

# Namaste React Notes

## Lecture 1- Inception

### Hello World Program by using HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
    <h1>Hello World using HTML</h1>
  </div>
</body>

</html>
```

### Hello World Program by using Javascript

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <script>
    const heading = document.createElement("h1");
    heading.innerHTML = "Hello World from JavaScript"
    const root = document.getElementById("root")
    root.appendChild(heading)
  </script>
</body>

</html>
```

## Injecting React into Html file using CDN(Content Delivery Network)

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
</body>

</html>
```

## Hello World Program using React

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
  <script>
    const heading = React.createElement("h1", {}, "Hello World from React")
    const root = ReactDOM.createRoot(document.getElementById("root"))
    root.render(heading)
  </script>
</body>

</html>
```

## Separating the JavaScript Code, CSS and HTML into separate files

### index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="index.css">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
  <script src="App.js"></script>
</body>

</html>
```

### App.js

```
const heading = React.createElement(
  "h1",
  { id: "heading", "data-testid": "heading", testid: "heading" },
  "Hello World from React"
);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(heading);
```

### index.css

```
#root {
  background-color: aqua;
}
```

### Output:

**Hello World from React**

---

### If we console.log(heading) in App.js

```
const heading = React.createElement(
  "h1",
  { id: "heading", "data-testid": "heading", testid: "heading" },
  "Hello World from React"
);
console.log(heading)
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(heading);
```

### Output:

```
▼ Object i
  $$typeof: Symbol(react.element)
  key: null
  ▼ props:
    children: "Hello World from React"
    data-testid: "heading"
    id: "heading"
    testid: "heading"
    ► [[Prototype]]: Object
  ref: null
  type: "h1"
  _owner: null
  ► _store: {validated: false}
  _self: null
  _source: null
  ► [[Prototype]]: Object
>
```

### Creating Nested Elements in React

#### Trying to create

```
<div id="parent">
  <div id="child">
    <h1 id="inner-child">Hello World!</h1>
  </div>
</div>
```

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement(
    "div",
    { id: "child" },
    React.createElement("h1", { id: "inner-child" }, "Hello World!")
  )
);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(parent);
```

```
▼ {$$typeof: Symbol(react.element), type: 'div', key: null, ref: null, props: {...}, ...} ⓘ
  $$typeof: Symbol(react.element)
  key: null
  ▼ props:
    ▼ children:
      $$typeof: Symbol(react.element)
      key: null
      ▼ props:
        ► children: {$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {...}, ...}
        id: "child"
        ► [[Prototype]]: Object
        ref: null
        type: "div"
        _owner: null
        ► _store: {validated: true}
        _self: null
        _source: null
        ► [[Prototype]]: Object
        id: "parent"
        ► [[Prototype]]: Object
        ref: null
        type: "div"
        _owner: null
        ► _store: {validated: false}
        _self: null
        _source: null
        ► [[Prototype]]: Object
    > |
```

Notice the children in the above example

### Creating Siblings in React

```
<div id="parent">
  <div id="child">
    <h1 id="inner-child1">H1 Tag</h1>
    <h2 id="inner-child2">H2 Tag</h2>
  </div>
</div>
```

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement(
    "div",
    { id: "child" },
    [React.createElement("h1", { id: "inner-child1", key:"1" }, "H1 Tag"),
     React.createElement("h2", { id: "inner-child2", key:"2" }, "H2 Tag")]
  )
);
console.log(parent);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(parent);
```

## Siblings are passed inside an Array

### H1 Tag

### H2 Tag

---

