



Lecture 02: Bundler in React :

React में ID, Class और Inline Style कैसे Add करें?

```
const element = React.createElement(  
  "h1",  
  {  
    id: "first",  
    className: "Saurav",  
    style: {  
      backgroundColor: "blue",  
      fontSize: "30px",  
      color: "pink",  
    },  
  },  
  "HelloCoder Army"  
);
```

Render in DOM:

```
const root = ReactDOM.createRoot(document.getElementById("root"));  
root.render(element);
```

? अगर हम दूसरा Element Render करें?

```
const element2 = React.createElement(  
  "h1",
```


```
{  
  id: "second",  
  className: "Saurav",  
  style: {  
    backgroundColor: "black",  
    fontSize: "30px",  
    color: "white",  
  },  
},  
"majaa aaya mujhe"  
);  
  
root.render(element2);
```

React क्या करता है?

जब नया element render होता है, तो React पुराना content **पूरी तरह से replace** कर देता है। इसलिए पहले वाला element DOM से हट जाता है।

Multiple Elements Render करने का तरीका

```
const parent = React.createElement("div", {}, element, element2);  
root.render(parent);
```

 इससे दोनों elements एक साथ render हो जाते हैं।

`createElement()` से Code Complex क्यों होता है?

अगर HTML nested है, तो हर element को अलग से `createElement` से wrap करना पड़ता है:

```
React.createElement("div", {},
```

```
React.createElement("h1", {},  
  React.createElement("p", {}, "Nested Text")  
)  
);
```

👉 ये code पढ़ने और लिखने दोनों में कठिन होता है।

🌟 JSX: A Better Way to Write UI

JSX आपको HTML जैसा syntax JS में लिखने की सुविधा देता है, जो Babel द्वारा React.createElement में convert हो जाता है।

```
const element = (  
  <div>  
    <h1>Hello React</h1>  
    <p>majaa aaya mujhe</p>  
  </div>  
);
```

✅ JSX code ज़्यादा readable होता है और maintain करना आसान होता है।

📁 Hosting React Project on Server (Without Bundler)

❌ Problems:

- 📄 Extra comments और spaces → File size बढ़ता है
 - 🐢 React और ReactDOM को CDN से load करना slow होता है
 - 📦 पूरा React/ReactDOM copy करना unnecessary load लाता है
-

🧠 Code Optimize कैसे करें?

- 🚫 Unused code और comments हटा दो
- 🎯 सिर्फ required parts ही load करो
- 📦 Use a **Bundler** — that's the best solution!

💡 What is a Bundler?

A bundler is a tool that takes all JavaScript, CSS, images, etc., and bundles them into optimized files for faster performance.

✅ Features:

- 📏 File size reduce करता है
- ⚡ Faster load time
- 🧼 Cleans code
- 🧠 Automatically resolves dependencies

🧰 Popular Bundlers

Tool	Use Case
Webpack	Most used, flexible
Parcel	Zero config, beginner-friendly
Vite	Fast dev server
ESBuild	Ultra-fast bundling

🔧 Getting Started with Parcel

◆ Step 1: `npm init`

```
npm init
```

- ये command एक `package.json` file बनाता है
- ये file आपकी project की metadata रखती है (name, version, dependencies आदि)

◆ `npm init -y` (Shortcut)

```
npm init -y
```






- बिना सवाल पूछे default `package.json` बना देता है



What is NPM?

NPM (Node Package Manager) एक online library है जहाँ JavaScript packages और tools जैसे React, Parcel, Vite, आदि मिलते हैं।

📖 NPM में क्या-क्या मिलता है?

-  React
-  ReactDOM
-  Parcel
-  TypeScript
-  Thousands of other packages

🌐 Official website: <https://www.npmjs.com>



Installing Parcel

```
npm install parcel
```

💡 Parcel install होने पर:

- एक नया folder `node_modules` बनता है
 - इसके अंदर **Parcel** और इसके **dependencies** install हो जाते हैं
-

? Parcel के साथ अलग-अलग packages क्यों आते हैं?

👉 Parcel खुद एक independent tool नहीं है, बल्कि:

- ये internally दूसरे existing packages (like Babel, core modules, compilers) को use करता है
- इसलिए वो सारे packages `node_modules` में install हो जाते हैं

✂ यह system को modular बनाता है और future updates आसान करता है

🔧 React Install कैसे करें?

```
npm install react
```

```
npm install react-dom
```

🔧 इसके बाद:

- React और ReactDOM `node_modules` और `package.json` में दिखेंगे
-

📄 Files Overview:

1. `package.json`

- Project से जुड़ी meta-information और dependency list होती है

2. `package-lock.json`

- हर installed package का **exact version** record करता है
 - Ensures consistency across environments
-


Versioning in NPM

Version Format	Meaning
18.2.3	Major.Minor.Patch
^18.2.3	Accept Minor & Patch
~18.2.3	Accept only Patch
18.2.3	Exact version (locked)

Example:


- **^18.2.3** → Can update to 18.2.4, 18.3.0 but not 19.0.0
- **~18.2.3** → Only patch updates like 18.2.4
- No symbol → Exact lock

GitHub Best Practice

 **node_modules** GitHub पर upload ना करें — यह बहुत heavy होता है।

Upload only:

1. **index.html**
2. **script.js / App.js**
3. **package.json**
4. **package-lock.json**


 फिर **npm install** चलाकर सभी dependencies दुबारा install कर सकते हैं।

What Happens When You Run `npm install`?

- `package.json` & `package-lock.json` पढ़ता है
 - Automatically missing packages को install करता है
 - Exact versions restore करता है
-

क्या होगा अगर `package-lock.json` delete कर दो?

👉 तब `npm install` सिर्फ `package.json` देखेगा

 और वो packages का latest compatible version install करेगा

इससे versions थोड़ा बदल सकते हैं।

Final Summary (One Page View)

Topic	Notes
<code>React.createElement()</code>	Manual way to create elements
<code>JSX</code>	Clean and readable UI syntax
<code>render()</code>	Only shows latest content
<code>Bundler</code>	Tool to optimize, compress, and bundle
<code>Parcel</code>	Easy bundler with zero-config
<code>npm init</code>	Creates package.json

<code>npm install parcel</code>	Installs parcel in <code>node_modules</code>
<code>React install</code>	<code>npm install react react-dom</code>
<code>package.json</code>	Contains dependency list
<code>package-lock.json</code>	Locks exact version
<code>GitHub practice</code>	Avoid uploading <code>node_modules</code>
<code>npm install</code>	Installs all required dependencies
<code>Versioning</code>	Follows Semantic Versioning (Major.Minor.Patch)

BONUS TIP:

If you're working on a team or uploading to GitHub:

- Add `node_modules/` to `.gitignore`
- Use `npm install` on new systems