REACT

11. What is the difference between class components and functional components?

-The main difference between class components and functional components is in the way they are defined and handled. Class components are defined using the JavaScript class syntax, while functional components are defined using JavaScript functions.

12. How would you set the state of a component?

-I would set it using the setState() function.
One way would be to pass an object that contains the values you want to update in the state, and another way would be to pass a function that takes the current state as an argument and returns an object that contains the values you want. to update on the status.

13. What happens when we change the state of a component?

-This is a process called rendering where React re-evaluates the component's code and determines if there are any changes to its state or properties.
If there are changes to the component's state, React updates the DOM (Document Object Model) to reflect these changes and displays the new version of the component in the UI.

14. Write a component that renders a list of elements, given a prop that contains an array of objects in the following form:
{ firstname: "demo", lastname: "demo", dni: 1234 }

```
 import React from "react";
 function ListaDeElementos(props)

 const elementos = props.items.map(item => (
   <li key={item.dni}>
    {item.firstname} {item.lastname} ({item.dni})
   </li>
 ));

 return (
   <div>
    <h2>Lista de elementos:</h2>
    <ul>{elementos}</ul>
   </div>
 );
}
export default ListaDeElementos;
```

15. Write a component that receives a prop called "data", and renders an H1 with the content of data.

```
import React from "react";

function Component (props) {
 return <h1>{props.datos}</h1>;
}

export default Component;
```

16. What is the error in the following component?

```
export const Button = ({text}) => {
    <div className = "button">
        <h1 className="text-button">{text}</h1>
    </div>;
};
```

-The error of this component is that the function is not returning anything, there must be a return.

17. Explain, in your own words, what this component does.

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

export const Component = () => {
  const [state, setState] = useState(1);

  const click = (propertyValue: number) => {
    setState(propertyValue+1);
  };

  return (
    <>
      <h1>{state}</h1>
      <button onClick={() => click(state)}>Click here</button>
    </>
  );
}
```

-What this component does is initialize the internal state with 1 at the beginning. Defines a click function that takes "propertyValue" as an argument and increments the state by 1 when calling the "setState" function.
The component renders an h1 element that displays the current value of the internal state, and a button that, when clicked, calls the "click" function with the current value of the internal state.

18. How would you make a Rest API call from JavaScript?

One way could be to use the Fetch API.
The “fetch(url, options)” method is called passing the REST API URL and a fetch(url, options) object as arguments. In the options object, you must specify the HTTP method using the "method" property and the headers using the "headers" property.
If data needs to be sent with the request, specify it in the "body" property.
Use the “then()” method to handle the response from the REST API when it is received.

19. What is the use of the useEffect hook?

It is used to run side effects after rendering a component.
A side effect is anything that is not related to the rendering of the component.

20. How would you develop a custom hook that exposes methods to save and modify a counter? The hook must have a parameter that allows the developer to specify how much the counter should add for each call to the setter.

Create the custom hook function. In this function, define the initial state of the counter using the “`useState”` hook, and create two functions to increment and decrement the counter value. The “`useCallback”` function can be used to memoize the functions and prevent them from being recreated on every component render.

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

function useCounter(initialValue, incrementAmount) {
  const [count, setCount] = useState(initialValue);

  const increment = useCallback(() => {
    setCount(count => count + incrementAmount);
  }, [incrementAmount]);

  const decrement = useCallback(() => {
    setCount(count => count - incrementAmount);
  }, [incrementAmount]);

  return [count, increment, decrement];
}
```

The “useCounter” function accepts two arguments: the initial value of the counter and the amount that will be added to each call to the “increment” function or subtracted from each call to the “decrement” function.

The “useCounter” function returns an array with three elements: the current value of the counter, the “increment” function, and the “decrement” function.