**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:**

****

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

const heading = React.createElement(

  "h1",

  { id: "heading", "data-testid": "heading", testid: "heading" },

  "Hello World from React"

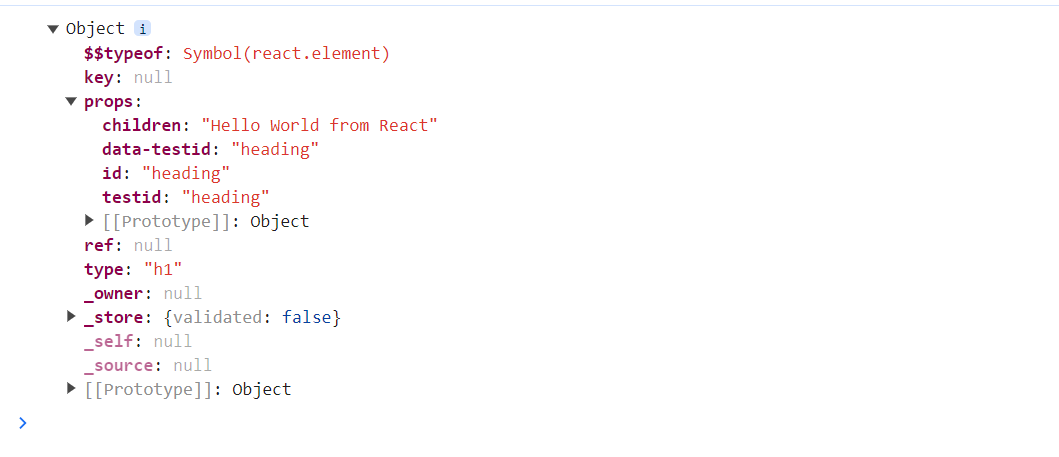
);

console.log(heading)

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

root.render(heading);

**Output:**

****

**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" },

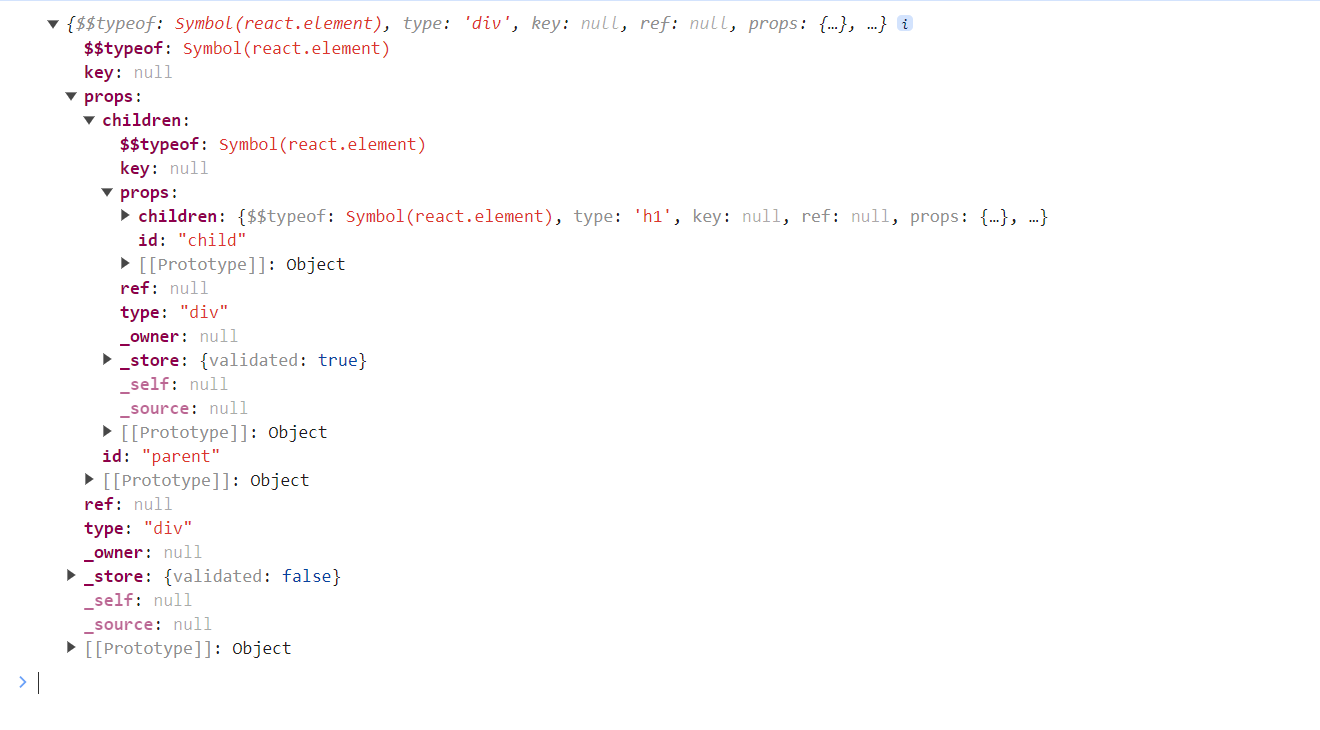
    React.createElement("h1", { id: "inner-child" }, "Hello World!")

