How Do You Reference Values Using Refs?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-understanding-effects-and-referencing-values-in-react/how-do-you-reference-values-using-refs

In React, there may be situations where you need direct access to a DOM element. That's where "refs" come in handy. Refs can also store mutable values, but state is a better choice for that.

In vanilla JavaScript, you used the `getElementById()` and `querySelector()` methods to access DOM elements. But in React, you use refs to access elements in the DOM.

One of the main differences is that, with refs, you don't need identifiers like IDs and classes to reference elements.

So, how can you create and use refs? React provides a `useRef()` hook that lets you do just that.

The first step is to import the hook from React:

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

Next, you need to create a variable that holds the ref with the initial value of the ref inside the `useRef` hook, say a `sectionRef` initialized to `null`:

```
const sectionRef = useRef(null);
```

The final thing to do is to attach the ref variable to the element in your JSX by using the `ref` attribute:

```
<section ref={sectionRef}>
  {/* Section content */}
</section>
```

If you log the ref to the console, you'll see it's an object with the current value, in this case, `null`:

```
console.log(sectionRef); // { current: null }
```

You can also log the current value to the console with the `current` property so you can see the value directly:

```
console.log(sectionRef.current); // null
```

The subsequent values of the ref depend on the component lifecycle.

For example, the initial value of `sectionRef` will always be `null` because that's what it was initialized to. After the component is mounted, the value of the ref will be the `section` element the ref is attached to.

If the component is unmounted, the ref's value goes back to the initial value of `null`.

A typical example to showcase a ref is to focus an input element on render, or by clicking a button.

Here's how to do that when you click a button:

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

const Focus = () => {
  const inputRef = useRef(null);

  const handleFocus = () => {
    if (inputRef.current) {
      inputRef.current.focus();
    }
  };

  return (
    <div>
      <input ref={inputRef} type="text" placeholder="Enter text" />
      <button onClick={handleFocus}>Focus Input</button>
    </div>
  );
};

export default Focus;
```

In the code above, the `inputRef` is created and attached to the `input` element. There's also a button with an `onClick` event that calls a `handleFocus` function.

All the `handleFocus` function does is call the `focus()` method on the `input` element. Note that, because `input` is a built-in component that comes with React, the actual `input` DOM element is set to the `current` property of the ref. So you call the `focus()` method with `input.current.focus()`.

Here are some best practices you should be aware of while working with refs:

- Use refs mainly to interact with the DOM. You can also use them for mutable data, but state is a better choice for that.
- Don't use refs for basic state management – that is what `useState` is for.
- Make sure you check that `ref.current` exists before accessing its properties. Here's how to do that again:

```
const handleFocus = () => {
  if (inputRef.current) {
    inputRef.current.focus();
  }
};
```

This prevents errors in case the ref is accessed before it is attached to the DOM or after it is removed.

Questions

How can you create and use refs in React?

By importing and using the `useState` hook.

By importing and using the `useEffect` hook.

By importing and using the `useRef` hook.

By importing and using the `useMemo` hook.

What does a ref look like if you log it to the console?

An object with a `current` property.

A primitive value like a string or number.

A function that tracks component state.

A list of all DOM nodes in the component.

What is the value of the ref after the component is mounted?

Always `null`.

The props of the component.

A random value generated by React.

The DOM element the ref is attached to.