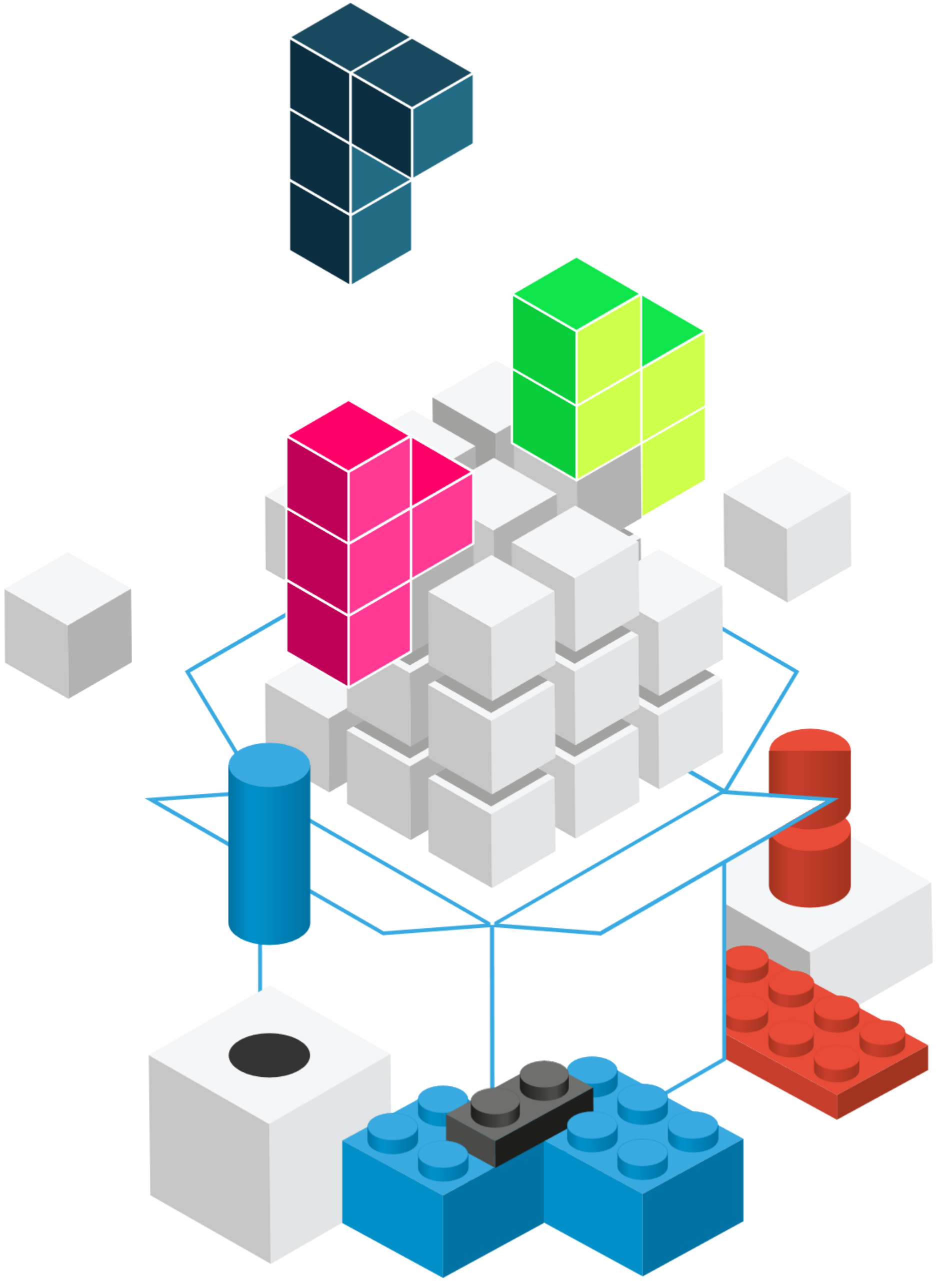
<WA1/>
<AW1/>
2021

# Components and State

## The Foundations of React

Fulvio Corno
Luigi De Russis
Enrico Masala



# Outline

- React Components
  - Props and State
- React design process
  - Top-down information flow