  )

);

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

root.render(parent);

****

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")]

  )

);

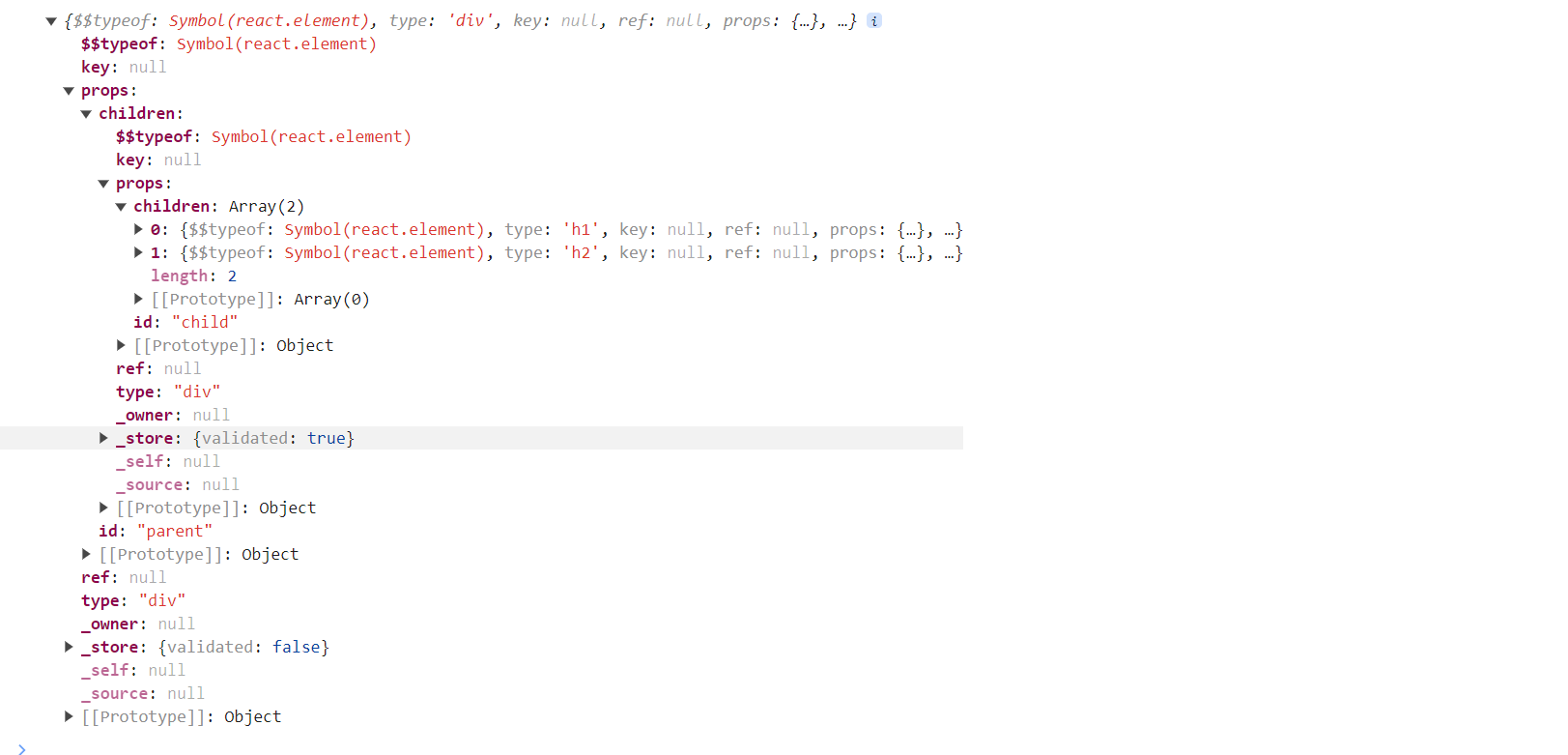
console.log(parent);

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

root.render(parent);

