React.js

Lecture 3

Picanteverde
(aka Alejandro Hernández)

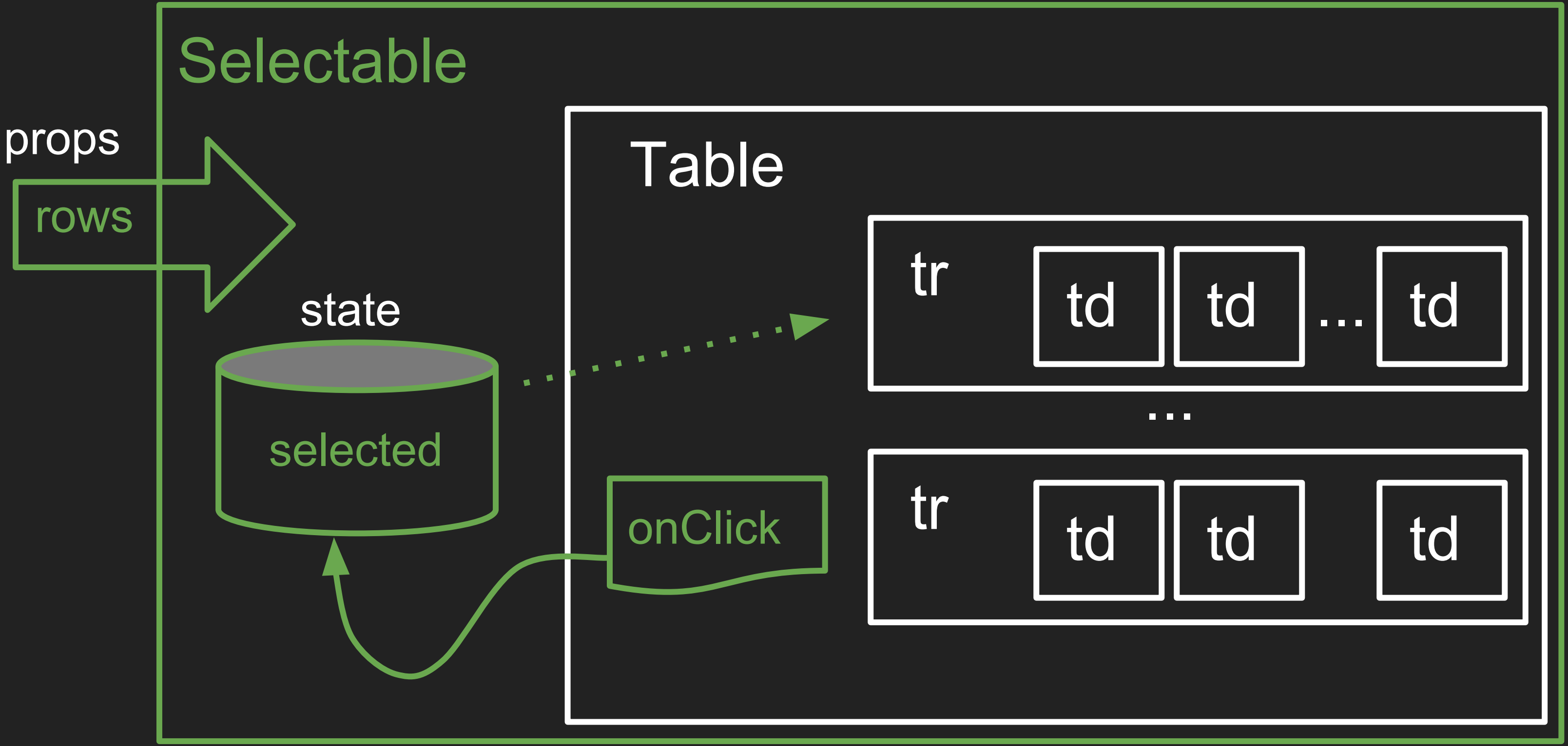
https://commons.wikimedia.org/wiki/File:React.js_logo.svg

