# **Para crear una REACT-APP desde la terminal:**

npx create-react-app NombreDeLaApp

**JavaScript:**

*const* nombre = 'sebi';

*const* apellido = 'Epstein';

//const nombreCompleto = nombre + ' ' + apellido;

*const* nombreCompleto = ` ${nombre} ${apellido}`;

**Clonar un objeto:**

*const* persona = {

    nombre: 'tony',

    apellido: 'Stark',

    edad: 45,

    direccion: {

        ciudad: 'tucuman',

        calle: '25 de mayo',

        numero: 710,

    },

};

*const* persona2 = { ...persona }; // Clono el objeto de persona en persona2

persona2.nombre = 'Pedrito' // Cambio el nombre de persona2 SOLAMENTE

console.log( persona );

console.log( persona2 );

**Ese es el operador SPREAD**

**Arreglo en JS:**

//const arreglo = new Array();

*const* arreglo = [1,2,3,4];

//arreglo.push(1);

*let* arreglo2 = [...arreglo, 5];

console.log( `este es el arreglo 1: ${arreglo}` );

console.log( `este es el arreglo 1: ${arreglo2}` );

**Función MAP()**

*const* arreglo2 = [1,2,3,4];

*const* arreglo3 = arreglo2.map(*function*(*numero*){

 return numero \* 2;

});

**Funciones**

// FUNCION CON NORMAL

*const* saludar = *function* (*nombre*) {

    return `Hola ${ nombre }`;

}

// FUNCION FLECHA

*const* saludar2 = (*nombre*) *=>* {

    return `Hola ${ nombre }`;

}

// FUNCION FLECHA LIGERA

*const* saludar3 = (*nombre*) *=>* `Hola ${ nombre }`;

// FUNCION FLECHA CON RETORNO DE OBJETO

*const* getUser = () *=>* ({

       uid: 'ABC12345',

        username: 'usuario1',

    });

console.log( saludar("sebi"));

console.log( saludar2("sebi"));

console.log( saludar3("sebi"));

console.log( getUser());

**Desestructuración**

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

};

*const* { nombre, edad, clave } = persona;

console.log( nombre );

console.log( edad );

console.log( clave );

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

    rango: 'soldado',

};

*const* retornaPersona = ({ *nombre*, *edad*, *rango* = 'capitan' }) *=>* {

    console.log(edad, nombre, rango);

}

retornaPersona( persona );

*const* persona = {

    nombre: 'tony',

    edad: 45,

    clave: 'ironman',

    rango: 'soldado',

};

*const* Prueba = ({ *clave*, *nombre*, *edad*, *rango* = 'capitan' }) *=>* {

    //console.log(edad, nombre, rango);

    return {

        nombreClave: clave,

        anios: edad,

        latIng:{

            lat: 14.2552,

            lng: -12.4425

        }

    }

}

*const* {nombreClave, anios, latIng:{lat, lng}} = Prueba( persona );

console.log(nombreClave, anios);

console.log(lat, lng);

# **Filter Y Find**

import { heroes } from './data/heroes';

*const* getHeroeById = (*id*) *=>* {

    return heroes.find( (*heroes*) *=>* id === heroes.id);

}

//console.log( getHeroeById(4) );

*const* getHeroeByOwner = ( *owner* ) *=>* heroes.filter( (*heroes*) *=>* owner === heroes.owner);

console.log( getHeroeByOwner('DC') );

# **Promesa:**

*const* promesa = new *Promise*( (*resolve*, *reject*) *=>* {

    setTimeout(() *=>* {

*const* p1 = getHeroeById(2);

       // console.log(heroe);

        //getHeroeByOwner();

        resolve( p1 );

    }, 2000);

});

Promesa con Id del héroe

*const* getHeroeByIdAsync = ( *id* ) *=>* {

    return new *Promise*( (*resolve*, *reject*) *=>* {

        setTimeout(() *=>* {

*const* p1 = getHeroeById(22);

           // console.log(heroe);

            //getHeroeByOwner();

            if( p1 ){

                resolve( p1 );