```
▼ {$$typeof: Symbol(react.element), type: 'div', key: null, ref: null, props: {...}} ⓘ
  $$typeof: Symbol(react.element)
  key: null
  ▼ props:
    ▼ children:
      $$typeof: Symbol(react.element)
      key: null
      ▼ props:
        ▼ children: Array(2)
          ► 0: {$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {...}, ...}
          ► 1: {$$typeof: Symbol(react.element), type: 'h2', key: null, ref: null, props: {...}, ...}
          length: 2
          ► [[Prototype]]: Array(0)
          id: "child"
          ► [[Prototype]]: Object
          ref: null
          type: "div"
          _owner: null
          ► _store: {validated: true}
          _self: null
          _source: null
          ► [[Prototype]]: Object
          id: "parent"
          ► [[Prototype]]: Object
          ref: null
          type: "div"
          _owner: null
          ► _store: {validated: false}
          _self: null
          _source: null
          ► [[Prototype]]: Object
    ►
  >
```

**It becomes extremely complex to write React Code like this. So, there came the need for JSX (HTML Like syntax inside Javascript)**

## Lecture-02 Igniting Our App

**npm** is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js and is included as a recommended feature in the Node.js installer.

### Project Scaffolding Steps:

#### 1. **npm init (Creates package.json)**

```
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (namaste-react-02th-feb)
version: (1.0.0)
description:
entry point: (App.js)
test command:
git repository: (https://github.com/AkashDR/Namaste-React.git)
keywords: namaste react
author: Akash D R
license: (ISC)
About to write to C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb\package.json:

{
  "name": "namaste-react-02th-feb",
  "version": "1.0.0",
  "description": "Namaste React Course",
  "main": "App.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/AkashDR/Namaste-React.git"
  },
  "keywords": [
    "namaste",
    "react"
  ],
  "author": "Akash D R",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/AkashDR/Namaste-React/issues"
  },
  "homepage": "https://github.com/AkashDR/Namaste-React#readme"
}

Is this OK? (yes)
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> █
```

```

() package.json > ...
1  {
2    "name": "namaste-react-02th-feb",
3    "version": "1.0.0",
4    "description": "Namaste React Course",
5    "main": "App.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "repository": {
10     "type": "git",
11     "url": "git+https://github.com/AkashDR/Namaste-React.git"
12   },
13   "keywords": [
14     "namaste",
15     "react"
16   ],
17   "author": "Akash D R",
18   "license": "ISC",
19   "bugs": {
20     "url": "https://github.com/AkashDR/Namaste-React/issues"
21   },
22   "homepage": "https://github.com/AkashDR/Namaste-React#readme"
23 }
24

```

Package.json is configuration for npm. It contains details of all the packages/libraries the project has like version, package Name etc.

## 2. npm install -D parcel (Installs parcel as Developer Dependency)

The above command adds **package-lock.json** file, node modules folder, parcel package and other dependency package of the parcel

A bundler helps in creating production ready apps. Example of bundlers include webpack, vite, parcel etc.

### Difference between Dev Dependency and Normal Dependency:

<https://www.geeksforgeeks.org/difference-between-dependencies-devdependencies-and-peerdependencies/>

While installing parcel or any bundler, if you get this error

**npm ERR! 404 Not Found - GET https://registry.npmjs.org/create-react-app/webpack**

then we have to set the registry. Only if we set the registry, then npm would download the packages from that registry. Steps to resolve the issue are listed below



## Missing repository registry

```
$ npm set registry https://registry.npmjs.org/
```

## Clean cache

```
$ npm cache clean  
$ npm rebuild
```

---

### Difference between Caret and Tilde

Caret(^) consider only patch and minor version update automatically. Caret(^) is less safer than Tilde(~) for production app. because here minor feature will also update automatically .

<https://www.geeksforgeeks.org/difference-between-tilde-and-caret-in-package-json/>

<https://www.linkedin.com/pulse/difference-bw-tilde-notation-caret-alok-tiwari/>

### 3. cat > .gitignore

Create .gitignore file by using above command and add all the files which should not be committed

```
❖ .gitignore  
1  # Excluding following files/item from commit  
2  /node_modules  
3  .parcel-cache/  
4  dist/
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
no changes added to commit (use "git add" and/or "git commit -a")  
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> git status  
On branch main  
Your branch is up to date with 'origin/main'.
```

