos

DEV.F::
DESARROLLAMOS(PERSONAS);

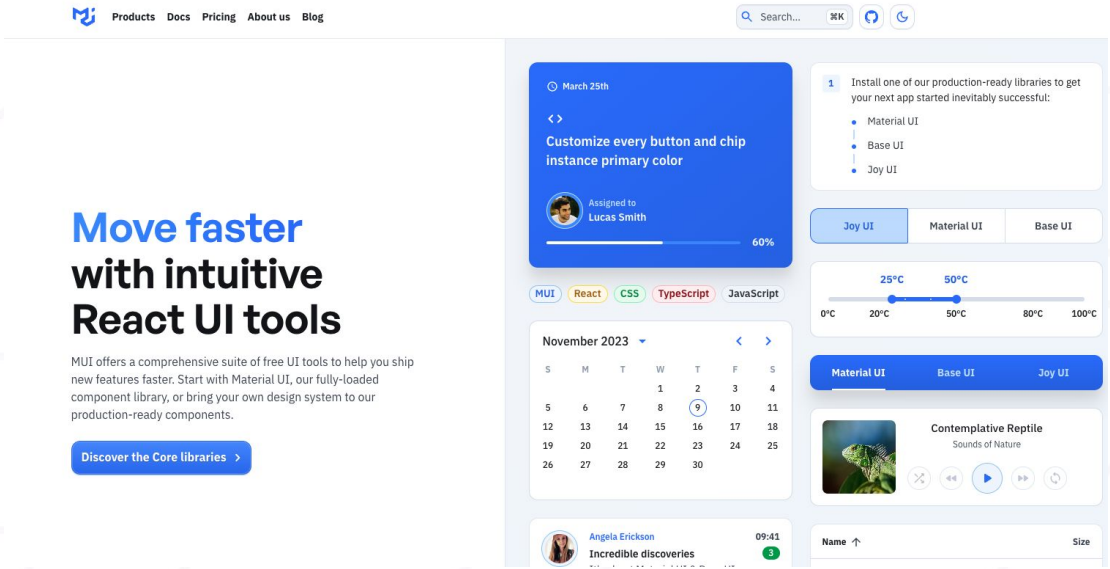
Estilos

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.



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.

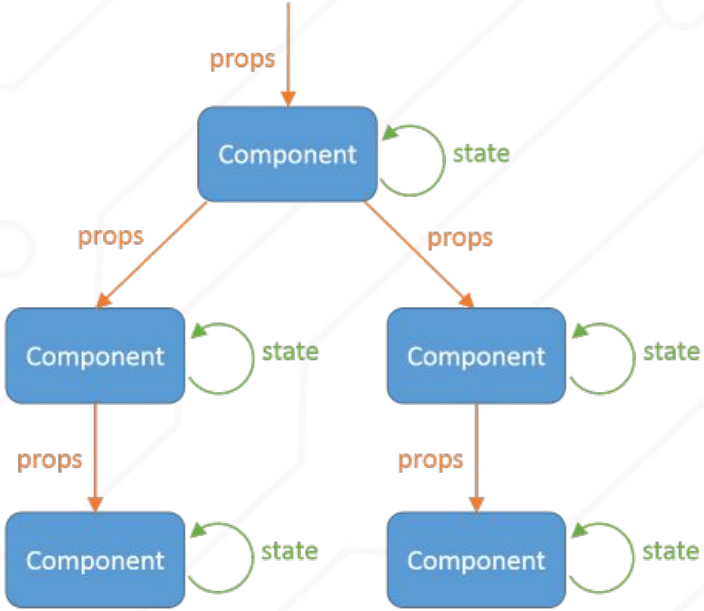
- **useEffect** se ejecuta cada vez que se se actualiza el render. Se puede condicionar:
  - , => Como condición se actualiza hasta el infinito.
  - ,[] => Solo se actualiza la primera vez.
  - ,[total] => se ejecuta cuando cambia la variable de estado total.

