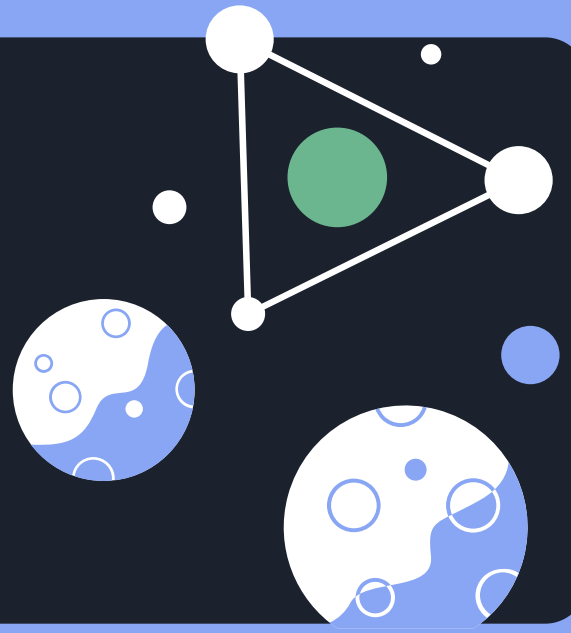
# Intro to React



By: Ahmed Kamran'23

TIW 2023

# About me

I am a senior CS student. I have taught myself a lot of web dev tools from online resources to develop my personal projects.
I have developed a few projects using React.

# Overview

01 What is React?

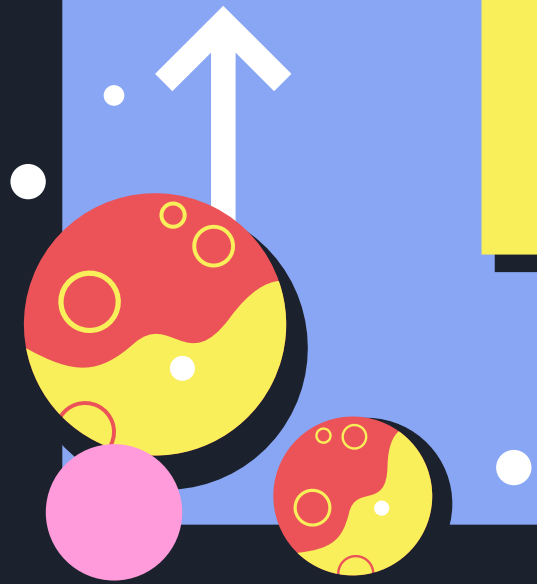
02 React Features

03 Building a ToDo List app

04 Hosting your app online

# 01

# What is React?

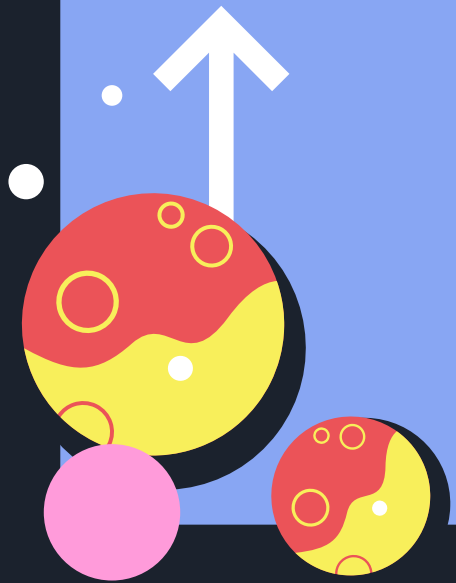


# Elements of a web app

- HTML provides the structure and content of a web page. It defines the various elements on a page, such as headings, paragraphs, lists, and links.
- CSS (Cascading Style Sheets) is used to style and layout web pages. It controls the appearance of HTML elements, including the colors, fonts, spacing, and layout.
- JavaScript is a programming language used to add interactivity and dynamic behavior to web pages. It can be used to respond to user actions, manipulate the DOM, and make network requests.
- React is JavaScript library that allows developers to create user interfaces using JavaScript

# 02

# React Features



# Features of React

- Component-Based Architecture
- Virtual DOM
- Declarative Programming Style
- Easy State Management
- JSX Syntax


