

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. The background is a dark navy blue with subtle diagonal lines.

Conditional rendering



Agenda

- Conditional rendering
- Debuggning
- Hooks
- Repetition av event handling
- Funktioner som returnerar funktioner
- Skicka event handlers till barnkomponenter



Conditional rendering

Ofta vill vi kunna låta en komponent bestämma
hur och när den ska visa saker....

```
const History = (props) => {  
  if (props.allClicks.length === 0) {  
    return (  
      <div>  
        the app is used by pressing the buttons  
      </div>  
    )  
  }  
  return (  
    <div>  
      button press history: {props.allClicks.join(' ')}  
    </div>  
  )  
}
```

```
const App = () => {  
  // ...  
  
  return (  
    <div>  
      {left}  
      <button onClick={handleLeftClick}>left</button>  
      <button onClick={handleRightClick}>right</button>  
      {right}  
      <History allClicks={allClicks} />  
    </div>  
  )  
}
```



Conditional rendering

Övning 1: skapa en List komponent som

- om listan är tom visar: `<p>“Sorry empty”</p>`
- om listan har flera element visar dessa i ``

Övning 2: Lägg tillägg i List komponenten

- om listan har bara 1 värde visar: `<p>{value}</p>`

```
const History = (props) => {  
  if (props.allClicks.length === 0) {  
    return (  
      <div>  
        the app is used by pressing the buttons  
      </div>  
    )  
  }  
  return (  
    <div>  
      button press history: {props.allClicks.join(' ')}  
    </div>  
  )  
}
```

```
const App = () => {  
  // ...  
  return (  
    <div>  
      {left}  
      <button onClick={handleLeftClick}>left</button>  
      <button onClick={handleRightClick}>right</button>  
      {right}  
      <History allClicks={allClicks} />  
    </div>  
  )  
}
```



Conditional rendering

Vi kan också bestämma utifrån vad en komponent ska visa

```
return (  
  <div>  
    {clicks.left}  
    <button onClick={handleLeft}>Left</button>  
    <button onClick={handleRight}>Right</button>  
    {clicks.right}  
    <br />  
    <p>{allClicks.join(" ")}</p>  
    {allClicks.length == 0 ? <p>Empty</p> : <p>{allClicks.length}</p>}  
  </div>  
)
```



Conditional rendering

Övning: Skapa en komponent `<ClicksCounter size={allClicks.length} />`

```
return (  
  <div>  
    {clicks.left}  
    <button onClick={handleLeft}>Left</button>  
    <button onClick={handleRight}>Right</button>  
    {clicks.right}  
    <br />  
    <p>{allClicks.join(" ")}</p>  
    {allClicks.length == 0 ? <p>Empty</p> : <p>{allClicks.length}</p>}  
  </div>  
)
```



Conditional rendering

Övning: Skapa en komponent som består av en show-knapp och en hide-knapp.

- Trycker man på show-knapp ska en bild visas ()
- Trycker man på hide-knapp ska bilden döljas

Tips: använd conditional rendering...



Debuggning

- En stor del av tiden som en utvecklare arbetar debuggar hon. (70 %)
- En del av tiden läser hon dokumentation (10 %)
- En del av tiden möten (10 %)
- Resten av tiden skriver hon ibland ny kod (10 %)



Debuggning

Oftast använder vi konsollen för att debugga

```
const Button = ({ handleClick, text }) => (  
  <button onClick={handleClick}>  
    {text}  
  </button>  
)
```

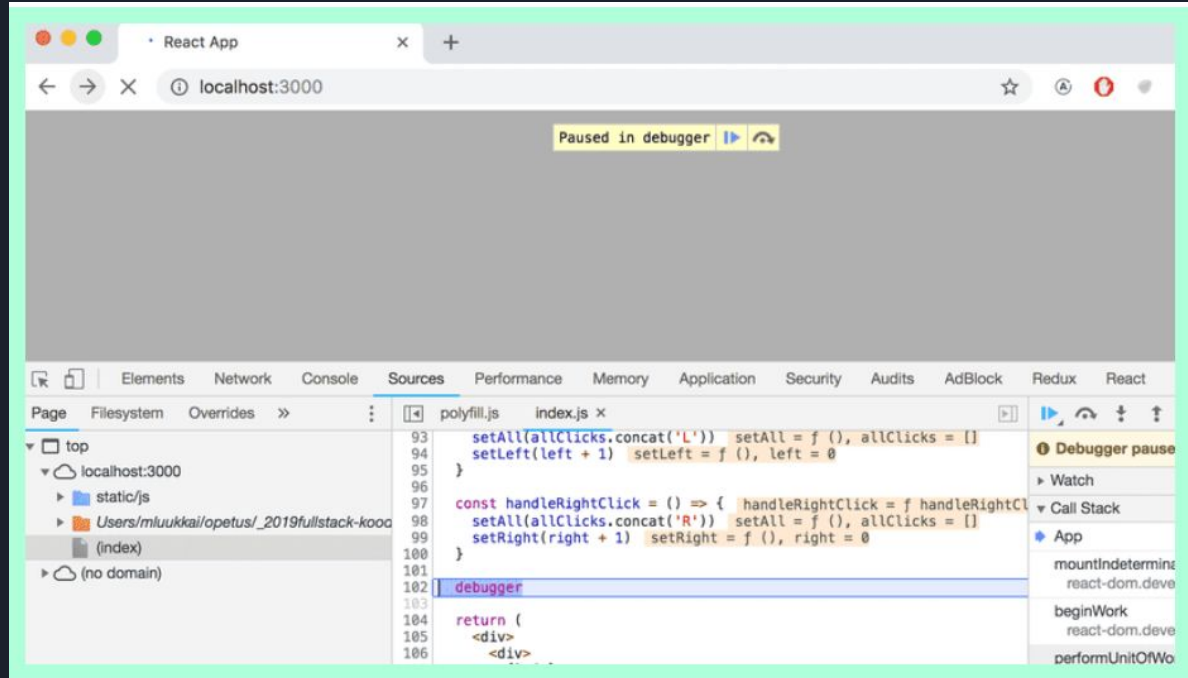


```
const Button = (props) => {  
  console.log(props)  
  const { handleClick, text } = props  
  return (  
    <button onClick={handleClick}>  
      {text}  
    </button>  
  )  
}
```