BACK ← to REACT

React.js

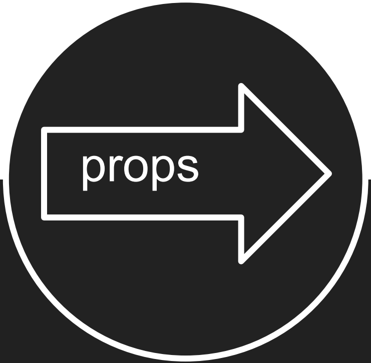
Assignment

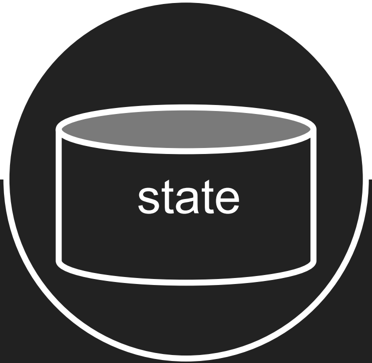
React.js


Selectable
props
rows
state
selected
Table
tr
td
td
...
td
...
tr
td
td
td
onClick

Props

State

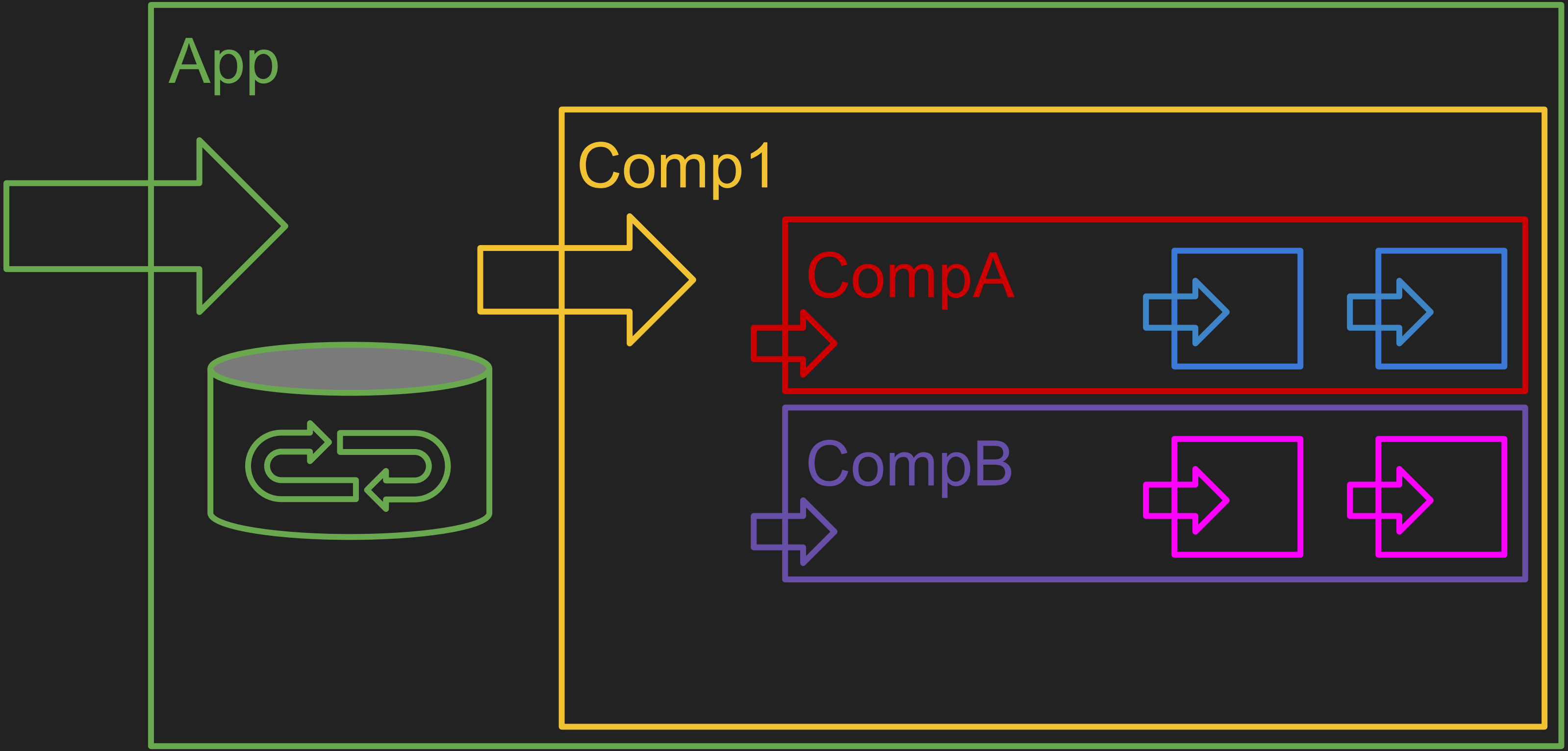




- Data coming the parent element
- As attributes in JSX
- Changes not necessarily trigger a re render
- All data needed (optional props)