```
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
        modified:   App.js  
        modified:   Namaste React Notes.docx  
        modified:   Namaste React Notes.pdf  
        modified:   index.html
```

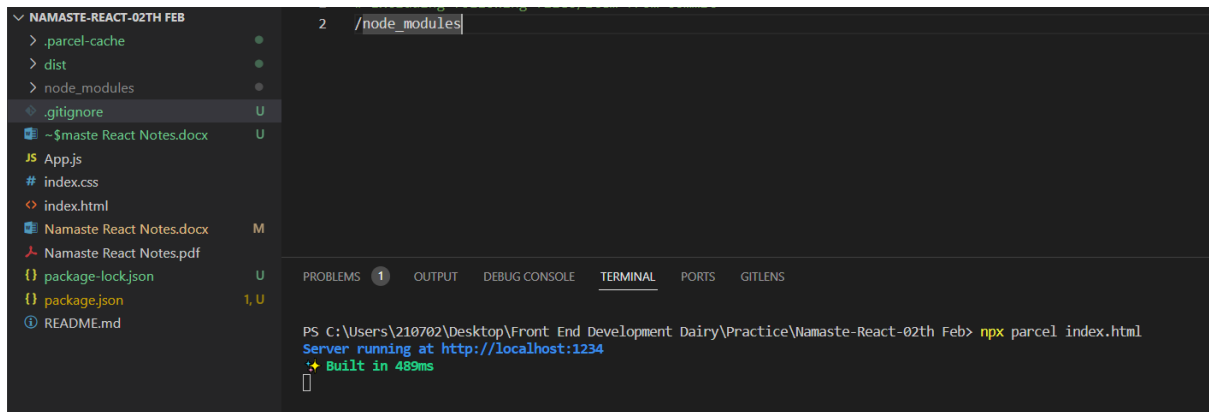
```
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
        .gitignore  
        package-lock.json  
        package.json  
        ~$maste React Notes.docx
```