            } else {

                reject('No se puede encontrar el heroe');

            }

        }, 2000);

    });

}

getHeroeByIdAsync(4)

    .then( *heroe* *=>* console.log('Heroe', heroe))

    .catch( *err* *=>* console.warn( err ))

**Forma más fácil de then y catch**

getHeroeByIdAsync(4)

    .then( console.log /\*heroe => console.log('Heroe', heroe)\*/)

    .catch( console.warn /\*err => console.warn( err )\*/);

# **Fetch**

*const* apiKey = 'xOEnsoKrVNzdRsi94IkBuB9PabY4Sv0S'

*const* peticion = fetch(`https://api.giphy.com/v1/gifs/random?api\_key=${ apiKey }`);

peticion

.then( *resp* *=>* resp.json() )

.then( ({*data*} ) *=>* {

*const* { url } = data.images.original;

*const* img = document.createElement('img');

    img.src= url;

    document.body.append( img );

})

.catch( console.warn);

# **Async Await**

*const* getImagen = async () *=>* {

  try {

*const* apiKey = "xOEnsoKrVNzdRsi94IkBuB9PabY4Sv0S";

*const* resp = await fetch(

      `https://api.giphy.com/v1/gifs/random?api\_key=${apiKey}`

    );

*const* { data } = await resp.json();

*const* { url } = data.images.original;

*const* img = document.createElement("img");

    img.src = url;

    document.body.append(img);

  } catch (error) {

      // MANEJO DEL ERRROR

      console.error( error )

  }

};

getImagen();

# **Operación Ternaria:**

*const* mensaje = (activo) ? 'Activo' : 'Inactivo'; // ESPERA EL IF Y ELSE

*const* mensaje = activo && 'Activo' ; //SOLO ESPERA ENTRAR AL IF

**REACT**

# **Para crear una REACT-APP desde la terminal:**

npx create-react-app NombreDeLaApp

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentenode\_modules**

* Tiene las librerías y los paquetes que hace que la aplicación REACT funcione
* No se la modifican manualmente.
* Para modificar es con comando para instalar algo.

**Texto

Descripción generada automáticamentePublic**

* Favicon.ico: el icono de REACT
* Manifest.json: se ven dónde están los logos. NADA que ver con REACT
* Robots.txt: configuración por defecto, no es importante. NADA que ver con REACT

**.gitignore**

* Es un archivo que se maneja por la parte de git

**package.json**

* Casi nunca se toca.

**README.md**

* Es un archivo que describe la aplicación.

Texto

Descripción generada automáticamente con confianza baja**src**

* Es en donde mas vamos a pasar el tiempo.
* .test.js donde se realizan las pruebas

# **Hola mundo en REACT**

import React from 'react'

import ReactDOM from 'react-dom'

*const* saludo = <h1>Hola mundo</h1>;

*const* divRoot = document.querySelector('#app');

ReactDOM.render( saludo, divRoot )

console.log(divRoot);

# **Renderizar algo en REACT**

**Index.js**

import React from 'react'

import ReactDOM from 'react-dom/client'

import PrimeraApp from './PrimeraApp';

import './index.css'

*const* divRoot = ReactDOM.createRoot(document.getElementById("app"));

divRoot.render(<*PrimeraApp* />);

**PrimeraApp.js**

import React from 'react'

// componente en una funcion FC

*const* PrimeraApp = () *=>* {

    return <h1>Hola mundo</h1>;

};

export default PrimeraApp;

# **Renderizar mas de una cosa:**

**Opción A: Fragment**

import React, { Fragment } from 'react'

// componente en una funcion FC

*const* PrimeraApp = () *=>* {

    return (

        <*Fragment*>

            <h1>Hola mundo</h1>

            <p> Primer Parrafo</p>

        </*Fragment*>

    );

};

export default PrimeraApp;

**Opción B:**

import React from 'react'

*const* PrimeraApp = () *=>* {

    return (

        <>

            <h1>Hola mundo</h1>

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Impresión de variables en el HTML**

*const* PrimeraApp = () *=>* {

*const* saludo = 'Hola mundo'

