React

JavaScript library

Presentation by Diogo Correia

What is React?

- React is a JavaScript library for front-end development.
- React presents the following features:

Declarative view

- React makes it easier to create UIs through it's declarative programming views, in a way that you have to say what you want, not how you want it.

Components

- React uses a technique of dividing the code into encapsulated components, making use of Object-oriented Programming with JavaScript's classes.

User Interface

- Each component renders a piece of HTML code, with actual HTML syntax, avoiding the painful functions from the document, in JavaScript, to manipulate elements.

How does React work?

Setup

- To setup React locally, it is used NodeJS package manager (NPM).
- By running the following command

```
npx create-react-app my-app
```

you create a development environment.

- When inside the app directory, the app will be running on localhost after running the command

```
npm start
```

- The *.js* files should start with

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
```

in order to gain access to the necessary classes and methods.

How does React work?

Writing code

- As stated previously, React code is based on components.
  - can be seen as a section of the interface;
  - can have children components, which can receive information from its parent through props;
  - the information in each component is stored in states.

- The document is rendered a first time using ReactDOM.render and by “passing” the component which renders the first document elements. To update the stored data in a certain component and re-render the interface, you can either use the function *setState()* or, if that component doesn’t have a state, you can just use the function *forceUpdate()*.

How does React work?

- **Writing code**
  - **Tic tac toe**

```
class Board extends React.Component {
  constructor(props) {
    ...
  }

  renderSquare(i) {
    return (
      <Square
        value={this.state.squares[i]}
        onClick= {() => this.handleClick(i)}
      />
    );
  }
}
```

parent

```
class Square extends React.Component {
  render() {  // renders specific elements
    return (
      <button
        className="square"
        onClick={() => this.props.onClick()}
      >
        {this.props.value}
      </button>
    );
  }
}
```

child

From React tutorial @https://reactjs.org/

How does React work?

- In the interaction that we just saw in the previous slide between Board and Square, we are passing down two props from Board to Square: value and onClick.

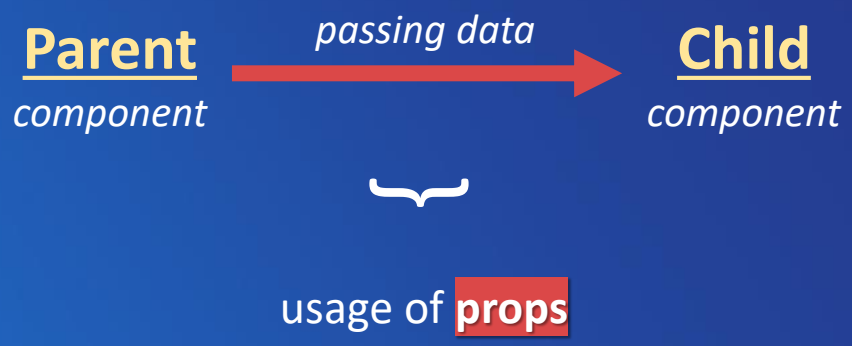
To be more specific…

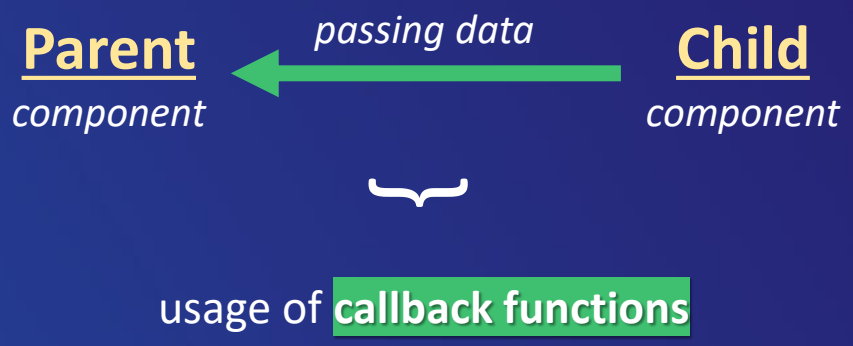
- Board is keeping an array *squares* with the information of the data in each square of the tictactoe game. It also has a function that deals with a click event, *handleClick()*. Square uses *handleClick()* to fill the clicked position in *squares*, and then uses *squares* to get the value that was just filled by the click handler, either 'X' or 'O', and then writes it in the <button>.

Why React?

- React makes UI updates a much easier and more intuitive job.

- There is a very good connection between parent a children components, making it easy to access values within one another through props or callback functions.





Code examples

1 Tic-tac-toe