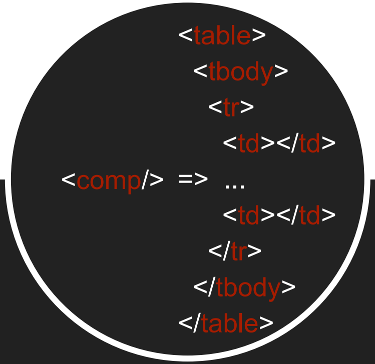
- Data that changes overtime
- Every change triggers a re render
- Minimal amount of data

React.js



Example 1

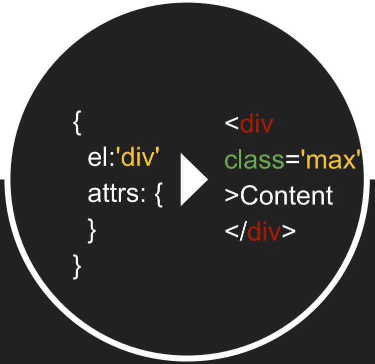
Component Lifecycle



```
             <table>
               <tbody>
                 <tr>
                   <td></td>
<comp/>  =>  ...
                   <td></td>
                 </tr>
               </tbody>
             </table>
```

Mounting

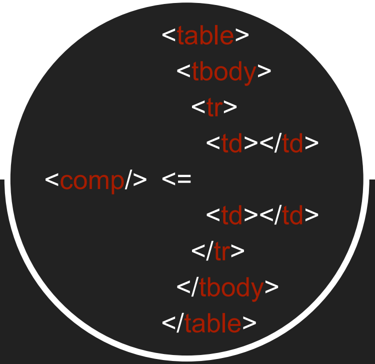
- insert into the DOM
- getInitialState()
- componentWillMount()
- componentDidMount()



```
{                 <div
  el:'div'        class='max'
  attrs: {        >Content
  }               </div>
}
```

Updating

- Re rendering to VDOM
- componentWillReceiveProps(nextProps)
- shouldComponentUpdate(nextProps, nextState)
- componentWillUpdate(nextProps, nextState)
- componentDidUpdate(prevProps, prevState)



```
             <table>
               <tbody>
                 <tr>
                   <td></td>
<comp/>  <=
                   <td></td>
                 </tr>
               </tbody>
             </table>
```

Unmounting

- removing from the DOM
- componentWillUnmount()

Components

Stateless

VS

Stateful

- Simpler
- Only receive Props
- changing the props trigger a re render

- Complex
- have State
- changes in the state trigger a re render
- keep props and state in synch

Example 2

React.js

Forms

React.js

Form elements props

```
<input type="text" value={val} /> <textarea value={val} />

<input type="checkbox" checked={true} />
<input type="radio" checked={true} />

<select>
  <option selected={true} /></option>
</select>
```

React.js

Form elements onChange event

```
<input type="text" onChange={handleOnChange} />

handleOnChange(evt){
  console.log(evt.target.value);
}
```

Form Components

Controlled

VS

Uncontrolled

- control the value
- onChange
- act between the event and the value

- maintain their own internal state
- onChange

React.js

Refs

React.js

Getting references to my instances with the ref attribute

```
<AddNewForm ref={(ref) => {
  this.myInstanceRef = ref;
}} />
```

Allows me to call methods defined on the class

React.js

Server side

React.js

This is not a node.js course

but...

React.js

Ajax?

- XMLHttpRequest
- jQuery.ajax
- Fetch API https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- Axios (get used to Promises)

React.js

Axios

```
$ npm install --save axios

import axios from 'axios';

axios.post(url, data).then((response) => {
  console.log(response.data);
});
```

React.js

When?