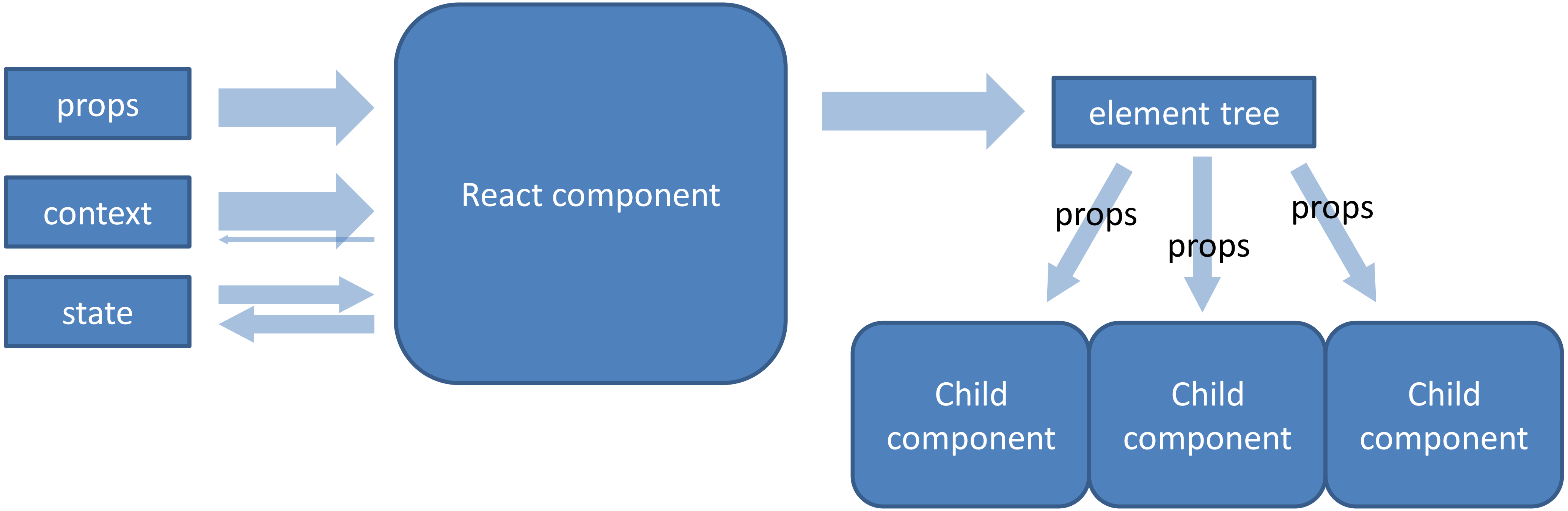
https://reactjs.org/docs/react-component.html

Full Stack React, Chapter “Advanced Component Configuration with props, state, and children”

React Handbook, Chapters “Props” and “State”

# COMPONENTS: PROPS AND STATE

# Props, State, Context



# Props, State, Context

- **Props** are immutable pieces of data that are passed into child components from parents
- **State** is where a component holds data, locally
  - When state changes, usually the component needs to be re-rendered
  - State is *private* to the component and is *mutable* from inside the component, only
- **Context** is a sort of “global” and “implicit” props, that are automatically passed to all interested components

# Passing Props

- In JSX, every attribute is converted to a prop
  - `<Header headerText='Hello'/>`
  - `props.headerText` will contain the string `"hello"`
- `props` is the argument of the Component Function and collects all passed props
  - They are all read-only
- May be any JS object, or other React elements
  - `<UserError level={3}/>`
  - `<ResultsTable displayData={latestResults}/>`

# State

- An object containing local data, private to a component, that may be mutated by the component itself
- To define a state variable, use the `useState` hook

# State Example

```
function WelcomeButton(props) {
  let [english, setEnglish] =
    useState(true) ;

  return (<button>
      {english ? 'Hello' : 'Ciao'}
    </button>) ;
}
```

- Call `useState` with the *initial version* of an object describing the component state
- Inside the component, you may refer the state variable to customize the result according to the current state
- Never n-e-v-e-r modify the state variable directly
  - Always use the set*Variable* function

# State Modifications

- **All** modifications to the state must be requested through set*Variable*(*newValue*)
- It will apply the modification asynchronously (not immediately)