```
no changes added to commit (use "git add" and/or "git commit -a")  
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> █
```

## How to create .gitignore file

[https://www.youtube.com/watch?v=ErJyWO8TGoM&ab\\_channel=codebasics](https://www.youtube.com/watch?v=ErJyWO8TGoM&ab_channel=codebasics)

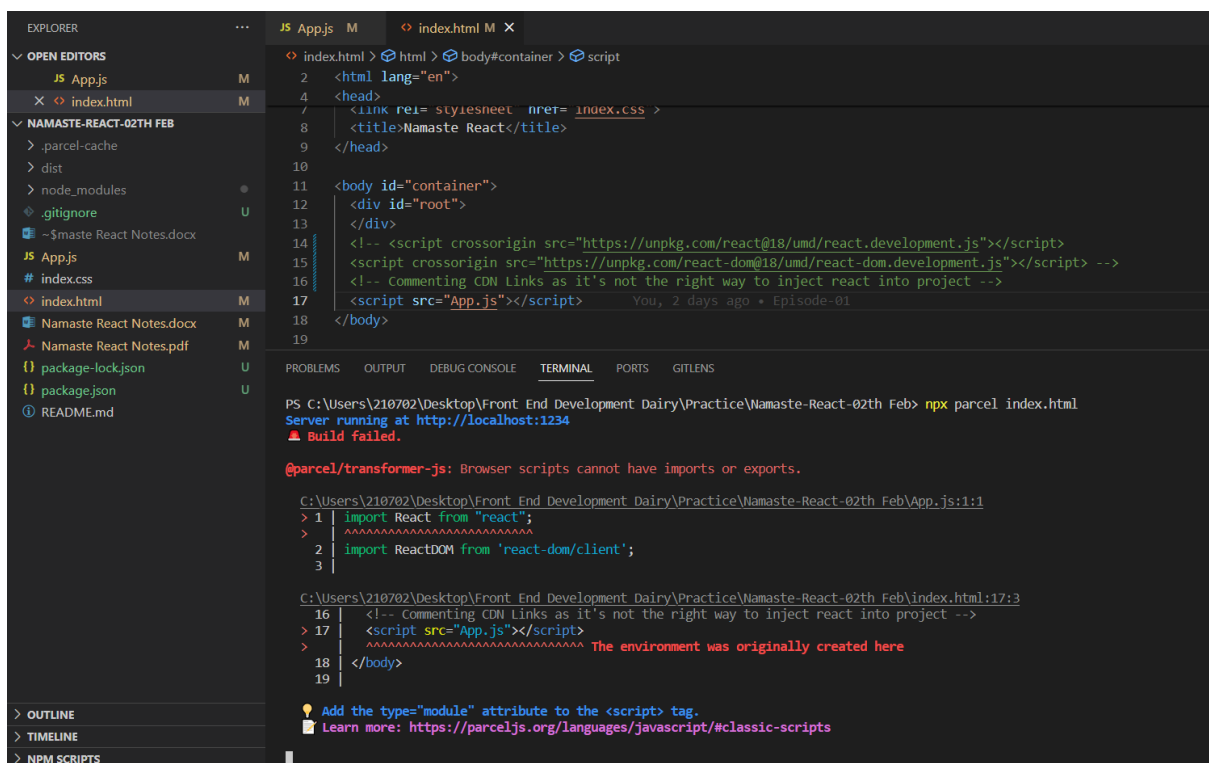
### 4. `npx create index.html`



The screenshot shows the VS Code interface with a file explorer on the left listing files like `.parcel-cache`, `dist`, `node_modules`, `.gitignore`, and `index.html`. The terminal at the bottom displays the command `npx parcel index.html` and its output: `Server running at http://localhost:1234` and `Built in 489ms`.

Start/Index the application using the above command. Notice that it has created `.parcel-cache` and `dist` folder inside the project. Project starts on port 1234

**npx** is used to execute the package.

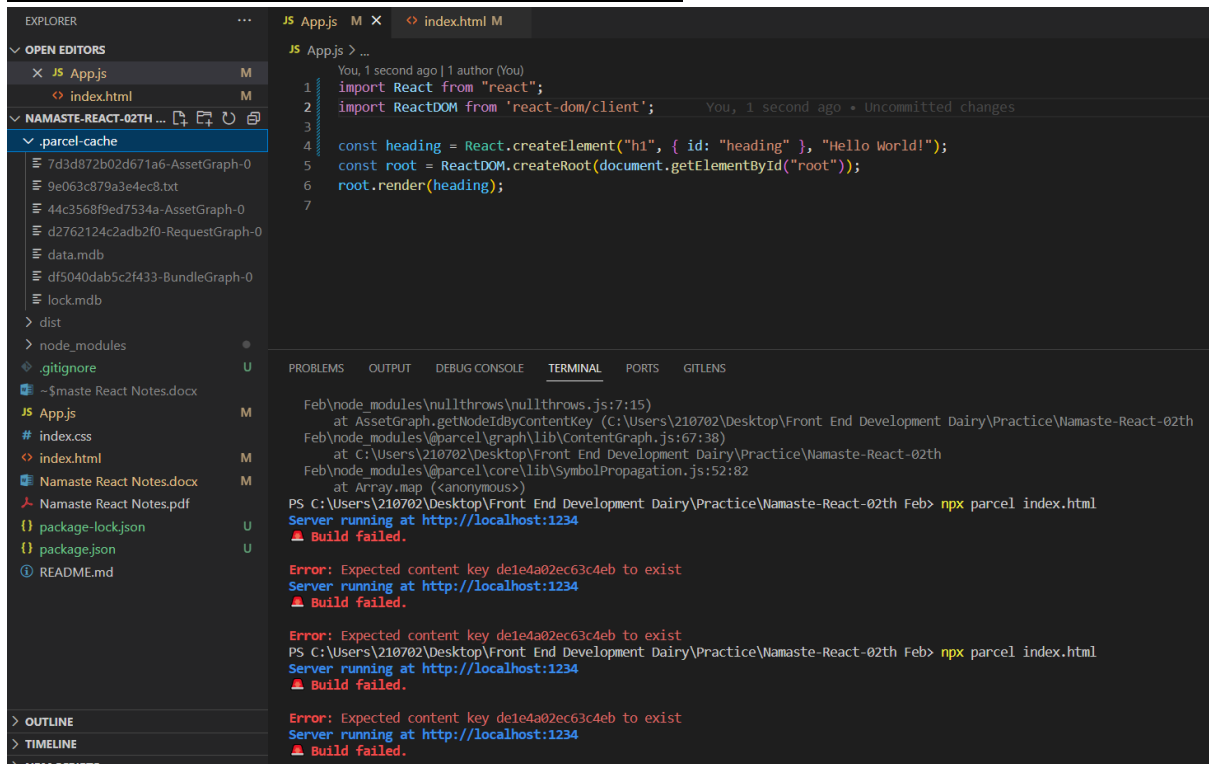


The screenshot shows the VS Code interface with the `index.html` file open. The terminal at the bottom displays the command `npx parcel index.html` and its output, which includes an error: `@parcel/transformer-js: Browser scripts cannot have imports or exports.` The error points to the `App.js` file, which contains `import React from 'react';` and `import ReactDOM from 'react-dom/client';`. The terminal also shows the `index.html` file content, which includes `<script src='App.js'></script>`. The error message suggests adding the `type="module"` attribute to the `<script>` tag.

Error: Browser scripts can't have imports or exports. Solution would be mention attribute type as module in index.html. As `App.js` is not a normal file it is a module.

