React.js – Introduktion

- React är ett **JavaScript-bibliotek** för att bygga användargränssnitt.
- Det använder en deklarativ och komponentbaserad arkitektur där hela UI:t delas upp i små, återanvändbara komponenter.
- Med React kan du skapa dynamiska Single Page Applications (SPA) som uppdaterar endast de delar av sidan som behöver förändras.

React-Dom och Virtual DOM

- React använder `react-dom` som är ett bibliotek som hanterar rendering av dina React-komponenter till webbläsarens riktiga DOM.
- Det översätter den virtuella DOM, som React arbetar med, till de faktiska DOM-noderna som webbläsaren visar.
- Det ser även till att uppdatera och avmontera komponenter på ett effektivt sätt.

Vad är Virtual DOM?

- Virtual DOM är en lättviktig, intern kopia av webbläsarens DOM som React använder för att optimera rendering.
- När state eller props ändras, jämför React den nya Virtual DOM med den tidigare versionen (diffing).
- Endast de delar av den riktiga DOM som behöver ändras uppdateras, vilket ger en snabbare och mer effektiv applikation.

JSX

- JSX står för **JavaScript XML** och låter oss skriva HTML i React på ett enklare sätt.
- JSX gör det lättare att skriva och använda HTML i React.
- JSX konverterar HTML-taggar till **React-element**.
- Gör React-kod mer läsbar.

Exempel: Med och utan JSX

Med JSX

```
const myElement = <h1>Rubrik</h1>;
```

Utan JSX

```
const myElement = React.createElement("h1", {}, "Utan JSX");
```

JSX och JavaScript-uttryck

Du kan använda JavaScript-uttryck inuti `{}` i JSX:

```
const jsx = "JSX";
const myElement = <h1>React med {jsx}</h1>;
```

Hantering av stora HTML-block

JSX tillåter HTML över flera rader genom att använda parenteser:

```
const myElement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);
```

JSX kräver ett toppnivåelement

Alla JSX-strukturer måste vara inneslutna i ett enda **toppelement**.

```
const myElement = (
  <div>
    <p>Jag är en paragraf.</p>
    <p>Jag är också en paragraf.</p>
  </div>
);
```

Alternativt med ett så kallat fragment (tom tagg):

```
const myElement = (
  <>
    <p>Jag är en paragraf.</p>
    <p>Jag är också en paragraf.</p>
  </>
);
```

Element måste stängas korrekt

JSX följer XML-regler, så **alla element måste stängas**.

Rätt:

```
const myElement = <input type="text" />;
```

Det här orsakar fel i JSX:

```
const myElement = <input type="text">; // Saknar stängning
```

`class` vs `className`

Eftersom `class` är ett reserverat ord i JavaScript, används **`className`** i JSX:

```
const myElement = <h1 className="myclass">Hello World</h1>;
```

Villkor i JSX

JSX stöder **if-satser**, men de måste placeras **utanför** JSX eller användas med sk **ternary operator**.

If-sats:

```
const x = 5;
let text = "Goodbye";
if (x < 10) {
  text = "Hello";
}
const myElement = <h1>{text}</h1>;
```

Ternary operator:

```
const x = 5;
const myElement = <h1>{x < 10 ? "Hello" : "Goodbye"}</h1>;
```

Skapa en React-komponent

En React-komponent är en vanlig JavaScript-funktion som returnerar JSX:

- arrow-function (mer kompakt och modernt, vi kör på det)

```
const ImageComponent = () => <img src="image.png" />;
```

- function

```
function ImageComponent() {
  return <img src="image.png" />;
}
```

- Man kan även använda klasser, men det anses förlegat

Filer

- Komponenter läggs typiskt i egna filer och med ändelsen `.jsx`
- När typescript används blir ändelsen istället: `.tsx`

```
//ImageComponent.jsx:
const ImageComponent = () => <img src="image.png" />;
export default ImageComponent;
```

Använda en komponent

```
//App.jsx:
import ImageComponent from "./components/ImageComponent";

const App = () => {
  return (
    <div>
      <ImageComponent />
    </div>
  );
};
```

Resultat i HTML:

```
<img src="image.png" />
```

Logik innan `return` i en komponent

En komponent kan innehålla logik innan den returnerar JSX:

```
const ImageComponent = () => {
  const imageUrl = "image.png";
  return <img src={imageUrl} />;
};
```

Blanda HTML och JavaScript

JavaScript-kod i JSX omsluts av `{}`:

```
const ImageText = () => {
  const animalType = "fågel";
  const speed = 20; // km/h

  return (
    <p>
      Bilden visar en {animalType}. Den kan flyga i {speed} km/h
    </p>
  );
};
```

Nästlade komponenter

Komponenter kan användas inuti andra komponenter:

```
const ImageWithText = () => (
  <div>
    <ImageComponent />
    <ImageText />
  </div>
);
```

React Fragment

En React-komponent kan endast returnera **ett** element. För att returnera flera element kan `<>...</>` användas:

```
<>
  <ImageComponent />
  <ImageText />
</>
```