Web Development

CSE343

Lab 6

More React

Recap

- Last time we saw that we can use React to break down the UI into reusable components
- Components are functions that return JSX
- We were able to update the screen using states
- Any update to the state will be reflected on the screen
- We saw how to pass props ("arguments") to components

Agenda

- Following the tutorials in: https://react.dev/learn/
- High level overview of the steps React does to display the UI
- How React interacts with state
- How to manage your state

Render and Commit

- Before your components are displayed on screen, they must be rendered by React.
- Understanding the steps in this process will help you think about how your code executes and explain its behavior.
  1. Triggering a render
  2. Rendering
  3. Committing

UI update steps – triggering a render

- There are two reasons for a component to render:
  - It’s the component’s initial render.
    - When your app starts, you need to trigger the initial render.
    - Generated automatically most of the time.
  - The component’s (or one of its ancestors’) state has been updated.
    - Whenever a state is "set" using setState, a render is triggered

UI update steps – rendering

- React basically calls the component
- The process is recursive: if the updated component returns some other component, React will render that component next, until there are no components left

UI update steps – committing

- After rendering (calling) your components, React will modify the DOM.
  - For the initial render, React will use the appendChild() DOM API to put all the DOM nodes it has created on screen.
  - For re-renders, React will apply the minimal necessary operations (calculated while rendering!) to make the DOM match the latest rendering output.
- React only changes the DOM nodes if there's a difference between renders.

How React interacts with state

- State variables might look like regular JavaScript variables that you can read and write to.
- However, state behaves more like a snapshot. Setting it does not change the state variable you already have, but instead **triggers a re-render**.

How React interacts with state

- State variables might look like regular JavaScript variables that you can read and write to.
- However, state behaves more like a snapshot. Setting it does not change the state variable you already have, but instead **triggers a re-render**.

How React interacts with state

- To further understand this let's look at this example
- What do you think will happen when this button is clicked?

```
export default function Counter() {
  const [number, setNumber] = useState(0);

  return (
    <>
      <h1>{number}</h1>
      <button onClick={() => {
        setNumber(number + 1);
        setNumber(number + 1);
        setNumber(number + 1);
      }}>+3</button>
    </>
  )
}
```

How React interacts with state

- To further understand this let's look at this example
- What do you think will happen when this button is clicked?
- It only increments once per click!
- Why?
  - Setting state only changes it for the next render.
- Let's trace it

```
export default function Counter() {
  const [number, setNumber] = useState(0);

  return (
    <>
      <h1>{number}</h1>
      <button onClick={() => {
        setNumber(number + 1);
        setNumber(number + 1);
        setNumber(number + 1);
      }}>+3</button>
    </>
  )
}
```

How React interacts with state

- Like we said, React treats the state variable as snapshots
- So if we substitute in the handler function

```
<button onClick={() => {
  setNumber(number + 1);
  setNumber(number + 1);
  setNumber(number + 1);
}}>+3</button>
```

```
<button onClick={() => {
  setNumber(0 + 1);
  setNumber(0 + 1);
  setNumber(0 + 1);
}}>+3</button>
```

How React interacts with state

- Let's look at this example
- To further understand why this happens, we need to understand that React batches state updates
- It waits for all the code in the event handlers to finish, then it performs the render
- This lets you update multiple state variables—even from multiple components—without triggering too many re-renders.

How React interacts with state

- If you want to update a state multiple times in a function (like our case of incrementing it 3 times), we can pass a function to setState that calculates the value of the state based on the previous value

```
setNumber(n => n + 1);
setNumber(n => n + 1);
setNumber(n => n + 1);
```

How React interacts with state

- Let's solve this challenge

How to manage your state

- As your application grows, the amount of state you are holding grows
- You need to carefully design your state management
- Redundant or duplicate state is a common source of bugs
- Let's look at this example

How to manage your state

- Sometimes, you want the state of two components to always change together
- To do it, remove state from both of them, move it to their closest common parent, and then pass it down to them via props
- This is known as “lifting state up”, and it’s one of the most common things you will do writing React code
- example

Next steps in React

- Install React to run it locally. Just ChatGPT it
- We have gone over the main concepts of React. However, React is a huge framework with tons of features and APIs that cannot be contained in a 3 hours lab.
- If you get stuck, you can always ask me or ask your best friend ChatGPT (just promise you won't copy and paste the code).
- What I would recommend that you ask ChatGPT about the name of the concept, then googling it and read the documentation. That's how you learn!

