<body>

    <div *id*="root"></div>

    <script>

*// Creating HTML tag using Javascript*

*const* heading = document.createElement("h1");

      heading.innerHTML = "Hello World From Javascript!";

*const* root = document.getElementById("root");

      root.appendChild(heading);

    </script>

  </body>

Q:-How does browser understand what is document,createElement,getElementById all these things how browser can Understand??

* Browser has Javascript Engine that exectute this JavaScript
* But Browser Don’t Understand React so first we need to get react into our project

# There is 2 way of adding react to our project

* Via CDN links
* npm install react / npm i react

CDN Links

<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>

1 and 2 links are different for a reason check BookPage-11

Q:- What is CORS (Cross-Origin Resource Sharing)

* <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

# Creating HTML Tag Element in React

<body>

// Before root and after root all element work fine only affected those inside the root by react (but what is rendering matter, I’m just giving example of root)

    <div *id*="root">

*//Whatever inside the id root will be replaced after root.render*

</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>

*// Its the Job of 1st CDN Link react*

*const* heading = React.createElement("h1", {}, "Hello World From React!");

*// Its the Job of 2nd CDN Link react-dom*

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

      root.render(heading);

    </script>

</body>

* *const* heading = React.createElement("h1", {}, "Hello World From React!");
* this heading Element at the end of the is Javascript Object
* root.render(heading);
* this render method is basically responsible to take this “heading” object and put it up, Convert it into the “HTML heading tag” and put it up(root i.e mention in code) on the DOM

# NPM is not Node package Manager, in npm’s official website there is no place where it has written that npm is Node Package Manager.

NPM Does not have Full Form 😅, so basically npm manages packages but it does not stand for Node Package Manager..

# **Initializing a New Project**

To create a new project, navigate to your desired project folder and run the following command:

$ npm init

This command will prompt you to enter some basic information about your project, such as the name, version, description, and entry point. Once you've provided the required information, ***npm will generate a package.json file***, which will contain all your project's metadata and dependencies.

# **Understanding package.json**

The package.json file is the heart of your project, as it stores all the necessary information about your project, such as its name, version, description, dependencies, and more. Here's a simple example of a package.json file:

{

"name": "my-project",

"version": "1.0.0",

"description": "A simple example project",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"dependencies": {

"express": "^4.17.1"

}

}

In this example, the project is named "my-project" and has a single dependency: the Express.js framework. The dependencies object lists all the packages required for your project to run correctly.

# **what-is-npm-and-how-does-it-work**

( <https://reintech.io/blog/what-is-npm-and-how-does-it-work> )

# **Package.json** is Configuration for npm

(<https://heynode.com/tutorial/what-packagejson/#:~:text=Your%20package.,dependencies%20required%20by%20the%20application>)

* **Sometimes Packages also known as Dependencies**

# **React Bundler**

(<https://www.dhiwise.com/post/embark-an-enlightening-journey-with-react-bundler> )

* When we have HTML CSS and JS file our whole code needs to be bundled together, our whole code needs to be minified, whole code needs to be compressed and needs to be cleaned before it can be sent to production so Bundler helps you to do all that things(webpack, parcel, vite)
* These bundlers are the job of to basically bundles our app it packages our app properly so that it can be shipped to production
* That “create-react-app” behind the scene uses “webpack” bundler
* But in our project we are using “parcel” bundler

# **parcel Bundler**

* Parcel Bundler Ignite our App
* npm install -D parcel
* here -D is devDependency(search on net for more info)
* These are your development dependencies. Dependencies that you need at some point in the development workflow but not while running your code (e.g. Babel or Flow).
* ( <https://github.com/parcel-bundler/parcel> )

# **Dependencies vs devDependency**

* (<https://medium.com/@reemshakes/devdependencies-vs-dependencies-in-reactjs-db7261e13012#:~:text=When%20your%20project%20needs%20code,list%20of%20your%20project's%20dependencies> )

# **whats the difference between tilde(~) and caret(^) in package.json**

* ~version **“Approximately equivalent to version”**, will update you to all future patch versions, without incrementing the minor version. ~1.2.3 will use releases from 1.2.3 to <1.3.0.
* ^version **“Compatible with version”**, will update you to all future minor/patch versions, without incrementing the major version. ^1.2.3 will use releases from 1.2.3 to <2.0.0.
* (<https://stackoverflow.com/questions/22343224/whats-the-difference-between-tilde-and-caret-in-package-json> )