- New Value as a function
  - `(old_state) => updated_state`
  - Must return a **new** state value
  - Must **not** mutate the passed-in state

# Calling State Changes

- State changes are usually determined by asynchronous events
  - DOM event handlers
  - Server responses (e.g., API calls)
- The event handler is a function that in turn calls set*Variable*

```
function WelcomeButton(props) {
  let [english, setEnglish] =
    useState(true) ;

    const toggleLanguage = () => {
      setEnglish( e => !e ) ;
    }

  return (<button onClick={toggleLanguage}>
      {english ? 'Hello' : 'Ciao'}
    </button>);
}
```

# Calling State Changes

- State changes are usually determined by asynchronous events
  - DOM event handlers
  - Server responses (e.g., API calls)
- The event handler is a function that in turn calls set*Variable*
  - Often implemented as an arrow function

```
function WelcomeButton(props) {
  let [english, setEnglish] =
    useState(true) ;

  return (<button
    onClick={()=>setEnglish((eng)=>(!eng))}>
      {english ? 'Hello' : 'Ciao'}
    </button>);
}
```

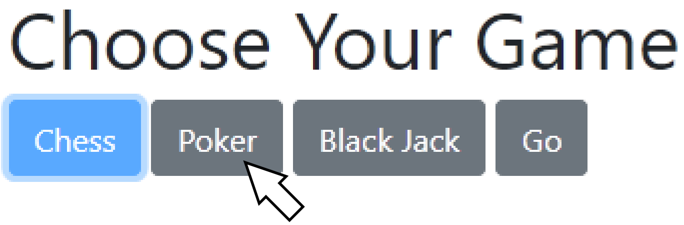
# Function or Object in setVariable?

- If the logic for computing the next state depends on the current state, **always** use a function
- ✗ `setCounter(counter+1)`
  - `counter` is evaluated when `setCounter` **is called**
  - The new state will be assigned later, asynchronously
  - In case many asynchronous requests are made, some update may rely on out-of-date information
- ✓ `setCounter((cnt)=>(cnt+1))`
  - The arrow function will be evaluated when the async call is made, with an up-to-date value of `cnt`: guaranteed to have the latest value

https://medium.com/@wisecobbler/using-a-function-in-setstate-instead-of-an-object-1f5cfd6e55d1

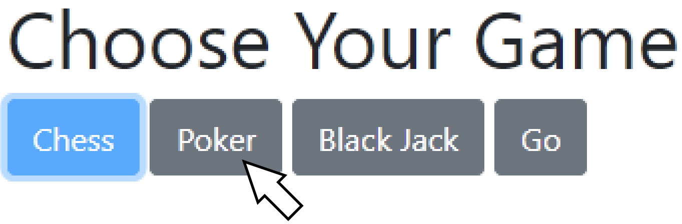
# Can Children Mutate Parent’s State?

- Each button may be selected or not, but only one may be selected at a time
- The information about what button is selected may not be in the button
- It is a state of a container component for “button group”