*const* persona = {

        nombre: 'sebi',

        apellido: 'Epstein',

    }

    return (

        <>

            <h1> { saludo } </h1>

            <pre> { JSON.stringify(persona, null, 3) } </pre>

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Comunicación entre componentes y Props**

**Index.js:**

import React from 'react'

import ReactDOM from 'react-dom/client'

import PrimeraApp from './PrimeraApp';

import './index.css'

*const* divRoot = ReactDOM.createRoot(document.getElementById("app"));

divRoot.render(<*PrimeraApp* saludo="hola soy sebi" />);

**PrimeraApp.js:**

//import React, { Fragment } from 'react'

// componente en una funcion FC

import React from 'react'

*const* PrimeraApp = ( {*saludo* = 'hola perro'} ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            {/\*<pre> { JSON.stringify(persona, null, 3) } </pre>\*/}

            <p> Primer Parrafo</p>

        </>

    );

};

export default PrimeraApp;

# **Validacion de Props**

import React from 'react';

import PropTypes from 'prop-types';

// IMPORTO PropTypes PARA PODER USARLO

*const* PrimeraApp = ( { *saludo* } ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            <p> Primer Parrafo</p>

        </>

    );

};

PrimeraApp.propTypes = {

    saludo: PropTypes.string.isRequired

    // Obligo a que saludo sea obligatorio y string

}

# **Props por defecto:**

**Opción A:**

*const* PrimeraApp = ( { *saludo*, *subtitulo* = 'Opcion A' } ) *=>* {

    return (

        <>

            <h1> { saludo } </h1>

            <p> { subtitulo }</p>

        </>

    );

};

**Opción B:**

PrimeraApp.defaultProps = {

 subtitulo: 'Soy un subtitulo',

}

# **Rafcp + tabulador**

import React from 'react'

import PropTypes from 'prop-types'

*const* CounterApp = *props* *=>* {

  return (

    <div>CounterApp</div>

  )

}

CounterApp.propTypes = {}

export default CounterApp

# **Evento del mousse: onClick**

import React from 'react'

import PropTypes from 'prop-types'

    // HandleAdd

*const* handleAdd = (*e*) *=>* {

        console.log(e)

    }

*const* CounterApp = ( { *value* } ) *=>* {

    return (

      <>

        <h1>CounterApp</h1>

        <h2> {value} </h2>

        <button onClick={ handleAdd }>+1</button>

      </>

    )

};

# **UseState y Hook**

import React, { useState } from 'react'

import PropTypes from 'prop-types'

*const* CounterApp = ( { *value* } ) *=>* {

*const* [ counter, setCounter ] = useState(0);

    // HandleAdd

*const* handleAdd = (*e*) *=>* {

        setCounter( counter + 1);

        setCounter( (*c*) *=>* c + 1 );

    }

    return (

      <>

        <h1>CounterApp</h1>

        <h2> { counter } </h2>

        <button onClick={ handleAdd }>+1</button>

      </>

    )

};

# **Pruebas**

import '@testing-library/jest-dom';

import { getSaludo } from "../../base/02-template-string";

describe('Pruebas en 02-template-string.test.js', () *=>* {

    test('getSaludo debe de retornar hola sebi', () *=>* {

*const* nombre = 'sebi';

*const* saludo = getSaludo(nombre);

        expect( saludo ).toBe( 'Hola ' + nombre);

     })

});

**HOOKS**

Todos los hooks empiezan con use

useState

### ¿Qué hace setState?

setState() programa una actualización al objeto estado de un componente. Cuando el estado cambia, el componente responde volviendo a renderiza

import React, { useState } from "react";

import "./counter.css";

export *const* CounterApp = () *=>* {

*const* [state, setState] = useState({

    counter1: 10,

    counter2: 20,

    counter3: 30,

    counter4: 40,

  });

*const* { counter1, counter2 } = state;

  return (

    <>

      <h1>

        Counter { counter1 }

      </h1>

      <h1>

        Counter { counter2 }

      </h1>

      <hr />

      <button

        className="btn btn-primary"

        onClick={() *=>* {

          setState({

              ...state,

              counter1: counter1 + 1,

             });