TIW 2023

# Interactive Session

- Go to GitHub Codespaces (*github.com/codespaces*)
- Chose the React Template
- Launch the template app
- Edit the code together

# src/App.js – Create initial components

```
1  import './App.css';
2  import React, { useState } from 'react';
3
4  function App() {
5    const [todos, setTodos] = useState([]);
6    return (
7      <div className='App'>
8        <h1>To-Do List</h1>
9        <form>
10         <input type="text" />
11         <button>Add</button>
12       </form>
13       <ul>
14         {todos.map((todo, index) => (
15           <li key={index}>{todo}</li>
16         ))}
17       </ul>
18     </div>
19   );
20 }
21
22 export default App;
```

useState() assigns an empty list to the variable todos and creates a function setTodos().

We use todos.map() to render a list of all the todos

setTodos() is a function that allows us to change the state of todos (add, edit, delete)

State changes trigger UI updates in React

# src/App.js – Add new todos

```
const addTodo = (event) => {
  event.preventDefault();
  const newTodo = event.target.elements[0].value;
  setTodos([...todos, newTodo]);
  event.target.reset();
}

return (
  <div className='App'>
    <h1>To-Do List</h1>
    <form onSubmit={addTodo}>
      <input type="text" />
      <button>Add</button>
    </form>      You, 1 minute ago • Uncommitted changes
    <ul>
      {todos.map((todo, index) => (
        <li key={index}>{todo}</li>
      ))}
    </ul>
  </div>
);
```

Create a function addTodo() that uses the setTodos() function to append a new item to the list of todos

Assign addTodo() as an eventListener to the form's onSubmit event

This will cause a new todo to be added on every form submission

# src/App.js – Delete todos

```
const deleteTodo = (index) => {
  const newTodos = [...todos];
  newTodos.splice(index, 1);
  setTodos(newTodos);
}

return (
  <div className='App'>
    <h1>To-Do List</h1>
    <form onSubmit={addTodo}>
      <input type="text" />
      <button>Add</button>
    </form>
    <ul>
      {todos.map((todo, index) => (
        <li key={index}>
          {todo}
          <button onClick={() => deleteTodo(index)}>Delete</button>
        </li>
      ))}
    </ul>
  </div>
);      You, 36 minutes ago • Initial commit …
```

Create a function deleteTodo() that uses the setTodos() function to replace the existing todo-list with a modified one that deletes the selected item

Assign deleteTodo() as an eventListener to the delete button's onClick event

This will cause a new todo to be added on every form submission

# src/App.css - Styling the app

## Paste the code below into App.css

```
.App {
text-align: center;
padding: 1em;
}

body {
font-family: Arial, sans-serif;
}

form {
margin-bottom: 16px;
}

input[type="text"] {
padding: 8px;
font-size: 16px;
border-radius: 4px;
border: 1px solid #ccc;
}

button {
padding: 8px;
font-size: 16px;
border-radius: 4px;
border: none;
background-color: #007bff;
color: #fff;
cursor: pointer;
margin-left: 1em;
}

button:hover{
background-color: #0069d9;
}

ul {
list-style-type: none;
padding: 0;
}
li {
display: flex;
justify-content: space-
between;
align-items: center;
padding: 8px;
margin-bottom: 8px;
border: 1px solid #ccc;
border-radius: 4px;
}
```

# src/App.js – Storing todos in local storage

```
import './App.css';
import React, { useState, useEffect } from 'react';

function getTodosFromLocalStorage() {
  const todos = localStorage.getItem('todos');
  if (todos) {
    return JSON.parse(todos);
  }
  return [];
}

function App() {
  const [todos, setTodos] = useState(getTodosFromLocalStorage())

  useEffect(() => {
    localStorage.setItem('todos', JSON.stringify(todos));
  }, [todos]);
```

Create a function getTodosFromLocalStorage() that retrieves the todos locally stored in the browser

Modify the useState function to initialize the list of todos with the items stored in local storage

Use the useEffect() function to store the newly updated list of todos in local storage every time there is a state change in the todos variable

# Components – src/TodoList.js

```jsx
import React from 'react';

function TodoList({ todos, deleteTodo }) {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={index}>
          {todo}
          <button onClick={() => deleteTodo(index)}>Delete</button>
        </li>
      ))}
    </ul>
  );
}

export default TodoList;
```