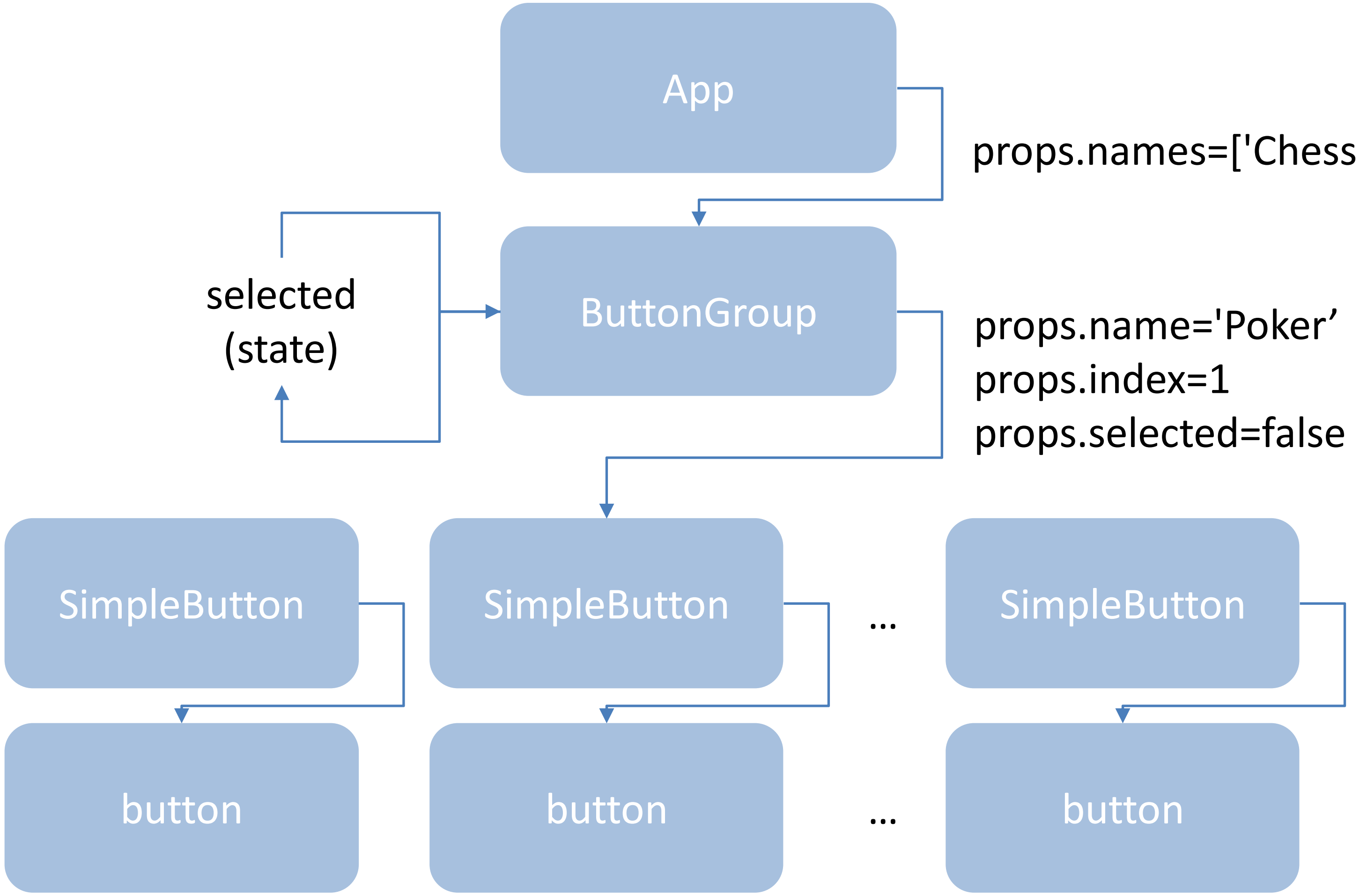


Choose Your Game

# Analysis



Choose Your Game



App

props.names=['Chess', 'Poker', 'Black Jack', 'Go']

selected
(state)

