



Lecture 3: Deploy Your React App



Method 1: बिना Optimization के Direct Production में Deploy करना

📁 Production में सीधे सभी फाइल्स डाल देना — बिना किसी optimization के:

`index.html`

`app.js`

`react`

`react-dom`

अन्य dependencies

◆ इस method में:

- कोई भी code optimization नहीं होता।
- पूरा raw code production में चला जाता है।
- Code size ज़्यादा होता है, loading slow होती है।

⚠ **Note:**

यह तरीका सिर्फ practice/testing के लिए use करें। Production में recommended नहीं है।



Method 2: Bundler से Optimized Deployment



Bundler क्या करता है?

- ☒ Code को **Optimize** करता है
 - ☒ सभी files (HTML, JS, CSS, Images) को **Bundle** करता है
 - ☒ Code size को **कम** करता है
 - ☒ Machine/browser के लिए code को समझना **आसान** हो जाता है
-

Bundler Output: **dist/** Folder

Bundler (जैसे Parcel) build करने के बाद एक folder generate करता है:

dist/

|— index.html

|— index.js

|— index.js.map ❌ (इस file को upload नहीं करते)

इससे फायदा:

- Code का size कम हो जाता है
- Execution fast होता है
- सिर्फ जरूरत की फाइलें ही जाती हैं Production में

Parcel से React App को Deploy करना

Step 1: React और ReactDOM को Import करें

```
import React from "react";
```

```
import ReactDOM from "react-dom/client";
```

Step 2: HTML में Script Tag को Type = Module देना होगा

```
<script type="module" src="index.js"></script>
```

Type="module" क्यों देते हैं?

- Modern JS features (**import**, **export**) use करने के लिए
 - Browser को बताता है कि यह JS code एक module है
 - Scope isolate होता है (Global variables टकराते नहीं)
-



Parcel Commands



Development Server Run करने के लिए:

```
npx parcel index.html
```



इससे local server चालू होगा, changes live दिखेंगे।



Production Build Generate करने के लिए:

```
npx parcel build index.html
```



इससे एक optimized `dist/` folder create होगा जिसमें production-ready files होंगी:

```
index.html
```

```
index.js
```

```
index.js.map (optional - remove for production)
```



Production में क्या Upload करना है?



Upload करें:



```
dist/index.html
```



```
dist/index.js
```



Upload नहीं करें:



```
index.js.map
```



क्यों नहीं?

Map file से कोई भी developer आपके original JS code को वापस trace कर सकता है। यह Security Risk होता है।

Parcel Cache और Dist Folder Explained

Parcel Cache:

- जब आप build करते हैं, Parcel कुछ data को cache करता है ताकि अगली बार faster build हो।

Dist Folder:

- Optimized और bundled production-ready code यहीं save होता है।

Production के फायदे

- ✓ Code Fast Execute होता है
- ✓ कम Space लेता है
- ✓ सिर्फ ज़रूरी files जाती हैं
- ✓ Developer Tools में readable नहीं होता (secured)
- ✓ SEO और Performance के लिए बेहतर होता है

Final Note:

Production में सिर्फ वही files दें जो ज़रूरी हों।


`index.js.map` को हटाना ना भूलें ✓

`dist/` folder ही deploy करें 🚀

Deploy Project using dist Folder (With Netlify Hosting) :

Step 1: Create Dist Folder

 `npx parcel build index.html`

 यह command एक `dist/` folder बनाएगा जिसमें आपकी **optimized production-ready files** होंगी:

`dist/`

|— `index.html`

|— `index.js`


|— `index.js.map` ✗ (इस file को remove कर देना चाहिए)

Step 2: Deploy on Netlify

✅ Netlify एक **hosting platform** है जहाँ आप अपने React project को deploy कर सकते हैं।

Steps:

1. Netlify पर जाएं  <https://www.netlify.com>
2. एक free account create करें
3. "Add new site" → "Deploy manually"
4. `dist/` folder को **drag & drop** करें

 अब आपको एक **live website link** मिल जाएगा जहाँ आपका project host हो चुका होगा!

JSX (JavaScript XML)

JSX क्या है?

JSX का मतलब है – **HTML-like code written inside React JavaScript code.**

- React directly HTML या JS को नहीं समझता
 - JSX का काम है: HTML+JS को **React readable format** में बदलना
-

JSX कैसे काम करता है?