**Siblings are passed inside an Array**

****

****

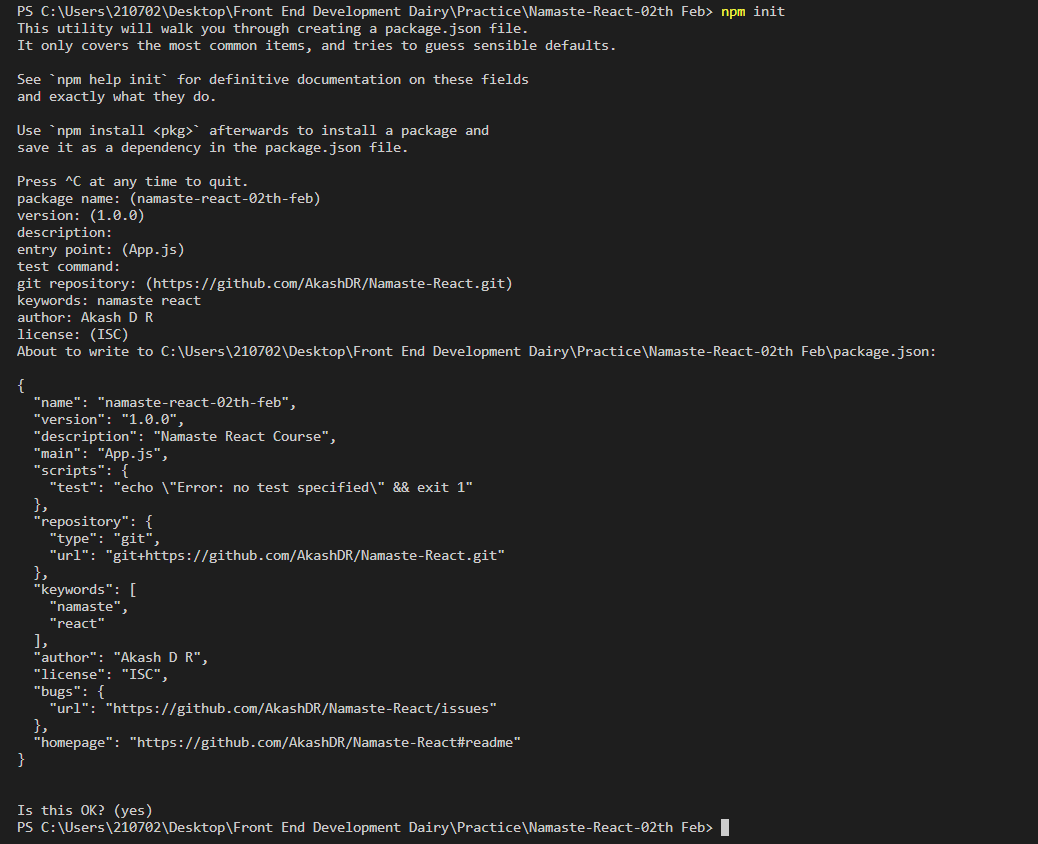
**It becomes extremely complex to write React Code like this. So, there came the need for JSX (HTML Like syntax inside Javascript)**

**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)**





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

1. **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**](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

****

**Difference between Caret and Tilde**

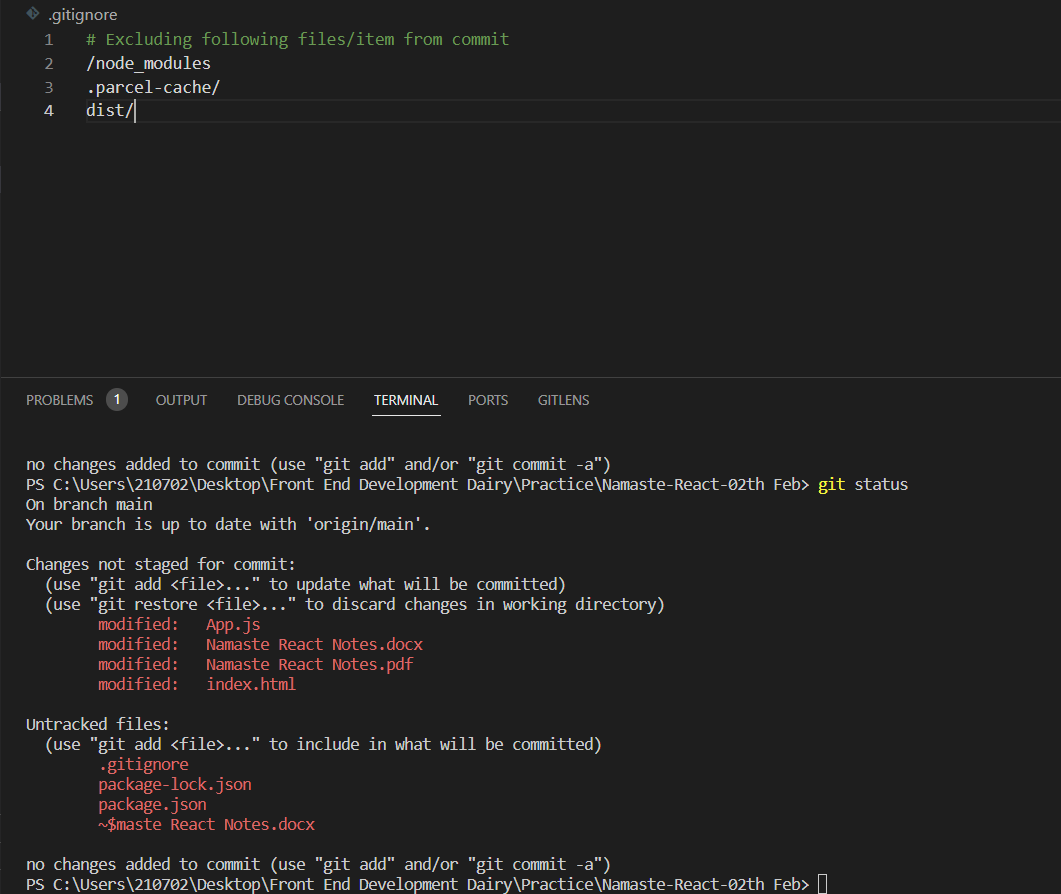
Caret(^) consider only patch and minor version update automatically. Caret(^) is less safer than Tilde(~) for production app. because here minor feature will also update automatically .