State

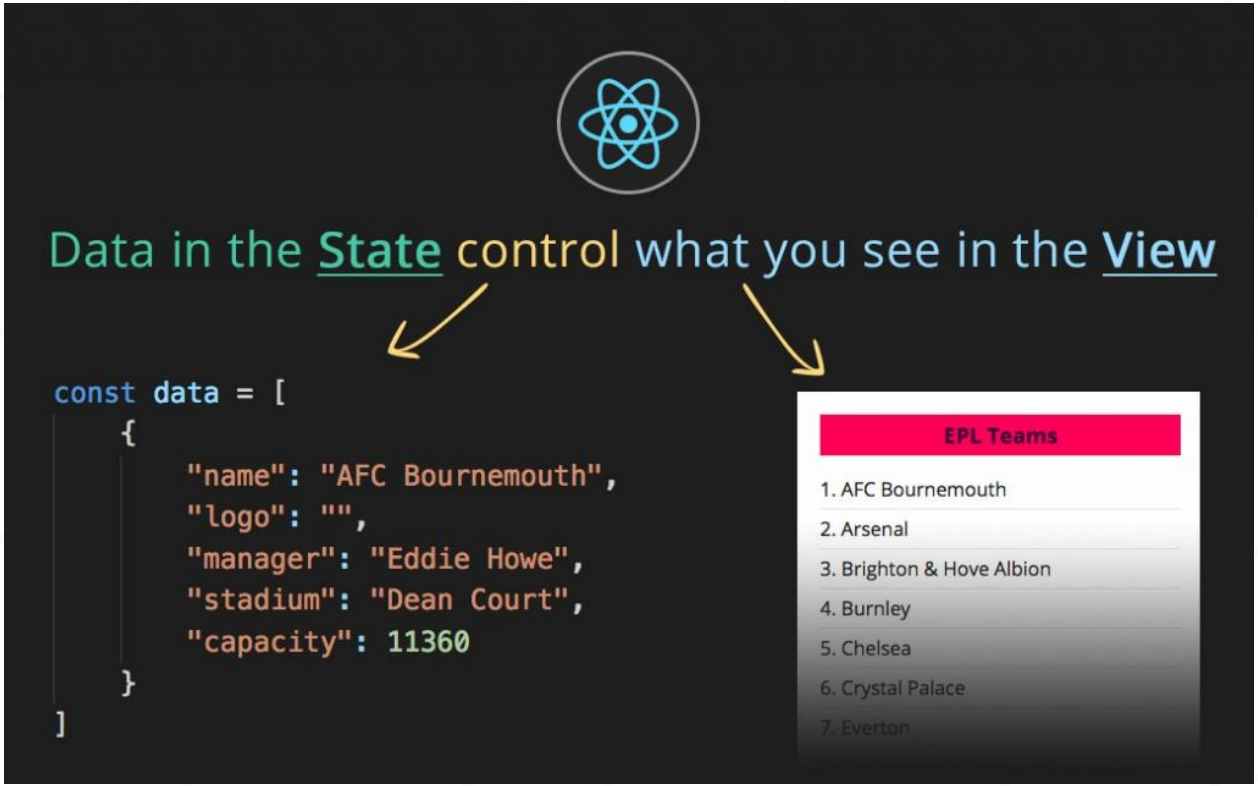
DEV.F
DESARROLLAMOS(PERSONAS);

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.

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.

Context API

DEV.F:
DESARROLLAMOS(PERSONAS);

Context API

Es un mecanismo para compartir datos entre componentes. Se maneja como un **espacio de información** (**context**) que es compartido a los hijos que lo requieran (deben estar envueltos por un **provider**). Cada componente obtiene los datos a través de useContext (cumpliendo la función de **consumers**).

Es una técnica que **simplifica el proceso de compartir props que deben pasar entre varios componentes** o incluso a través de todo el aplicativo.

Se recomienda usar Context API moderadamente y con buenas prácticas de diseño.

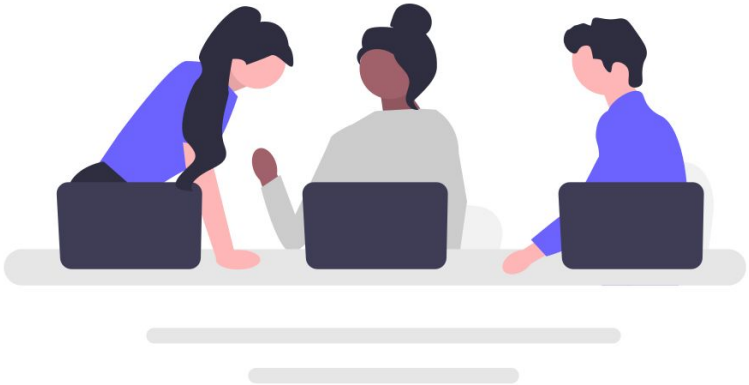
Ejemplo de uso.

Best Practices.

Casos de uso útiles

Use API Context en casos como:

- Jerarquía de componentes acotados que requieren una comunicación continua.
- Theming.
- User authentication.
- Multilenguaje.
- Datos prove



Consumo de API’s

DEV.F::
DESARROLLAMOS(PERSONAS);

Consumo de API’s

Por medio de la librería axios.

Portals

DEV.F::
DESARROLLAMOS(PERSONAS);

Portals

Nos permiten **renderizar elementos hijo en cualquier parte del DOM.**

- Funcionan mediante la función createPortal de ReactDOM.

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
```

Clasificación de componentes

DEV.F::
DESARROLLAMOS(PERSONAS);

Clasificación de componentes

- De clase.
- Funcionales.
- Statefull.
- Stateless.
- Containers.
- Pages.
- Utils.