For initialization - componentDidMount

On every change

react-router

React.js

react-router

- Allow us to render a component depending on the browser’s route
- Parse parameters and inject them as props
- browser history API

React.js react-router

```
$ npm install --save react-router
import { Router, Route, hashHistory } from 'react-router';

ReactDom.render(
  <Router history={hashHistory}>
    <Route path='/' component={App} />
  </Router>, document.getElementById('start'));
```

React.js

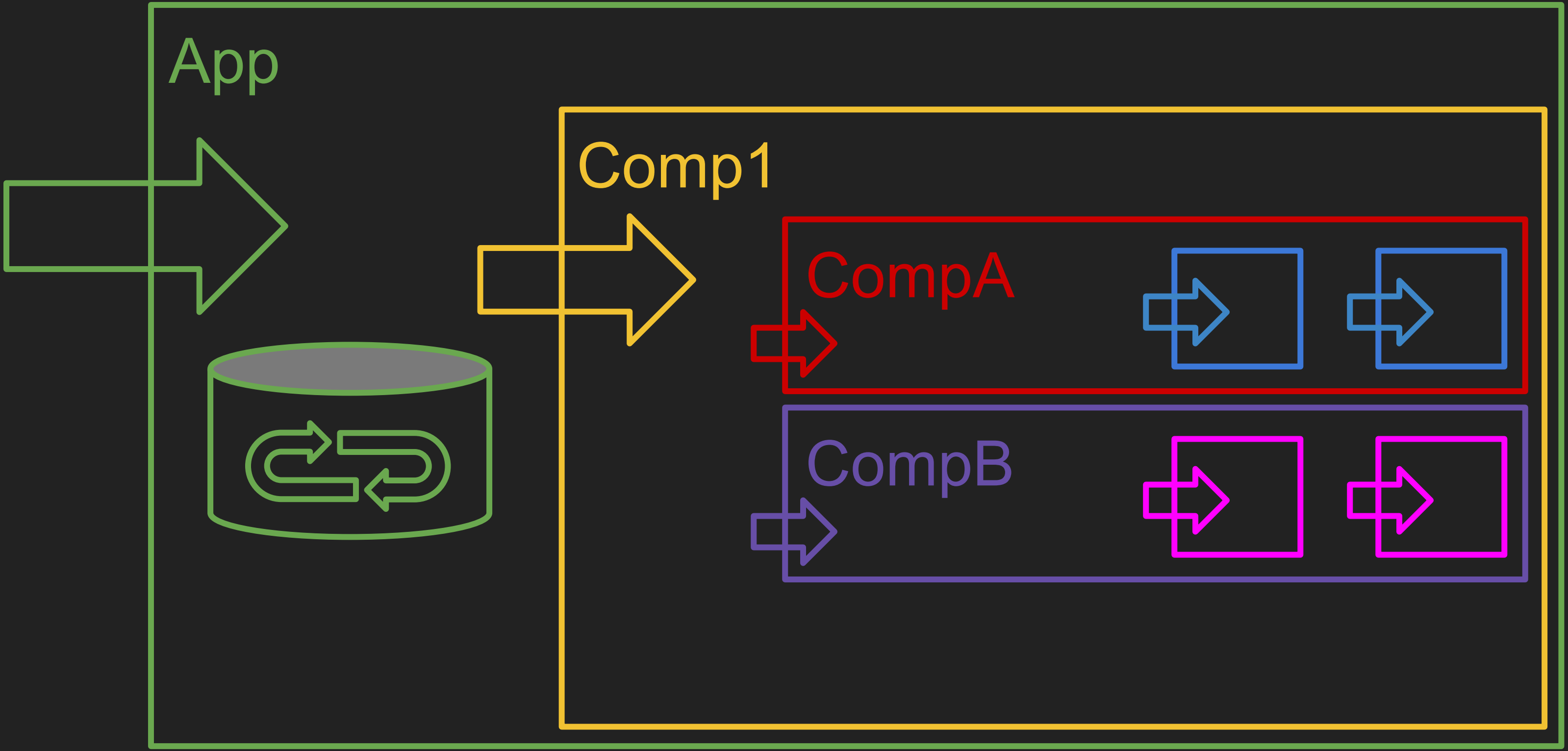
react-router

We could add more screens like this

```
ReactDom.render(
  <Router history={hashHistory}>
    <Route path='/' component={App} />
    <Route path='/add' component={Add} />
    <Route path='/edit' component={Edit} />
  </Router>, document.getElementById('start'));
```

React.js

react-router



React.js react-router

```
<div>
    {this.props.children}
</div>
```

React.js

react-router

```
<Link
  to={op.url}
  activeStyle={{ color: 'red' }}
>
  {op.label}
</Link>
```

React.js

react-router

```
contextTypes: {
  router: PropTypes.object.isRequired
},

edit: function () {
  let route = '/edit/' + this.state.selected;
  this.context.router.push(route);
},
```