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

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

1. **cat > .gitignore**

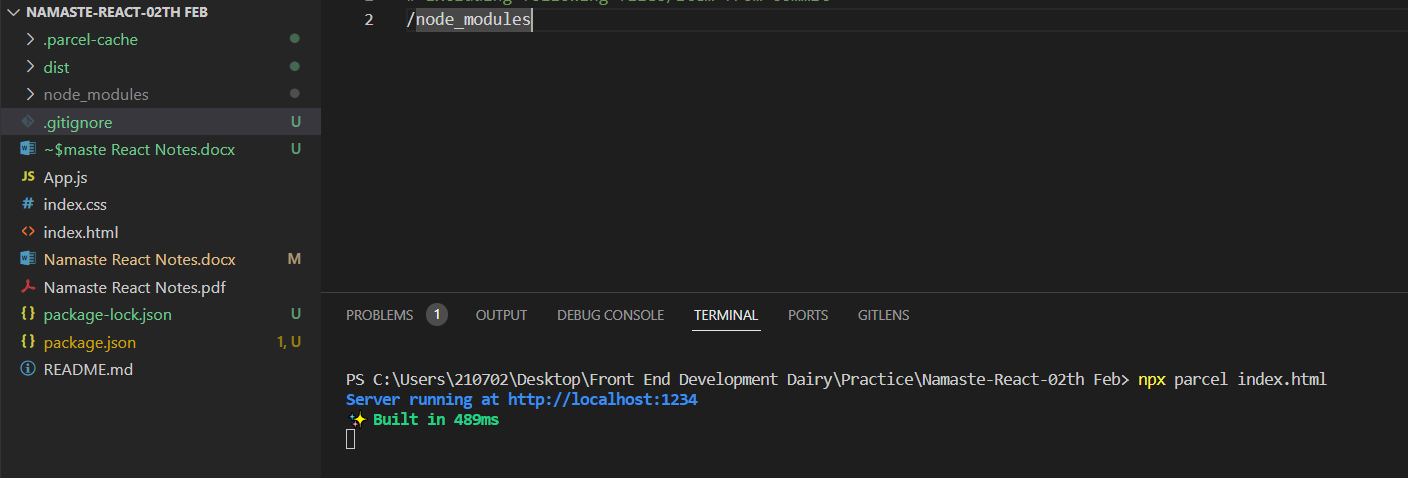
Create .gitignore file by using above command and add all the files which should not be committed