Debuggning

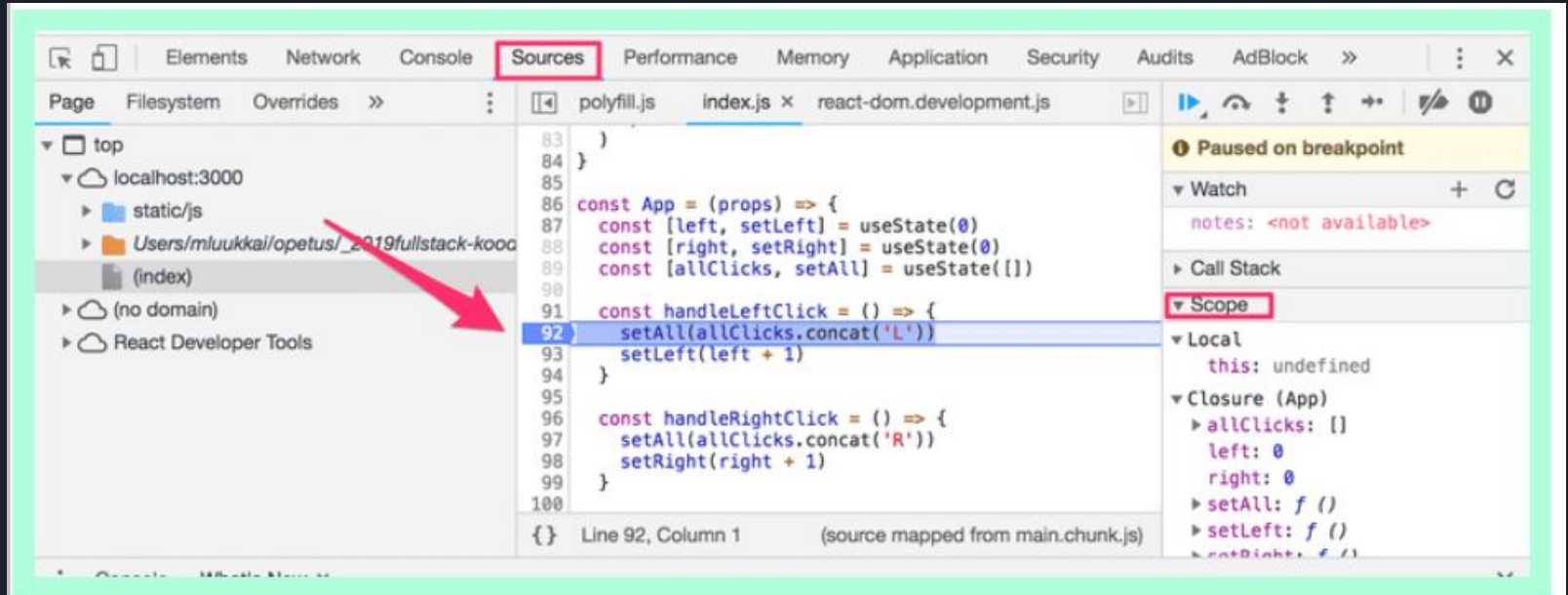
Ibland behöver vi debuggern

- debugger statement



Debugging

Med debuggern kan vi gå igenom koden rad för rad





Debuggning

Förutom konsol och debugger finns också en chrome extension

- React developer tools
(<https://chromewebstore.google.com/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi>)
- Installera och testa...