 JSX Code Flow:

JSX Code



Babel (Parcel के अंदर आता है)



`React.createElement(...)`



React



React Element (JS Object)



Render



HTML Element on Browser

🧠 Browser JSX को directly नहीं समझता – इसलिए Babel JSX को `React.createElement` में convert करता है

🔗 JSX vs `React.createElement()`

पहले ऐसा code लिखा जाता था:

```
React.createElement("h1", {}, "Hello React");
```

अब JSX ने इस process को आसान बना दिया है:

```
<h1>Hello React</h1>
```

JSX लिखना आसान है और HTML जैसा दिखता है, लेकिन internally ये JS object में convert होता है।

JSX Code Example (Fixed)

```
const obj = {
```

```
  name: "Harshal",
```

```
  age: 20
```

```
};
```

```
const New = (
```

```
  <>
```

```
    <h1>Hello Bhai</h1>
```

```
    <h1>Hello {obj.name}</h1>

  </>

);

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

root.render(New);
```

✅ JSX में हम variables, objects को `{ }` में wrap करके use कर सकते हैं।

React में Styling (Inline CSS)

React में style एक **JavaScript Object** के रूप में दिया जाता है।

```
const obj2 = {

  backgroundColor: "black",

  color: "pink"

};

const element = (

  <h3 style={obj2}>Hi React</h3>

);

// या directly object pass कर सकते हैं:

const element2 = (

  <h3 style={{ backgroundColor: "black", color: "pink" }}>

    Hi React

  </h3>

);

root.render(element2);
```

🧠 Style object में **camelCase** में properties होती हैं (`background-color` ❌, `backgroundColor` ✅)

Dynamic Attributes:

React में आप variables को directly JSX में embed कर सकते हैं:

Attributes can take **variables** or **expressions** too.

```
const money = 123;  
const element = <h2 data-money={money}>Balance Info</h2>;
```

React Components

1 Function-based Components (✅ Recommended)

```
function Greet() {  
  return <h1>Hi React</h1>;  
}
```

```
function Meet() {  
  return <h2>Aur Bhai Kaise Ho</h2>;  
}
```

```
const element1 = Greet();  
const element2 = Meet();
```

```
const newElement = (  
  <p>  
    {element1} {element2}  
  </p>  
>);
```



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

root.render(newElement);
```

- ✓ Function-based components JSX return करते हैं
- ✓ इन्हें आप direct render भी कर सकते हैं:

```
root.render(<Meet />);
```

2 Class-based Components (✗ Not Used Often Now)

Modern React में Functional Components + Hooks का ही ज़्यादा use होता है

Key Takeaways

- ✓ JSX एक HTML-like syntax है जो React में JavaScript के साथ काम करता है
 - ✓ Babel JSX को React-readable JS Object (`React.createElement`) में convert करता है
 - ✓ Styling के लिए React inline object-based CSS use करता है
 - ✓ `dist/` folder को Netlify पर host करके React App को deploy किया जा सकता है
 - ✓ `index.js.map` file को **production** में नहीं डालना चाहिए
-

Bonus Tip:

JSX में आप comments ऐसे लिख सकते हैं:

```
{/* This is a comment in JSX */}
```

✓ Ready-to-Use Deployment Summary

Create dist Folder:

```
npx parcel build index.html
```

Host on Netlify:

- Go to <https://www.netlify.com>
- Create account
- Drag `dist/` folder and deploy



Deployment के दो तरीके

1. Without Optimization

- सारी files (HTML, JS, React, etc.) सीधे production में डालना
- Code भारी होता है, slow load होता है
- Production के लिए suitable नहीं

2. With Optimization (Parcel से)

- `npx parcel build` से `dist/` folder बनता है
- Files optimize हो जाती हैं (छोटी + fast)
- सिर्फ ज़रूरी files (`index.html`, `index.js`) upload करते हैं
- `.map` file delete कर देनी चाहिए (security के लिए)



Hosting with Netlify

- Netlify एक free hosting platform है
- Account बनाओ, `dist/` folder drag & drop करो
- आपको एक live website link मिल जाएगा



JSX (JavaScript XML)

- JSX: HTML जैसी syntax जो React के JS code में लिखी जाती है
- React सीधे HTML/JS नहीं समझता, इसलिए JSX ज़रूरी है
- Parcel के अंदर Babel होता है जो JSX को JS object (React element) में बदलता है
- Browser JSX नहीं समझता — Babel इसे translate करता है

React में Styling

- CSS को JS object की तरह लिखा जाता है
 - Properties camelCase में होती हैं
 - React inline styling को support करता है
-

React Components

- React में दो तरह के components होते हैं:
 - **Function-based Components** (modern way)
 - **Class-based Components** (पुराना तरीका, अब rarely used)
 - Components reusable होते हैं और JSX return करते हैं
 - Function components को directly render किया जा सकता है
-

Parcel Cache & dist Folder

- Parcel cache से faster builds होते हैं
 - **dist/** folder में optimized, production-ready files होती हैं
 - **.map** file remove करनी चाहिए क्योंकि इससे source code trace किया जा सकता है
-

Final Deployment Checklist

- Parcel से build करो
- सिर्फ optimized files (**index.html**, **index.js**) रखें
- **.map** file delete करो
- Netlify पर upload करके live website बनाओ