ButtonGroup

props.name='Poker’
props.index=1
props.selected=false

SimpleButton

SimpleButton

…

SimpleButton

<button> attributes
Bootstrap classes

button

button

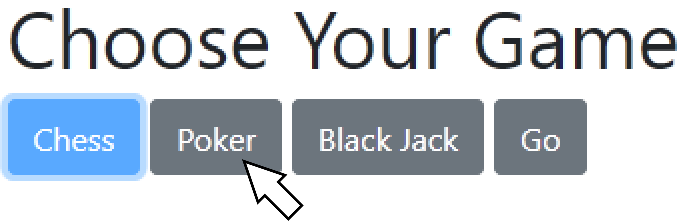
…

button

# How To Change The Chosen Button?

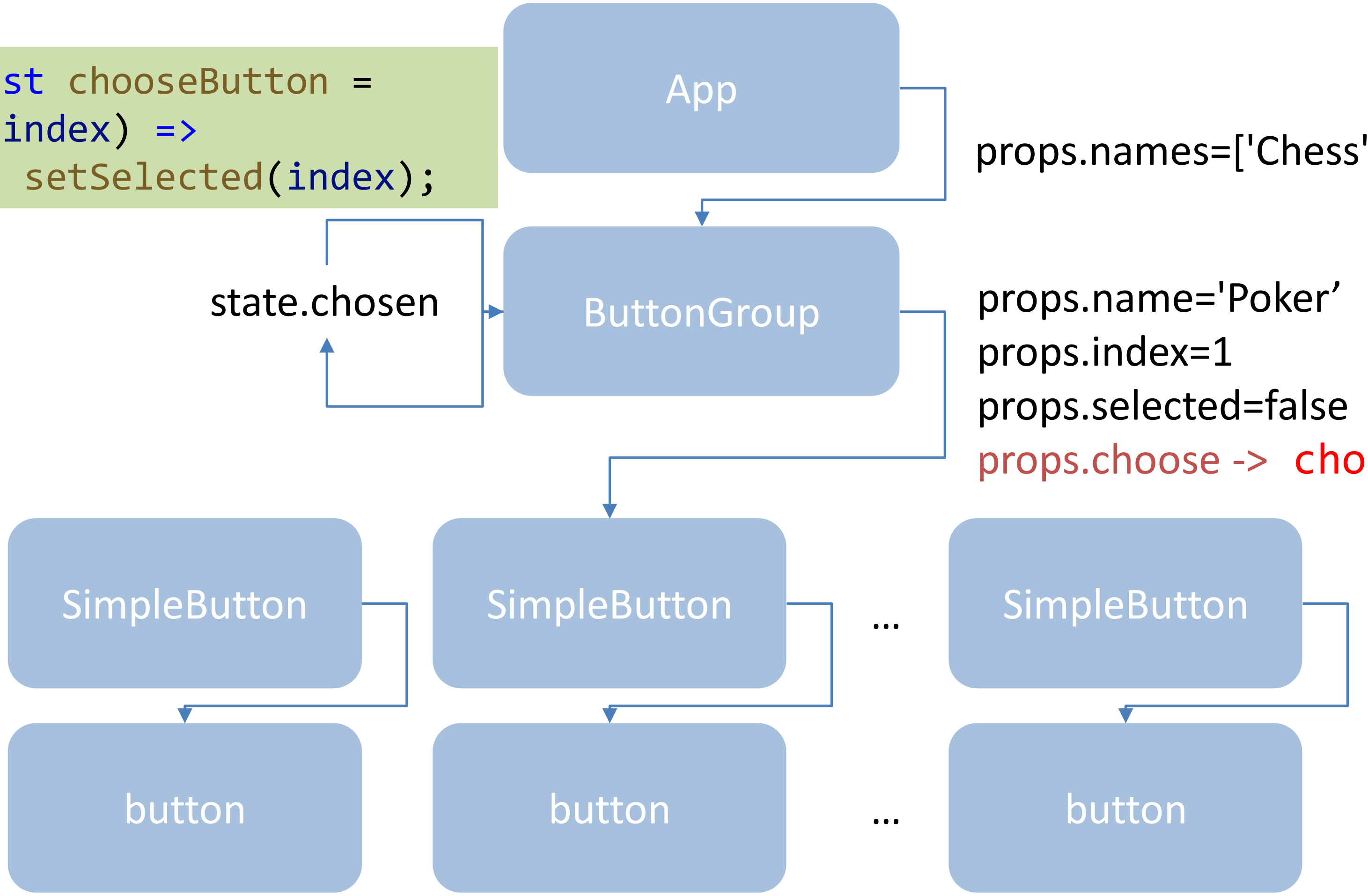
- Handle `onClick` event from the button
- ButtonGroup must offer a method for changing the chosen option
  - will call `setSelected()`
- The method reference must be passed down to SimpleButton, with all other props

# Solution



Choose Your Game

```
const chooseButton =
  (index) =>
    setSelected(index);
```



props.names=['Chess', 'Poker', 'Black Jack', 'Go']

props.name='Poker’
props.index=1
props.selected=false
props.choose -> `choose={chooseButton}`

<button> attributes
Bootstrap classes

```
onClick={() =>
  props.choose(props.index)}
```

# React Design Hints

- Try to implement *stateless* components instead of *stateful* ones
  - Stateless components are more reusable
  - Stateless components are faster to execute
  - Stateless components may be declared as a pure function
- Move *state* to common ancestors (“state lifting”)
- Pass *state* down to the children using *props*
- Allow children to ask for state updates, by passing down callback functions

# License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/