# Package.json vs Package-lock.json

* What is the purpose of package.json?
* Tilde (~) and carat (^), and their difference
* What is package-lock.json?
* What is the purpose of package-lock.json?
* Comparing package.json and package-lock.json
* What is the role of npm-shrinkwrap.json in versioning?
* (<https://www.atatus.com/blog/package-json-vs-package-lock-json/> )
* "node\_modules/@babel/code-frame": {
* "version": "7.23.5",
* "resolved": "https://registry.npmjs.org/@babel/code-frame/-/code-frame-7.23.5.tgz",
* "integrity": "sha512-CgH3s1a96LipHCmSUmYFPwY7MNx8C3avkq7i4Wl3cfa662ldtUe4VM1TPXX70pfmrlWTb6jLqTYrZyT2ZTJBgA==",
* "dev": true,
* "dependencies": {
* "@babel/highlight": "^7.23.4",
* "chalk": "^2.4.2"
* },
* "engines": {
* "node": ">=6.9.0"
* }
* },
* Above data is from package-lock.json, **here the integrity meaning**
* This is Hash, have you heard of that thing it is working on my Local, I Don’t know how it break / it’s not working on Production
* So basically to avoid that package-lock.json keeps hash to verify that whatever is there in my machine is the same version which is being deployed onto the production.
* It’s very important file package-lock.json it keeps a track of all the exact versions of all dependencies
* When we install parcel there is one more thing that was created i.e node\_modules

Q:- What is node\_modules folder in react (BookPage-15)

* When creating a new React app your project will be populated with a bunch of new folders and files, and you may have noticed a **node\_modules** folder that contains an insane amount of folders.
* Because our needs parcel / project has dependency parcel. Now parcel as a project has it’s own dependencies and those dependencies can have it’s own dependencies, those dependencies can gave their own dependencies, this is known as “**Transitive Dependencies**”, Now this parcel can itself be dependent on lot of things. Parcel cannot do all these things on its own, parcel needs help of a lot of other packages, example parcel needs help of Babel also.(*Akshay Saini*)
* node\_modules are one of the most important directories in your React project as React requires node\_modules to run. The node\_modules directory is where all the dependancies packages are stored that are used to build and run your react project.
* So this is you'll find packages like React and React-DOM, your build packages like Vite, Babal or Webpack, and linters like ESLint or Prettier to name just a few. This directory can contain hundreds of dependancies!
* Q:- **Why aren't they included in version control?**
* => The main reason is the sheer size of this directory. Rather than including hundreds of package dependancies in version control, we can instead track a file called **package.json** which contains information about the project, and a list of dependencies required by the app. Other developers can use the package.json file and **npm install** to regenerate the node\_modules.

# **😱 when accidentally tracked node\_modules**

* Best practice would be to include the node\_modules in the **.gitignore** file in your project before pushing any code,
* which will prevent version control from tracking this folder - but if you've accidentally tracked and pushed the node\_modules to GitHub, like I did in my first project 🤦🏻‍♀️, simply follow the steps below.
* **Remove node\_modules from version control**
* Create a .gitignore file in your project and add node\_modules
* **Remove the node\_modules:**
* ***git rm -r --cached node\_modules***
* Commit and push without the node\_modules. The node\_modules should now be deleted from your repository.

**# Igniting Our App / Running Our Application On a Development Server**

* npx parcel src/index.html
* Output
* ❯ npx parcel src/index.html
* Server running at http://localhost:1234
* ✨ Built in 5ms
* Parcel’s built-in development server is now running. The npx parcel command takes your entry point of src/index.html and builds your application with the necessary assets. The output also indicates that the application is running on <http://localhost:1234>.
* ***# How to bundle a web app with parcel***
* <https://www.digitalocean.com/community/tutorials/how-to-bundle-a-web-app-with-parcel-js>

1. **Babel in React**

( <https://www.scaler.com/topics/react/what-is-babel-in-react/> )

1. **Difference Between HTML and JSX**

( <https://www.freecodecamp.org/news/html-vs-jsx-whats-the-difference/> ) ( <https://codersera.com/blog/react-functional-components/> )

1. **Functional Componet**

( <https://codersera.com/blog/react-functional-components/> )

**Config-driven-UI**

[**https://www.freecodecamp.org/news/javascript-optional-chaining/**](https://www.freecodecamp.org/news/javascript-optional-chaining/)

**map array**