```
ReactDOM.render(
  <Game />,
  document.getElementById('root')
);
```

```
class Game extends React.Component {
  render() {
    return (
      <div className="game">
        <div className="game-board">
          <Board />
        </div>
        <div className="game-info">
          <div>{/* status */}</div>
          <ol>{/* TODO */}</ol>
        </div>
      </div>
    );
  }
}
```

```
class Board extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      squares: Array(9).fill(null),
      xTurn: true
    };
  }

  handleClick(i) {
    const squares = this.state.squares.slice();
    if (calculateWinner(squares) || squares[i]) {
      return;
    }
    squares[i] = this.state.xTurn ? 'X' : 'O';
    this.setState({
      squares: squares,
      xTurn: !this.state.xTurn
    });
  }

  renderSquare(i) {
    return (
      <Square
        value={this.state.squares[i]}
        onClick= {() => this.handleClick(i)}
      />
    );
  }

  render() {
    const winner = calculateWinner(this.state.squares);
    let status;
    if (winner) {
      status = 'Winner: ' + winner;
    }
    else {
      status = 'Next player: ' + (this.state.xTurn ? 'X' : 'O');
    }

    return (
      <div>
        <div className="status">{status}</div>
        <div className="board-row">
          {this.renderSquare(0)}
          {this.renderSquare(1)}
          {this.renderSquare(2)}
        </div>
        <div className="board-row">
          {this.renderSquare(3)}
          {this.renderSquare(4)}
          {this.renderSquare(5)}
        </div>
        <div className="board-row">
          {this.renderSquare(6)}
          {this.renderSquare(7)}
          {this.renderSquare(8)}
        </div>
      </div>
    );
  }
}
```

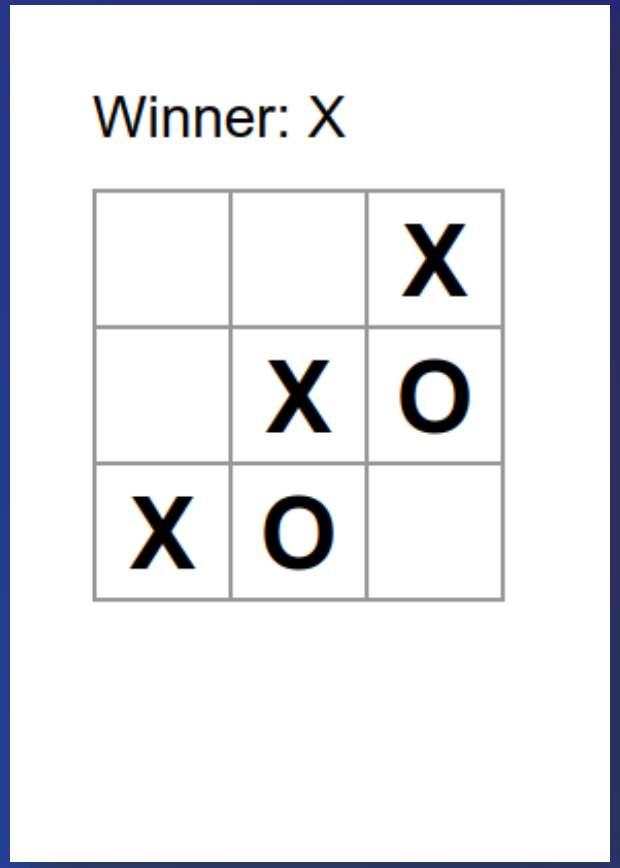
```
function Square(props) {
  return (
    <button
      className="square"
      onClick={props.onClick}
    >
      {props.value}
    </button>
  );
}
```

Doesn't need to be a class because it is just returning

From React tutorial @https://reactjs.org/

Code examples

1. **Tic-tac-toe**



Code examples

2 TODO list