Hooks

useState som vi använder i våra komponenter kallas för en hook och är typiska i React

Det finns några regler att följa som gäller hooks

- Vi anropar inte hooks från loopar
- Vi anropar inte hooks från if statements
- Vi anropar inte funktioner som inte är komponenter



Hooks

- Summering

```
const App = () => {  
  // these are ok  
  const [age, setAge] = useState(0)  
  const [name, setName] = useState('Juha Tauriainen')  
  
  if ( age > 10 ) {  
    // this does not work!  
    const [foobar, setFoobar] = useState(null)  
  }  
  
  for ( let i = 0; i < age; i++ ) {  
    // also this is not good  
    const [rightWay, setRightWay] = useState(false)  
  }  
  
  const notGood = () => {  
    // and this is also illegal  
    const [x, setX] = useState(-1000)  
  }  
  
  return (  
    //...  
  )  
}
```



Repetition av event handling

Övning: Lägg till event handler som ökar value.

- definiera först som funktionen utanför
- definiera sedan funktionen inline

```
const App = () => {  
  const [value, setValue] = useState(10)  
  
  return (  
    <div>  
      {value}  
      <button>reset to zero</button>  
    </div>  
  )  
}
```



Funktioner som returnerar funktioner

Vi bör här repetera funktioner igen

- Skriv en funktion som tar 2 tal och returnerar summan av dem (testa)
- Skriv en funktion som tar en array och returnerar antalet element i arrayen (testa)
- Skriv en funktion som tar ett objekt och namnet på en property och returnerar värdet av den propertyen (testa)
- Skriv en funktion som tar en array och returnerar det sista elementet i arrayen om det finns något (testa)



Funktioner som returnerar funktioner

Vi kan ha funktioner som returnerar funktioner

```
const higherOrderFunc = () => {  
  return () => console.log("hej");  
}
```

```
const test = higherOrderFunc();  
test();
```



Funktioner som returnerar funktioner

Testa skriv en funktion som returnerar en funktion som summerar 2 tal

```
const higherOrderFunc = () => {  
  return () => console.log("hej");  
}
```

```
const test = higherOrderFunc();  
test();
```



Funktioner som returnerar funktioner

Vad är meningen med detta egentligen?

```
const hoc = (text) => {  
  return () => console.log(text);  
}
```

```
const test1 = hoc("hello");  
const test2 = hoc("bonjour");  
test1(); // hello  
test2(); // bonjour
```



Funktioner som returnerar funktioner

Övning: skriv en funktion `multiplier` som tar en siffra och returnerar en funktion som multiplicerar sin input med den siffran

```
const multiplier = (nr) => {  
  |   return // ???  
}
```

```
const test = multiplier(3);  
console.log(test(4)) // 12
```



Funktioner som returnerar funktioner

Vi kan använda detta koncept för att slippa upprepa oss minska kod

```
const App = () => {  
  const [value, setValue] = useState(10)  
  
  const hello = (who) => {  
    const handler = () => {  
      console.log('hello', who)  
    }  
    return handler  
  }  
  
  return (  
    <div>  
      {value}  
      <button onClick={hello('world')}>button</button>  
      <button onClick={hello('react')}>button</button>  
      <button onClick={hello('function')}>button</button>  
    </div>  
  )  
}
```

Funktioner som returnerar funktioner

```
const App = () => {  
  const [value, setValue] = useState(10)  
  
  const setToValue = (newValue) => () => {  
    console.log('value now', newValue) // print the new value to console  
    setValue(newValue)  
  }  
  
  return (  
    <div>  
      {value}  
      <button onClick={setToValue(1000)}>thousand</button>  
      <button onClick={setToValue(0)}>reset</button>  
      <button onClick={setToValue(value + 1)}>increment</button>  
    </div>  
  )  
}
```

Skicka vidare event handlers till barn komponenter


```
import React, {useState} from 'react'
import ReactDOM from 'react-dom'


const Button = (props) => (
  <button onClick={props.handleClick}>{props.text}</button>
)

const App = () => {
  const [value, setValue] = useState(10)

  const setToValue = (newValue) => {
    setValue(newValue)
  }

  return (
    <div>
      {value}
      <Button handleClick={() => setToValue(1000)} text="thousand" />
      <Button handleClick={() => setToValue(0)} text="reset" />
      <Button handleClick={() => setToValue(value + 1)} text="increment" />
    </div>
  )
}
```

Two red arrows originate from the `handleClick` and `text` props in the `Button` components within the `App` function's return statement. One arrow points to the `props.handleClick` parameter in the `Button` function definition, and the other points to the `props.text` parameter. This illustrates how props are passed from parent to child components.



Skicka vidare event handlers till barn komponenter

Övning

- Skapa en Button komponent som tar 2 props
 - event handler (vad ska hända när man trycker på knappen)
 - text (vad ska stå på knappen)
- I App
 - skapa ett state show (true/false)
 - rendera 2 Button komponenter och
 - 1 som visas när show = true (använd conditional rendering)