Create a file called TodoList.js in the src folder to store a new component that will contain the list of todos

# Components – src/App.js

```
You, 3 minutes ago | 1 author (You)
import './App.css';
import React, { useState, useEffect } from 'react';
import TodoList from './Todolist';
```

```
return (
  <div className='App'>
    <h1>To-Do List</h1>
    <form onSubmit={addTodo}>
      <input type="text" />
      <button>Add</button>
    </form>
    <TodoList todos={todos} deleteTodo={deleteTodo} />
  </div>
);
```

Import the TodoList component and render it
Pass props on to it to be rendered

# Adding IDs to Todos – src/App.js

```
function App() {
  const [todos, setTodos] = useState(getTodosFromLocalStorage())
  const [idCount, setIdCount] = useState(0);

  useEffect(() => {
    localStorage.setItem('todos', JSON.stringify(todos));
  }, [todos]);

  const addTodo = (event) => {
    event.preventDefault();
    const newTodoText = event.target.elements[0].value;
    const newTodo = { id: idCount, text: newTodoText, completed: false };
    setIdCount(idCount + 1);
    setTodos([...todos, newTodo]);
    event.target.reset();
  }

  const deleteTodo = (id) => {
    const newTodos = todos.filter((todo) => todo.id !== id);
    setTodos(newTodos);
  }
```

Add IDs to Todos so we can make edits to each todo

Create a idCount state variable to generate new IDs

Convert the todo to an object with an id property

Convert the delete function to use the ID rather than the index

# Adding IDs to Todos – src/TodoList.js

```
function TodoList({ todos, deleteTodo }) {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={todo.id}>
          {todo.text}
          <button onClick={() => deleteTodo(todo.id)}>Delete</button>
        </li>
      ))}
    </ul>
  );
}
```

Render the todo.text for each todo

Pass the todo.id when invoking the delete function

# Marking ToDos complete - src/App.js

```
const toggleTodo = (id) => {
  const newTodos = todos.map((todo) => {
    if (todo.id === id) {
      return { id: todo.id, text: todo.text, completed: !todo.completed };
    }
    return todo;
  });
  setTodos(newTodos);
}

return (
  <div className='App'>
    <h1>To-Do List</h1>
    <form onSubmit={addTodo}>
      <input type="text" />
      <button>Add</button>
    </form>
      <TodoList todos={todos} deleteTodo={deleteTodo} toggleTodo={toggleTodo} />
  </div>
);
```

Create a togggleTodo function that takes the id of a todo and changes the state of the todos to replace that item with a toggled item

Pass the toggleTodo function as a prop to the TodoList component

# Marking ToDos complete - src/TodoList.js

```
function TodoList({ todos, deleteTodo, toggleTodo }) {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={todo.id}>
          <input type="checkbox" checked={todo.completed} onChange={() => toggleTodo(todo.id)} />
          {todo.text}
          <button onClick={() => deleteTodo(todo.id)}>Delete</button>
        </li>
      ))}
    </ul>
  );
}
```

Add a checkbox input in the TodoList component to check Todos complete or incomplete

# Stylizing completed todos – src/TodoList.js

```
function TodoList({ todos, deleteTodo, toggleTodo }) {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={todo.id} className={todo.completed ? 'completed':''}>
          <input type="checkbox" checked={todo.completed} onChange={() => toggleTodo(todo.id)} />
          {todo.text}
          <button onClick={() => deleteTodo(todo.id)}>Delete</button>
        </li>
      ))}
    </ul>
  );
}
```

Add a css class to the list items that indicate if a todo is completed or not

# Stylizing completed todos - App.css

```css
.completed {
  text-decoration: line-through;
  color: #aaa;
}
```

Add a css class to the list items that indicate if a todo is completed or not

# Deployment to the web with Vercel

How can we deploy our React app permanently on the web

We will use a service called Vercel which makes deployment from GitHub extremely easy and has a generous free tier

Go to vercel.com and follow along

# Thanks!

Have any questions or want further resources?

akamran@colgate.edu

Go to our website for the slides and code
bit.ly/TIW2023/