```
ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

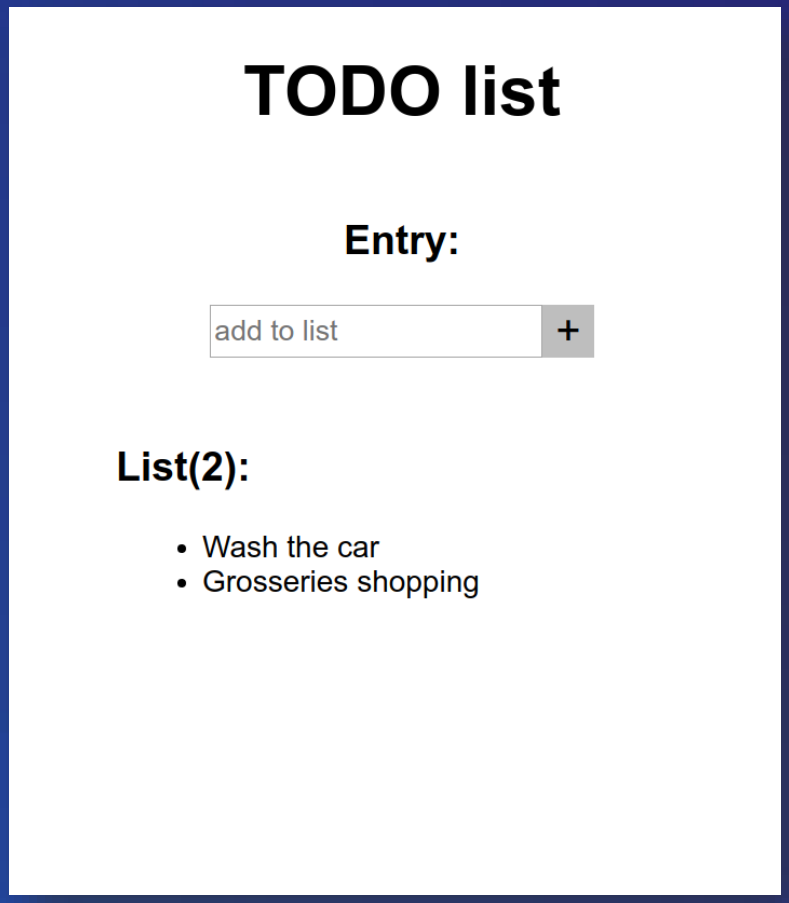
```
class List extends React.Component {
  render() {
    if(this.props.items.length === 0) {
      return (
        <p id="empty-list">The list is empty!</p>
      );
    }
    else {
      return (
        <ul>
          {this.props.items.map(item => (
            <li key={"item"+this.props.items.length} className="item">{item}</li>
          ))}
        </ul>
      );
    }
  }
}
```

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      inputVals: []
    }
  }

  updateVals = value => this.setState(state => ({inputVals: state.inputVals.concat(value)}));

  render() {
    return (
      <div className="todolist-container">
        <h1>TODO list</h1>
        <div className="middle">
          <div>
            <Input
              getVal={this.updateVals}
              app={this}
            />
          </div>
          <div className="content">
            <h3 id="list">List({this.state.inputVals.length}):</h3>
            <List items={this.state.inputVals} />
          </div>
        </div>
      </div>
    );
  }
}
```

```
class Input extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      inputVal: '',
      input: ''
    };
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(e) {
    this.setState({
      inputVal: e.target.value,
      input: e.target
    });
  }

  handleClick() {
    if (this.state.inputVal.length === 0) {
      return;
    }

    // send upwards the value of the input, through a callback
    this.props.getVal(this.state.inputVal);

    let input = this.state.input;
    input.value = '';

    this.setState({
      inputVal: '',
      input: ''
    });
  }

  render() {
    return (
      <div>
        <h3>Entry: </h3>
        <div className="inputs-wrapper">
          <input
            placeholder="add to list"
            onChange={this.handleChange}
          ></input>
          <button
            id="add"
            onClick={this.handleClick.bind(this)}
          >+</button>
        </div>
      </div>
    );
  }
}
```

Code examples

2 TODO list



Code examples

3 Calculator

```
ReactDOM.render(
  <Calculator />
  document.getElementById('root')
);
```

```
function CalculatorButton(props) {
  return (
    <button onClick={props.onClick} >
      {props.val}
    </button>
  );
}
```