**How to create .gitignore file**

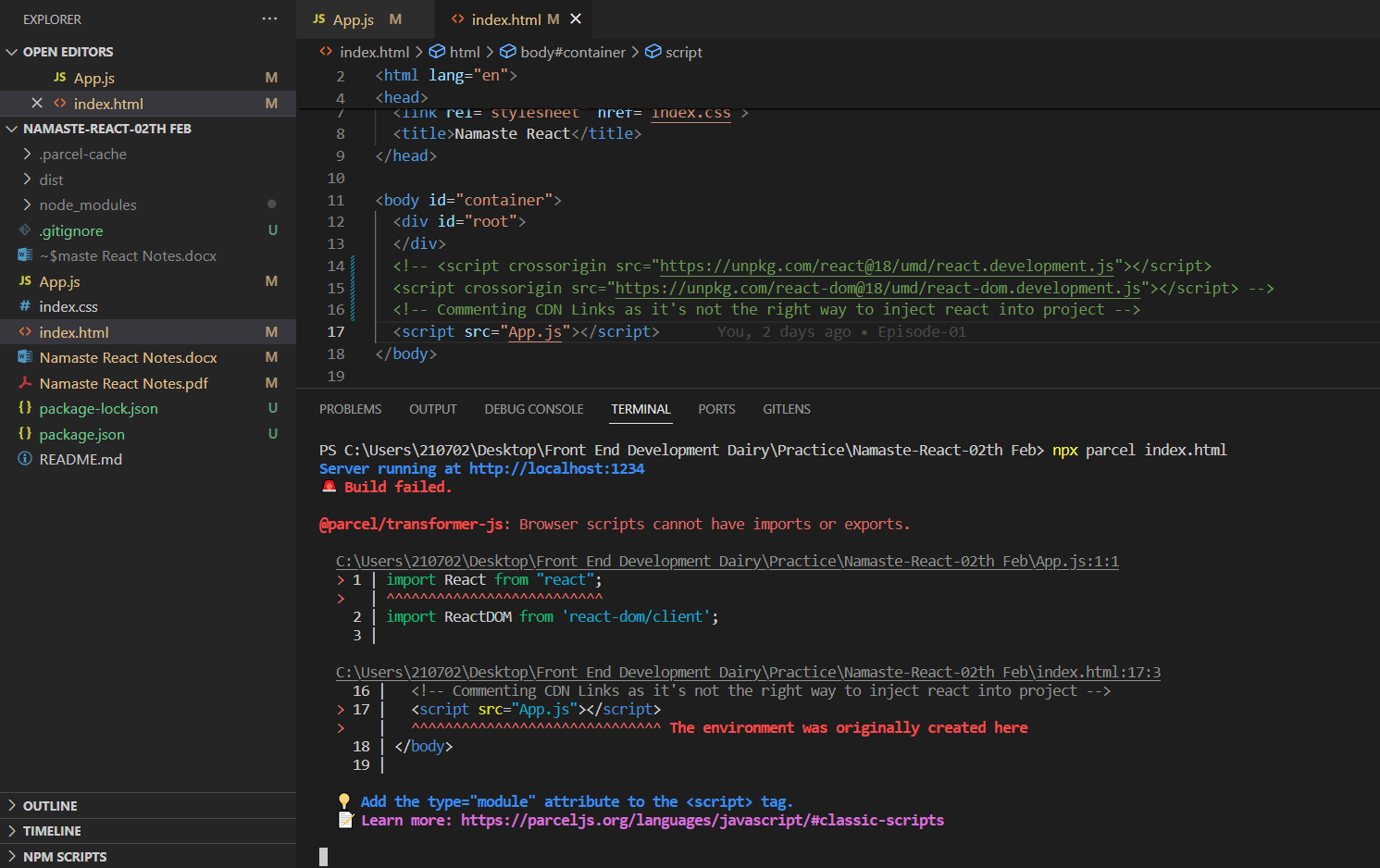
<https://www.youtube.com/watch?v=ErJyWO8TGoM&ab_channel=codebasics>