```
JS App.js M    <> index.html M X
<> index.html > html
2  <html lang="en">
4  <head>
7  <link rel="stylesheet" href="index.css" >
8  <title>Namaste React</title>
9  </head>
10
11 <body id="container">
12   <div id="root">
13   </div>
14   <!-- <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
15   <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script> -->
16   <!-- Commenting CDN Links as it's not the right way to inject react into project -->
17   <script type="module" src="App.js"></script>
18 </body>
19
20 </html>    You, 2 days ago • Episode-01
```

## Error: Expected content key de1e4a02ec63c4eb to exist



```
EXPLORER    ...    JS App.js M X    <> index.html M
v OPEN EDITORS
  X JS App.js M
  <> index.html M
  v NAMASTE-REACT-02TH ...
    v .parcel-cache
      7d3d872b02d671a6-AssetGraph-0
      9e063c879a3e4ec8.txt
      44c3568f9ed7534a-AssetGraph-0
      d2762124c2adb2f0-RequestGraph-0
      data.mdb
      df5040dab5c2f433-BundleGraph-0
      lock.mdb
    > dist
    > node_modules
    > .gitignore
    > ~$maste React Notes.docx
  JS App.js M
  # index.css
  <> index.html M
  Namaste React Notes.docx M
  Namaste React Notes.pdf
  {} package-lock.json U
  {} package.json U
  @ README.md

> OUTLINE
> TIMELINE
> NAMASTE-REACT-02TH ...

JS App.js > ...
You, 1 second ago | 1 author (You)
1  import React from "react";
2  import ReactDOM from 'react-dom/client';
3
4  const heading = React.createElement("h1", { id: "heading" }, "Hello World!");
5  const root = ReactDOM.createRoot(document.getElementById("root"));
6  root.render(heading);
7

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS
Feb\node_modules\nullthrows\nullthrows.js:7:15)
  at AssetGraph.getNodeByIdByContentKey (C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th
Feb\node_modules\parcel\graph\lib\ContentGraph.js:67:38)
  at C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th
Feb\node_modules\parcel\core\lib\SymbolPropagation.js:52:82
  at Array.map (<anonymous>)
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> npx parcel index.html
Server running at http://localhost:1234
Build failed.
Error: Expected content key de1e4a02ec63c4eb to exist
Server running at http://localhost:1234
Build failed.
Error: Expected content key de1e4a02ec63c4eb to exist
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> npx parcel index.html
Server running at http://localhost:1234
Build failed.
Error: Expected content key de1e4a02ec63c4eb to exist
Server running at http://localhost:1234
Build failed.
```

If you are using parcel then try to delete ".parcel-cache" folder. And then Rerun the build to solve the above issue

## What does Parcel do?

Read about these concepts in this page (1<sup>st</sup> page itself).

<https://parceljs.org/>