        }}

      >

        +1

      </button>

    </>

  );

};

useCounter – Personalizado

**useCounter.js**

import { useState } from "react"

export *const* useCounter = (*initialState* = 10) *=>* {

*const* [state, setState] = useState(initialState);

*const* increment = ( *factor* = 1) *=>* {

        setState( state + factor);

    }

*const* decrement = ( *factor* = 1) *=>*{

        setState( state - factor);

    }

*const* reset = () *=>* {

        setState( initialState );

    }

    return {

        state,

        increment,

        decrement,

        reset

    };

}

**CounterWithCustonHook.js**

import React from 'react';

import { useCounter } from '../../hooks/useCounter';

import './counter.css';

export *const* CounterWithCustonHook = () *=>* {

*const* {state, increment, decrement, reset } = useCounter( 10000 );

  return (

    <>

        <h1> Counter With Hook: {state} </h1>

        <hr/>

        <button onClick={ () *=>* increment(2) } className='btn btn-primary'> +1 </button>

        <button onClick={ reset } className='btn btn-primary'> Reset </button>

        <button onClick={ () *=>* decrement(1) } className='btn btn-primary'> -1 </button>

    </>

  )

}

useEffect y sus precauciones

import React, { useEffect, useState } from "react";

import "./effects.css";

export *const* SimpleForm = () *=>* {

*const* [formState, setFormState] = useState({

    name: "",

    email: "",

  });

*const* { name, email } = formState;

  useEffect(() *=>* {

    console.log("holaaa");

  }, []);

  useEffect(() *=>* {

    console.log("el formstate cambio");

  }, [formState]);

  useEffect(() *=>* {

    console.log("el email cambio");

  }, [email]);

*const* handleInputChange = ({ *target* }) *=>* {

    setFormState({

      ...formState,

      [target.name]: target.value,

    });

  };

  return (

    <>

      <h1>useEffect</h1>

      <hr />

      <div className="form-group">

        <input

          type="text"

          name="name"

          className="form-control"

          placeholder="Tu nombre"

          autoComplete="off"

          value={name}

          onChange={handleInputChange}

        />

      </div>

      <div className="form-group">

        <input

          type="text"

          name="email"

          className="form-control"

          placeholder="email@gmail.com"

          autoComplete="off"

          value={ email }

          onChange={handleInputChange}

        />

      </div>

    </>

  );

};

UseEffect cancelando el addEventListener

import React, { useEffect, useState } from "react";

export *const* Message = () *=>* {

*const* [coords, setCoords] = useState({ x: 0, y: 0 });

*const* { x, y } = coords;

  useEffect(() *=>* {

*const* mouseMove = (*e*) *=>* {

*const* coors = { x: e.x, y: e.y };

      setCoords(coors);

    };

    window.addEventListener("mousemove", mouseMove);

    return () *=>* {

      window.removeEventListener("mousemove", mouseMove);

    };

  }, []);

  return (

    <>

      <h3>Eres genial</h3>

      <p>

        x: {x}

        y: {y}

      </p>

    </>

  );

};

UseForm Para un formulario

**./hooks/useForm.js**

import { useState } from 'react'

export *const* useForm = ( *initialState*= {} ) *=>* {

*const* [values, setValues] = useState(initialState);

*const* handleInputChange = ({ *target* }) *=>* {

        setValues({

            ...values,

            [ target.name ]: target.value

        });

    }

    return [values, handleInputChange ];

useRef

import { useEffect, useRef, useState } from 'react'

export *const* useFetch = ( *url* ) *=>* {

*const* isMounted = useRef(true);

*const* [state, setState] = useState({

        data: null,

        loading: true,

        error: null

    });

    useEffect(() *=>* {

        return () *=>* {

            isMounted.current = false;

        }

    }, [])

    useEffect(() *=>* {

        setState({

            data:null,

            loading: true,

            error: null,

        })

        fetch(url)

        .then( *resp* *=>* resp.json() )