1. **npx create index.html**

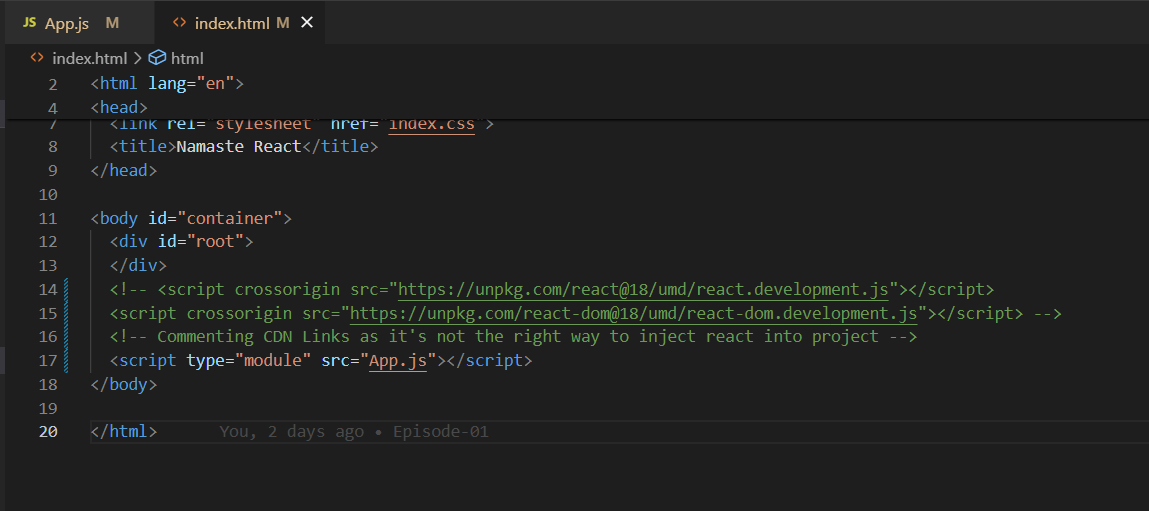
****

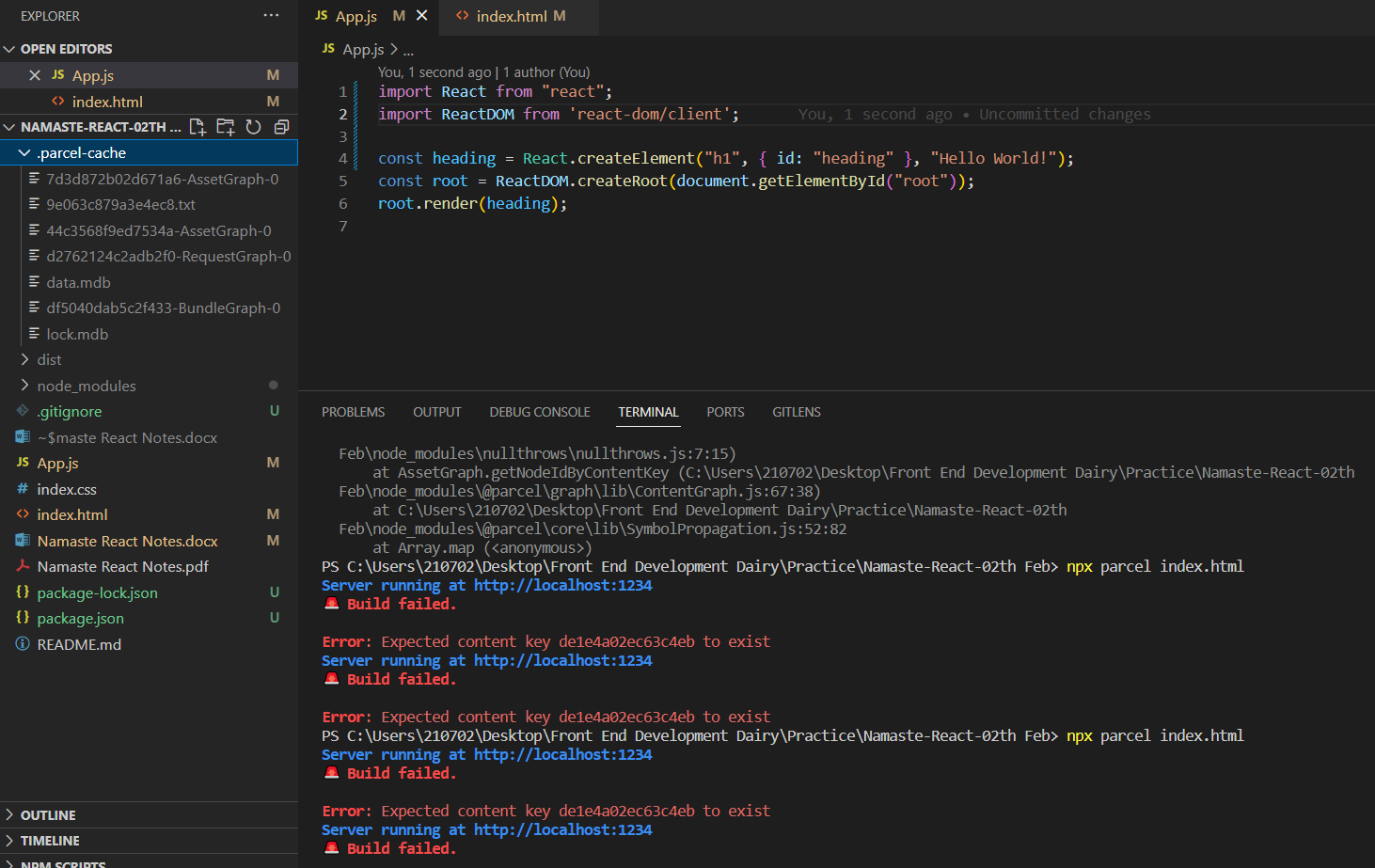
Start/Index the application using the above command. Notice that it has created .parcel-cache and dist folder inside the project. Project starts on port 1234