```
# Parcel
- Dev Build
- Local Server
- HMR = Hot Module Replacement
- File Watching Algorithm - written in C++
- Caching - Faster Builds
- Image Optimization
- Minification
- Bundling
- Compress
- Consistent Hashing
- Code Splitting
- Differential Bundling - support older browsers
- Diagnostic
- Error Handling
- HTTPs
- Tree Shaking - remove unused code
- Different dev and prod bundles
```

**Read about few of the definitions from below link**

<https://legacy.reactjs.org/docs/code-splitting.html>

Differential bundling is the concept of sending various copies of your code to different targets and letting the browser decide which one to download

**How to create dev build?**

**Code:** `npx parcel index.html` ("Notice the keyword build missing")

**How to create production ready build?**

**Code:** `npx parcel build index.html`

When you run this, the production build gets created in dist folder after all the optimization (Done by parcel).

```
PS C:\Users\210702\Desktop\Front End Development Dairy\Practice\Namaste-React-02th Feb> npx parcel build index.html
🚀 Built in 2.73s

dist\index.html          366 B    1.18s
dist\index.a28d1165.css   98 B     70ms
dist\index.b0ff1e7e.js  138.77 KB  1.52s
```

**Error: @parcel/namer-default: Target "main" declares an output file path of "App.js" which does not match the compiled bundle type "html".**

```
package.json > {} repository
1  {
2    "name": "namaste-react",
3    "version": "1.0.0",
4    "description": "This is Namaste React by Akshay Saini",
5    "main": "App.js",
6    "scripts": {
7      "test": "jest"
8    },
9    "repository": {
10     "type": "git",
11     "url": "git+https://github.com/namastedev/namaste-react.git"
12   },
13   "keywords": [],
14   "author": "Akshay Saini",
15   "license": "ISC",
16   "bugs": {
17     "url": "https://github.com/namastedev/namaste-react/issues"
18   },
19   "homepage": "https://github.com/namastedev/namaste-react#readme",
20   "devDependencies": {
21     "parcel": "^2.8.3",
22     "process": "^0.11.10"
23   },
24 }
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** GITLENS COMMENTS

```
4 | "description": "This is Namaste React by Akshay Saini",
> 5 | "main": "App.js",
> |      ^^^^^^^^^ Did you mean "App.html"?
6 | "scripts": {
7 |   "test": "jest"
8 | }
9 |
```

💡 Try changing the file extension of "main" in package.json.

To solve this error remove "main" in package.json

```
{ } package.json > ...
1  {
2    "name": "namaste-react",
3    "version": "1.0.0",
4    "description": "Namaste React Course",
5    "main": "App.js",
6    "scripts": {
```

