10 - CardNav

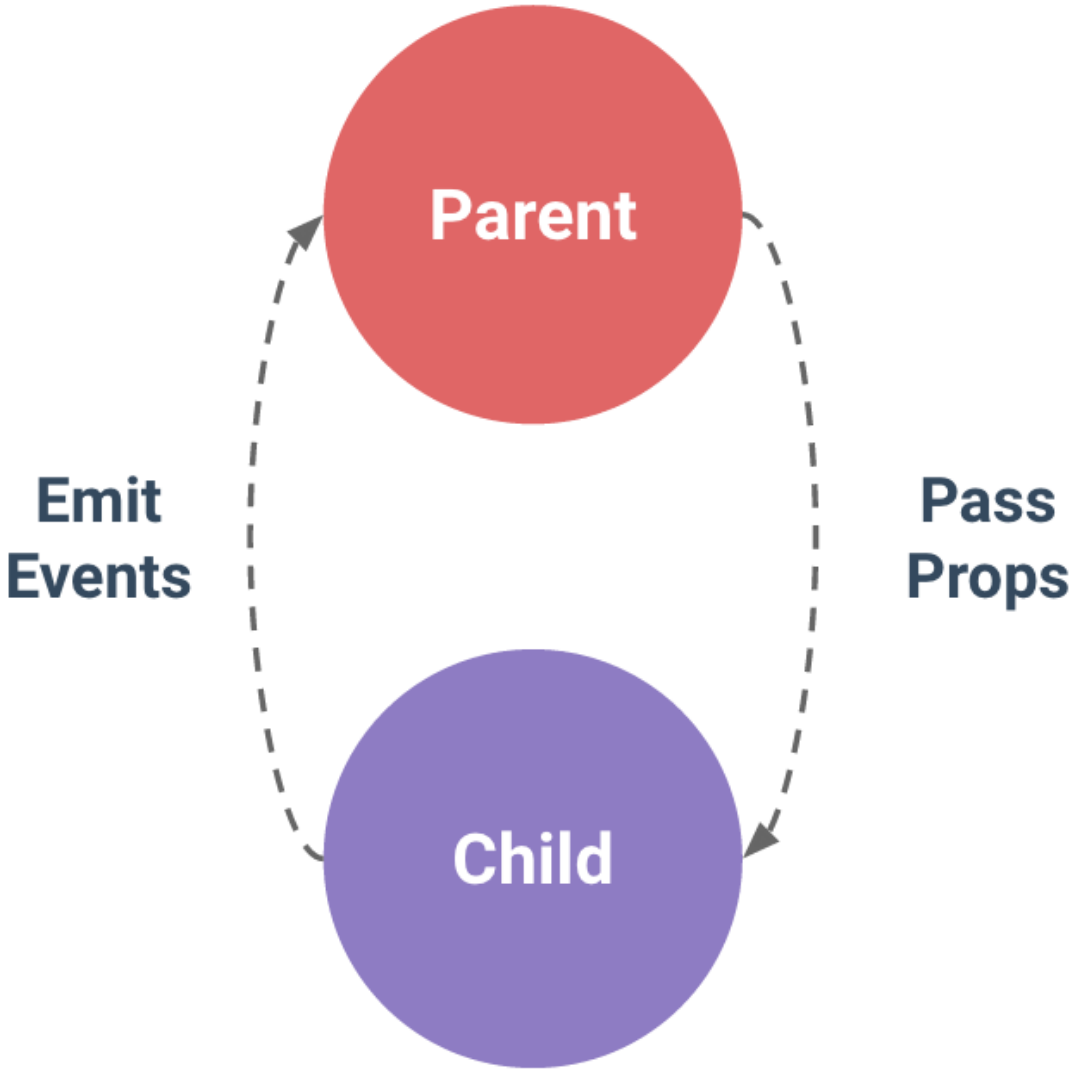
Handling Events

Handling events with React elements is very similar to handling events on DOM elements. There are some syntactic differences:

- React events are named using camelCase, rather than lowercase.
- With JSX you pass a function as the event handler, rather than a string.

Some events:

- onKeyDown
- onKeyPress
- onKeyUp
- onFocus
- onBlur
- onClick
- onChange
- onSubmit
- And more ...



In out app:

Create our cardNav component display first, that will render the arrows to change presentations. Notice the onClick attribute, onClick is an event, therefore should contain a function as event handler. CardNav

```
  const CardNav = ({next, handler}) => {
  return (
    <button style={navStyle(next)} onClick={handler(next)}>
      {next ? '>' : '<'}
    </button>
  );
 }

// Buttons styles
const navStyle = next => {
  const style = {
    position: 'fixed',
    top: '40%',
    outline: 'none',
    fontWeight: 'bold',
    borderRadius: '50%',
    fontSize: '25px'
  }

  // Change place depending on argument
  if (next)
    Object.assign(style, {
      right: '10px'
    })
  else
    Object.assign(style, {
      left: '10px'
    })

  return (style);
}
```

Add the cardNav components to our Info Display, one button to the next and other to the previous item. Info Display

```
const Info = ({ article, handler }) => {
return (
  <section className="box--center">
    <Card article={article} />
    <CardNav next handler={handler} />
    <CardNav prev handler={handler} />
  </section>
  );
}
```

Finally, add the bind to the function on the constructor, since it will be linked to the DOM event.

And create the handler function itself. Every funtion linked to an event will take an event as parameter, in this case we will also pass a boolean to diferentiate which button we are clicking. Info Container

```
class InfoContainer extends Component {

  // Bind the function to the DOM
  constructor() {
    this.handleClick = this.handleClick.bind(this);
  }

  // Add Click Event handler
  handleClick = (event, next) => {
    event.preventDefault();
    let index = articles.indexOf(this.findArticle());
    const len = articles.length - 1;
    if (next)
      index = index === len ? 0 : index + 1;
    else
      index = index === 0 ? len : index - 1;
    const id = articles[index] ? articles[index].id : articles[0].id;

    // Router history push
    this.props.history.push(id)
  }
}
```

Programmatically navigate with React Router

You may need to change the router dynamically, in this example when we click the button. React router dom lets do this in different ways, but it is usually done with this.props.history.push(param) It will change the parameter's value of the dynamic route where the router is located.

Read More:

- Programatically navigate
- React Events
- React Event Handling

<< return