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

****

**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.**

****

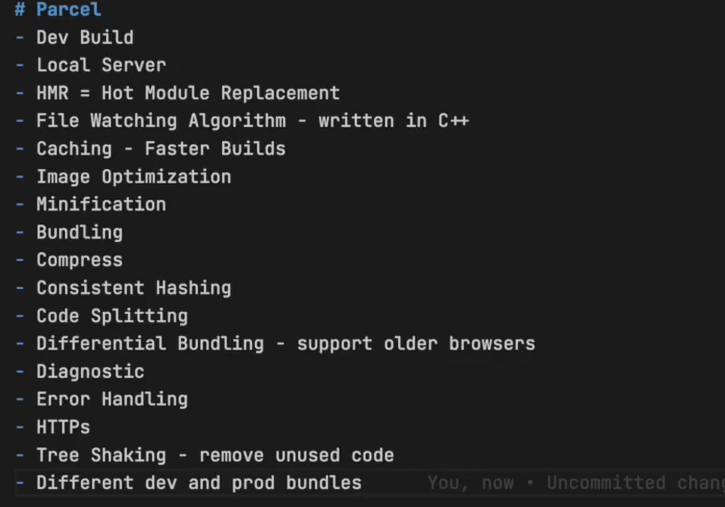
**Error: Expected content key de1e4a02ec63c4eb to exist**

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 (1st page itself).**

<https://parceljs.org/>

****

**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

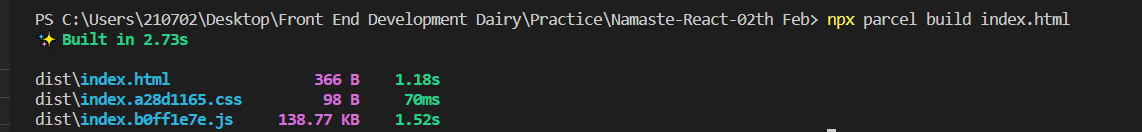
**How to create dev build?**

**Code: npx parcel index.html** (“Notice the keyword build missing”)

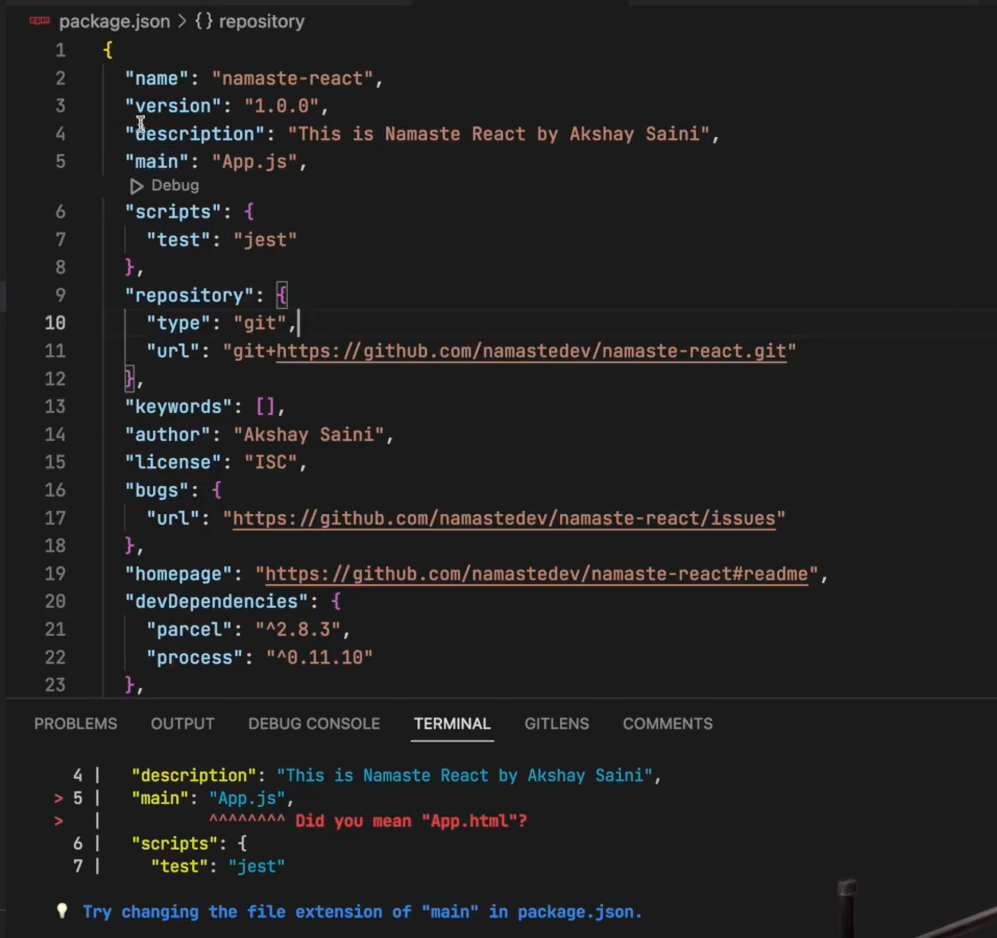
**How to create production ready build?**

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

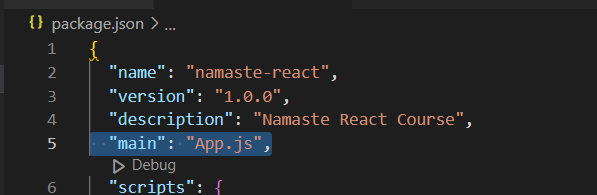
When you run this, the production build gets created in dist folder after all the optimization (Done by parcel).