```
class Calculator extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      op1: '',
      operand: '',
      op2: '',
      res: ''
    };
    this.renderButton = this.renderButton.bind(this);
  }

  handleClick (val) {
    if (this.state.res !== '') {
      this.setState({
        op1: '',
        operand: '',
        op2: '',
        res: ''
      });
    }

    if (val === 'C') {
      this.setState({
        op1: '',
        operand: '',
        op2: '',
        res: ''
      });
      return;
    }
// if val is a number
if (!isNaN(val)) {
  // if there is not an operand yet
  if (this.state.operand === '') {
    this.setState({
      op1: this.state.op1+val,
      operand: '',
      op2: '',
      res: ''
    });
  }
  else {
    this.setState({
      op1: this.state.op1,
      operand: this.state.operand,
      op2: this.state.op2+val,
      res: ''
    });
  }
}
else {
  if (this.state.operand === '' && this.state.op1 !== '') {
    this.setState({
      op1: this.state.op1,
      operand: val,
      op2: '',
      res: ''
    });
  }
  else if(this.state.op1 !== '' && this.state.op2 !== '' && this.state.operand !== '') {
    if (val === '=') {
      let result = '';
      let op1 = parseInt(this.state.op1);
      let op2 = parseInt(this.state.op2);
      switch(this.state.operand) {
        case '+':
          result = op1+op2;
          break;
        case '-':
          result = op1-op2;
          break;
        case 'x':
          result = op1*op2;
          break;
        case '/':
          result = op1/op2;
          break;
        default:
          result = 'ERROR';
          break;
      }

      this.setState({
        op1: '',
        operand: '',
        op2: '',
        res: result
      });
    }
    else {
      return;
    }
  }
}

renderButton(val) {
  return (
    <CalculatorButton
      onClick= {() => this.handleClick(val)}
      val={val}
    />
  );
}

render() {
  return (
    <div>
      <h1 id="title">Calculator</h1>
      <div className="calculator-container">
        <div className="calculator">
          <div className="screen">
            <Screen
              op1={this.state.op1}
              op2={this.state.op2}
              operand={this.state.operand}
              res={this.state.res}
            />
          </div>
          <div className="buttons">
            <div className="numbers">
              <Numbers renderButton={this.renderButton} />
            </div>
            <div className="operators">
              <Operators renderButton={this.renderButton} />
            </div>
          </div>
        </div>
      </div>
    );
  }
}
```

Code examples

3 Calculator

```
...
render() {
  return (
    <div>
      <h1 id="title">Calculator</h1>
      <div className="calculator-container">
        <div className="calculator">
          <div className="screen">
            <Screen
              op1={this.state.op1}
              op2={this.state.op2}
              operand={this.state.operand}
              res={this.state.res}
            />
          </div>
          <div className="buttons">
            <div className="numbers">
              <Numbers renderButton={this.renderButton} />
            </div>
            <div className="operators">
              <Operators renderButton={this.renderButton} />
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}
}
```

```
class Screen extends React.Component {
  render() {
    if (this.props.res === '') {
      return (
        <p>
          {this.props.op1}
          <span className="space"></span>
          {this.props.operand}
          <span className="space"></span>
          {this.props.op2}

        </p>
      );
    }
    else {
      return (
        <p>
          {this.props.res}
        </p>
      );
    }
  }
}
```

```
class Numbers extends React.Component {
  render() {
    return (
      <div>
        <div className="nums-row">
          {this.props.renderButton(1)}
          {this.props.renderButton(2)}
          {this.props.renderButton(3)}
        </div>
        <div className="nums-row">
          {this.props.renderButton(4)}
          {this.props.renderButton(5)}
          {this.props.renderButton(6)}
        </div>
        <div className="nums-row">
          {this.props.renderButton(7)}
          {this.props.renderButton(8)}
          {this.props.renderButton(9)}
        </div>
        <div className="nums-row">
          {this.props.renderButton('C')}
          {this.props.renderButton(0)}
          {this.props.renderButton('=')}
        </div>
      </div>
    );
  }
}
```

```
class Operators extends React.Component {
  render() {
    return (
      <div>
        {this.props.renderButton('+')}
        {this.props.renderButton('-')}
        {this.props.renderButton('x')}
        {this.props.renderButton('/')}
      </div>
    );
  }
}
```

Code examples

3 **Calculator**