Extra Resources

- How to make multipage web react apps: React Router
- In depth Learn React (This is where I got the lab content)
- Send React to the server side with next.js
- If you want a complete frontend bootcamp, I recommend this course. It's very long but it covers everything!

Assignment

- You need to implement the udemy landing page: https://www.udemy.com/
- Open the link in incognito to log you out
- You can skip the animations
- Implement until the "trusted companies" section
- Place all the dummy data in javascript objects and arrays. **We will need it later**
- You can use AI but there will be a discussion next lab
- You need to understand everything!

New-learner offer | Courses from E£259.99. Click button to see savings.
Ends in 5h 56m 33s.

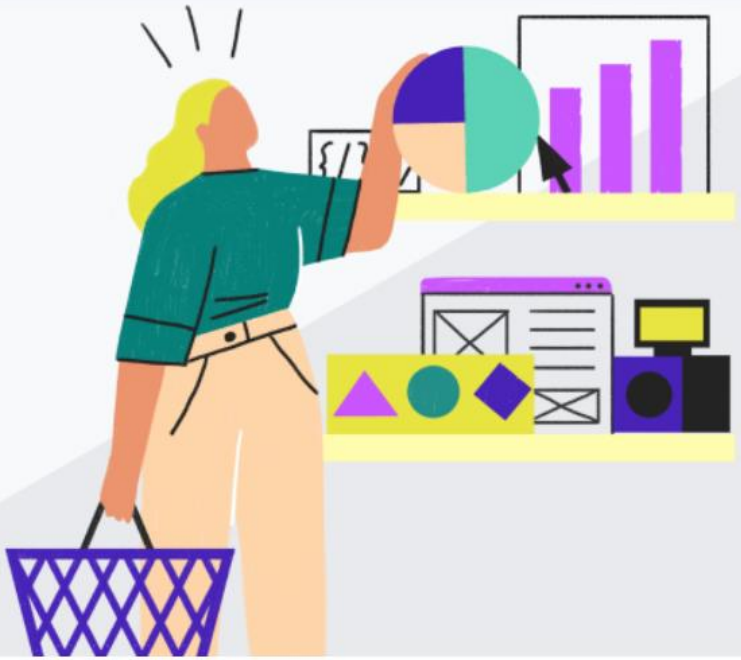
Click to redeem

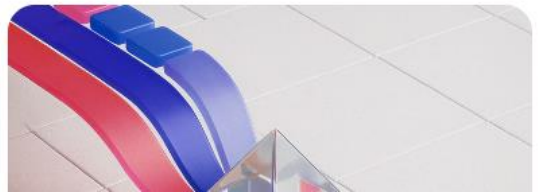
udemy Explore Search for anything Plans & Pricing Udemy Business Teach on Udemy Log in Sign up

Jump into learning — for less

If you're new to Udemy, we've got good news: For a limited time, courses start at just E£259.99 for new learners!

Sign up now





Learn essential career and life skills

Udemy helps you build in-demand skills fast and advance your career in a changing job market.

1M+

Generative AI

14.4M+

IT Certifications

8M+

Data Science

Reimagine your career in the AI era

Future-proof your skills with Personal Plan. Get access to a variety of fresh content from real-world experts.

- **Learn** AI and more
- **Prep** for a certification
- **Practice** with AI coaching
- **Advance** your career

Learn more

Starting at E£204.00/month

Skills to transform your career and life

From critical skills to technical topics, Udemy supports your professional development.

Artificial Intelligence (AI) | Python | Microsoft Excel | AI Agents & Agentic AI | Digital Marketing | Amazon AWS

The AI Engineer Course 2025: Complete AI Engineer Bootcamp

365 Careers

Bestseller | 4.6 | 10,801 ratings

E£349.99

Intro to AI Agents and Agentic AI

365 Careers

Bestseller | 4.4 | 895 ratings

E£349.99

Artificial Intelligence (AI) Foundations for Developers

Rahul Rajat Singh

Highest Rated | 5.0 | 21 ratings

E£349.99

The Complete Guide To AI Powered Salesforce Development

Matt Gerry

Hot & New | 4.8 | 74 ratings

E£349.99

Show all Artificial Intelligence (AI) courses →

Trusted by over 17,000 companies and millions of learners around the world