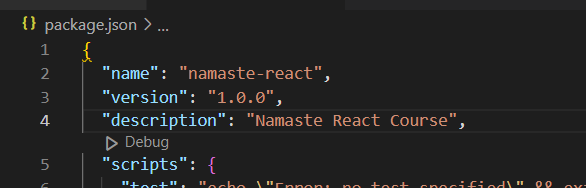
**Error: @parcel/namer-default: Target "main" declares an output file path of "App.js" which does not match the compiled bundle type "html".**

****

**To solve this error remove “main” in package.json**

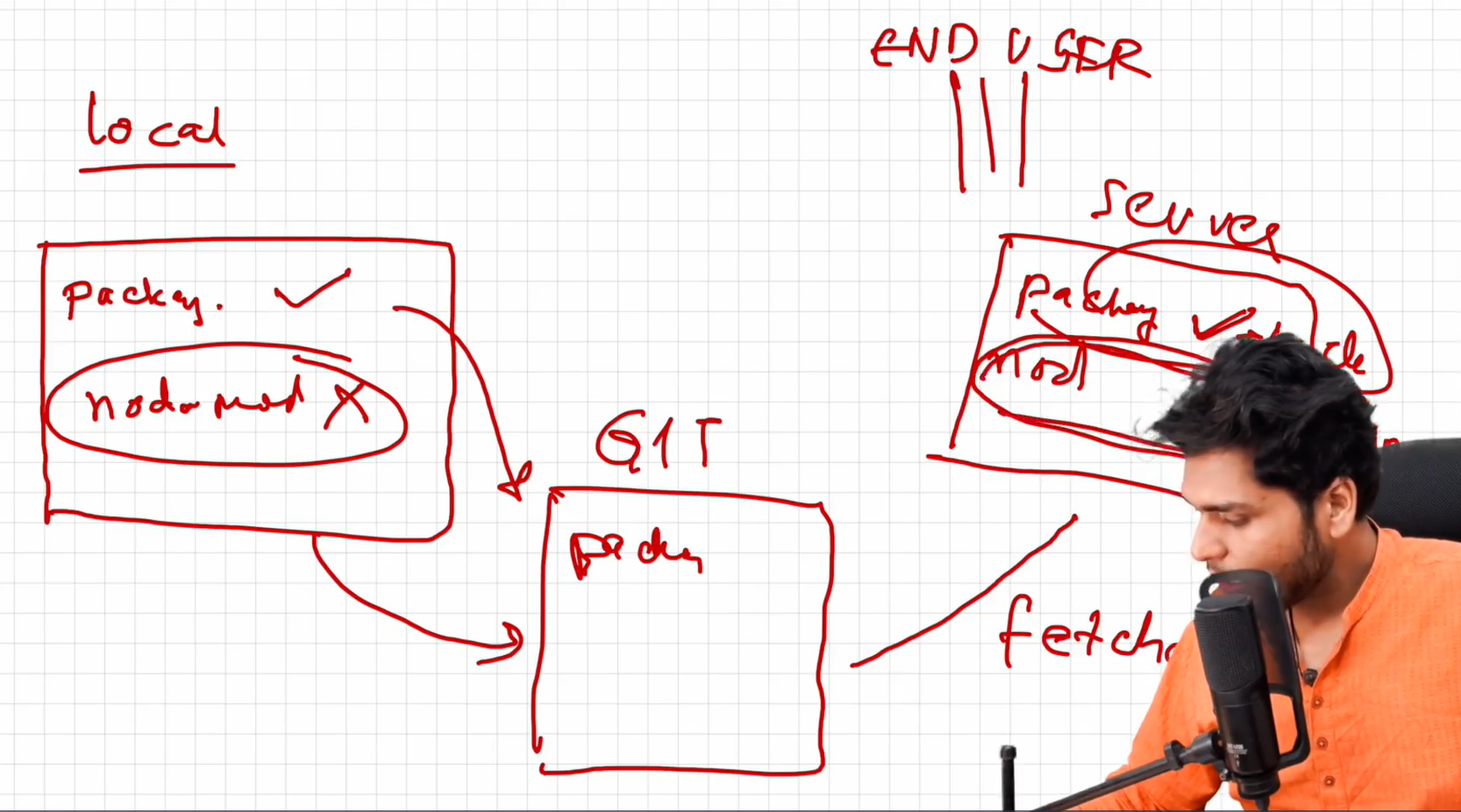
****

**Like this**

****

**Flow of Application:**

Server fetches package.json & packge-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=bGFzdCAyIHZlcnNpb25z>

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

Configuration is as follows:



**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**