        .then( *data* *=>* {

            if( isMounted.current){

                setState({

                    loading: false,

                    error: null,

                    data

                })

            }

        })

    }, [url]);

    return state;

}

useFetch - Personalizado + optimizaciones

**MultipleCustomHooks.js**

import React from "react";

import { useCounter } from "../../hooks/useCounter";

import { useFetch } from "../../hooks/useFetch";

import "../02-useEffect/effects.css";

export *const* MultipleCustomHooks = () *=>* {

*const* { counter, increment } = useCounter(1);

*const* { loading, data } = useFetch(

    `https://www.breakingbadapi.com/api/quotes/${counter}`

  );

*const* { author, quote } = !!data && data[0];

  console.log(author, quote);

  return (

    <>

      <h1>BreakingBad Quotes</h1>

      <hr />

      {loading ? (

        <div className="alert alert-info text-center">Loading...</div>

      ) : (

        <blockquote className="blockquote text-right">

          <p className="mb-0"> {quote}</p>

          <footer className="blockquote-footer"> {author} </footer>

        </blockquote>

      )}

      <button className="btn btn-primary" onClick={increment}>

        Siguiente Quote

      </button>

    </>

  );

};

**useFetch.js**

import { useEffect, useState } from 'react'

export *const* useFetch = ( *url* ) *=>* {

*const* [state, setState] = useState({

        data: null,

        loading: true,

        error: null

    });

    useEffect(() *=>* {

        setState({

            data:null,

            loading: true,

            error: null,

        })

        fetch(url)

        .then( *resp* *=>* resp.json() )

        .then( *data* *=>* {

            setState({

                loading: false,

                error: null,

                data

            })

        })

    }, [url]);

    return state;

}

useLayoutEffect

import React, { useLayoutEffect, useRef, useState } from "react";

import { useCounter } from "../../hooks/useCounter";

import { useFetch } from "../../hooks/useFetch";

import './layout.css'

export *const* Layout = () *=>* {

*const* { counter, increment } = useCounter(1);

*const* { data } = useFetch(

    `https://www.breakingbadapi.com/api/quotes/${counter}`

  );

*const* { quote } = !!data && data[0];

*const* pTag = useRef();

*const* [boxSize, setBoxSize] = useState({});

  useLayoutEffect( () *=>* {

      setBoxSize( pTag.current.getBoundingClientRect() );

  }, [quote])

  return (

    <>

      <h1>LAYOUT EFFECT</h1>

      <hr />

      <blockquote className="blockquote text-right">

        <p

         className="mb-0"

         ref={pTag}

         > {quote}</p>

      </blockquote>

      <pre>

    {

        JSON.stringify(boxSize, null, 3)

    }

      </pre>

      <button className="btn btn-primary" onClick={increment}>

        Siguiente Quote

      </button>

    </>

  );

};

Memo

**Memorize.js**

import React, { useState } from "react";

import { useCounter } from "../../hooks/useCounter";

import { Small } from "./Small";

import "../02-useEffect/effects.css";

export *const* Memorize = () *=>* {

*const* { counter, increment } = useCounter(10);

*const* [show, setShow] = useState(true);

  return (

    <>

      <h1>

        Counter: <*Small* value={ counter }/>

      </h1>

      <hr />

      <button className="btn btn-primary" onClick={increment}>

        +1

      </button>

      <button

      className="btn btn-outline-primary ml-3"

      onClick={ () *=>* {

          setShow( !show )

      }}

      >

          Show/Hide { JSON.stringify(show) }

      </button>

    </>

  );

};

**Small.js**

import React from "react";

export *const* Small = React.memo( ({ *value* }) *=>* {

  console.log("Me Volvi a llamar");

  return (

    <>

      <small>{value}</small>

    </>

  );

});

useMemo

**procesoPesado.js**

export *const* procesoPesado = (*iteraciones*) *=>* {

    for( *let* i = 0; i < iteraciones; i++){

        console.log('ahi vamos');

    }

    return `${iteraciones} Iteraciones Realizadas`;