Like this

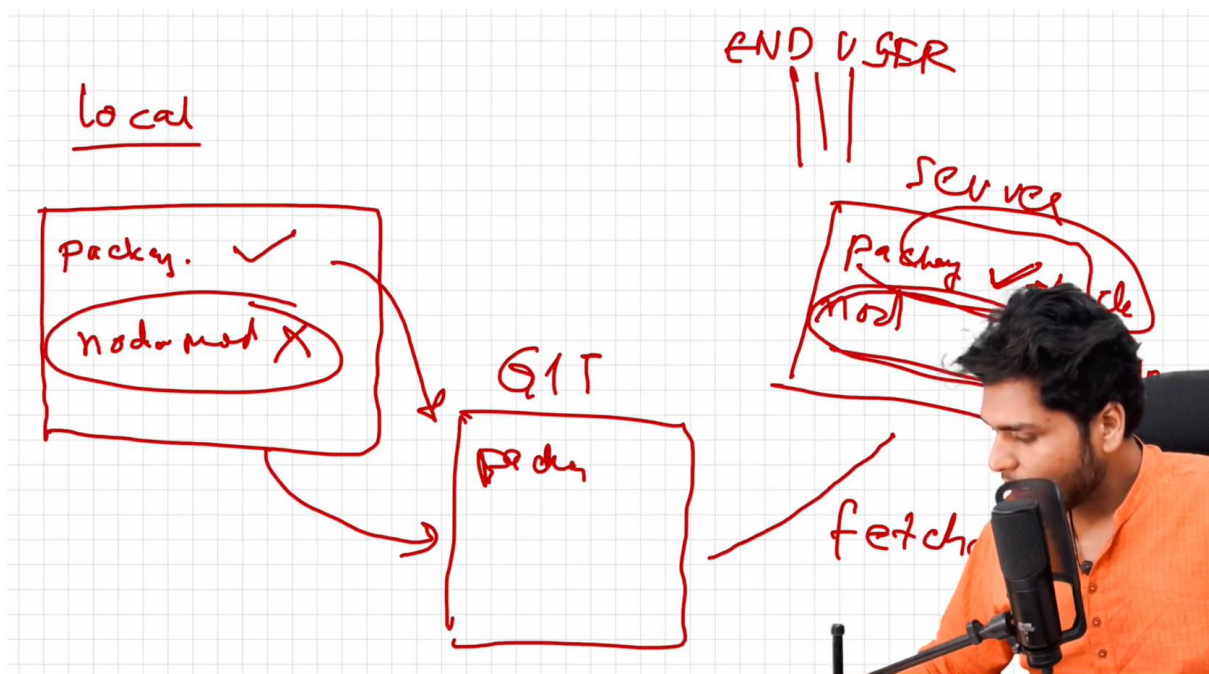
```

{} package.json > ...
1  {
2    "name": "namaste-react",
3    "version": "1.0.0",
4    "description": "Namaste React Course",
   ▶ Debug
5    "scripts": {
6      "test": "echo \\\"Error: no test specified\\\" && ex

```

### Flow of Application:

Server fetches package.json & package-lock.json from Git and executes the command to create production build. And hosts that build to the end user



### How to make your app compatible to older versions of browser?

Use browserlist package for achieving compatibility.

<https://browserslist.dev/?q=bGFzdCAyIHZlcuNpb25z>

<https://github.com/browserslist/browserslist#query-composition>

Configuration is as follows:

{ } package.json 1, U X

{ } package.json > ...

```
1  {
2    "name": "namaste-react",
3    "version": "1.0.0",
4    "description": "Namaste React Course",
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "repository": {
9      "type": "git",
10     "url": "git+https://github.com/AkashDR/Namaste-React.git"
11   },
12   "keywords": [],
13   "author": "Akash D R",
14   "license": "ISC",
15   "bugs": {
16     "url": "https://github.com/AkashDR/Namaste-React/issues"
17   },
18   "homepage": "https://github.com/AkashDR/Namaste-React#readme",
19   "devDependencies": {
20     "parcel": "^2.11.0",
21     "process": "^0.11.10"
22   },
23   "dependencies": {
24     "react": "^18.2.0",
25     "react-dom": "^18.2.0"
26   },
27   "browserlist": [
28     "last 10 Chrome version",
29     "last 10 Firefox version",
30     "last 10 versions"
31   ]
32 }
33
```

## Promises: How to extract data from Promises?

```
const cart = ["Shoes", "Pants", "Watches"];

function createOrder(cart, proceedToPayment) {
  console.log("Order Created", cart);
  console.log("Lets Wait");
  setTimeout(() => {
    proceedToPayment();
  }, 5000);
}

function proceedToPayment() {
  console.log("Proceeded to Payment");
}

createOrder(cart, proceedToPayment);

const URL1 = "https://api.github.com/users/mojombo";
fetch(URL1)
  .then((res) => {
    return res?.json();
  })
  .then((data) => {
    console.log(data, "data");
  });

let promise = new Promise((resolve, reject) => {
  reject("Hello JavaScript Failed!");
});

promise.then((result) => console.log(result)).catch(
  res=>{
    console.log(res)
  }
);

const URL2 = "https://api.github.com/users/mojombo";
const user = fetch(URL2)
  .then((res) => {
    return res?.json();
  })
  .then((data) => {
    console.log(data, "data");
  });
```



## **Lecture-03   Laying the Foundation**