}

**MemoHook.js**

import React, { useMemo, useState } from "react";

import { useCounter } from "../../hooks/useCounter";

import { procesoPesado } from "../../helpers/procesoPesado"

import "../02-useEffect/effects.css";

export *const* MemoHook = () *=>* {

*const* { counter, increment } = useCounter(100);

*const* [show, setShow] = useState(true);

*const* memoProesoPesado = useMemo(() *=>* procesoPesado(counter), [ counter ])

  return (

    <>

      <h1>MemoHook</h1>

      <h3>

        Counter: <small> { counter } </small>

      </h3>

      <hr />

    <p> { memoProesoPesado} </p>

      <button className="btn btn-primary" onClick={increment}>

        +1

      </button>

      <button

      className="btn btn-outline-primary ml-3"

      onClick={ () *=>* {

          setShow( !show )

      }}

      >

          Show/Hide { JSON.stringify(show) }

      </button>

    </>

  );

};

useCallback

**CallBackHooks.js**

import React, { useCallback, useEffect, useState } from "react";

import "../02-useEffect/effects.css";

import { ShowIncrement } from "./ShowIncrement";

export *const* CallBackHooks = () *=>* {

*const* [counter, setCounter] = useState(10);

//   const increment = () => {

//     setCounter(counter + 1);

//   };

*const* increment = useCallback( (*num*) *=>* {

    setCounter( *c* *=>* c + num);

  }, [ setCounter ])

  return (

    <>

      <h1> UseCallBack Hooks: {counter} </h1>

      <hr />

    <*ShowIncrement* increment={ increment }/>

    </>

  );

};

**ShowIncrement.js**

import React from 'react'

export *const* ShowIncrement = React.memo(( {*increment*} ) *=>* {

    console.log('Me volvi a generar :( ')

  return (

    <button

    className='btn btn-primary'

    onClick={ () *=>* {

        increment( 5 );

    } }

    >

        Incrementar

    </button>

  )

}

)

useReducer

**todoReducer.js**

export *const* todoReducer = ( *state* = [], *action* ) *=>* {

    switch (action.type) {

        case 'add':

            return [...state, action.payload]

        case 'delete':

            return state.filter( *todo* *=>* todo.id !== action.payload );

        case 'toggle':

            return state.map( *todo* *=>*

                ( todo.id === action.payload)

                    ? { ...todo, done: !todo.done}

                    : todo

                )

            case 'toggle-old':

            return state.map( *todo* *=>* {

                if(todo.id === action.payload){

                    return {

                        ...todo,

                        done: !todo.done

                    }

                } else {

                    return todo;

                }

            })

        default:

            return state;

    }

}

**TodoApp.js**

import React, { useEffect, useReducer } from "react";

import { todoReducer } from "./todoReducer";

import "./styles.css";

import { TodoList } from "./TodoList";

*const* init = () *=>* {

  return JSON.parse(localStorage.getItem("todos")) || [];

};

export *const* TodoApp = () *=>* {

*const* [todos, dispatch] = useReducer(todoReducer, [], init);

  useEffect(() *=>* {

    localStorage.setItem("todos", JSON.stringify(todos));

  }, [todos]);

*const* handleDelete = (*todoId*) *=>* {

*const* action = {

      type: "delete",

      payload: todoId,

    };

    dispatch(action);

  };

*const* handleToggle = (*todoId*) *=>* {

*const* action = {

      type: "toggle",

      payload: todoId,

    };

    dispatch(action);

  };

*const* handleAddTodo = (*newTodo*) *=>* {

    dispatch({

      type: 'add',

      payload: newTodo

    })

  }

  return (

    <>

      <h1> TodoApp ( {todos.length} ) </h1>

      <hr />

      <div className="row">

        <div className="col-7">

          <*TodoList*

            todos={todos}

            handleDelete={handleDelete}

            handleToggle={handleToggle}

          />

        </div>

        <div className="col-5">

         <*TodoApp*

         handleAddTodo={ handleAddTodo }

         />

        </div>

      </div>

    </>

  );

};

Reducers

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza baja

Context

Provider

useContext

React Router

Links y NavLinks

CreateContext

SPA ( Single Page Application )