Namaste React Notes

Lecture 1- Inception

Hello World Program by using HTML

```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

```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">
  </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)

```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">
  </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

```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">
  </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

```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

```javascript
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

```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",
     ▷ Debug
  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>
```
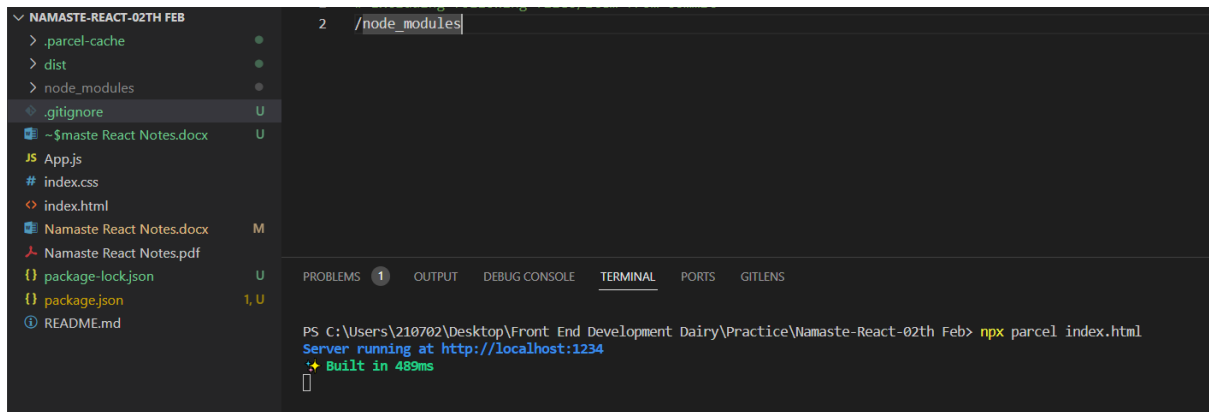
How to create .gitignore file
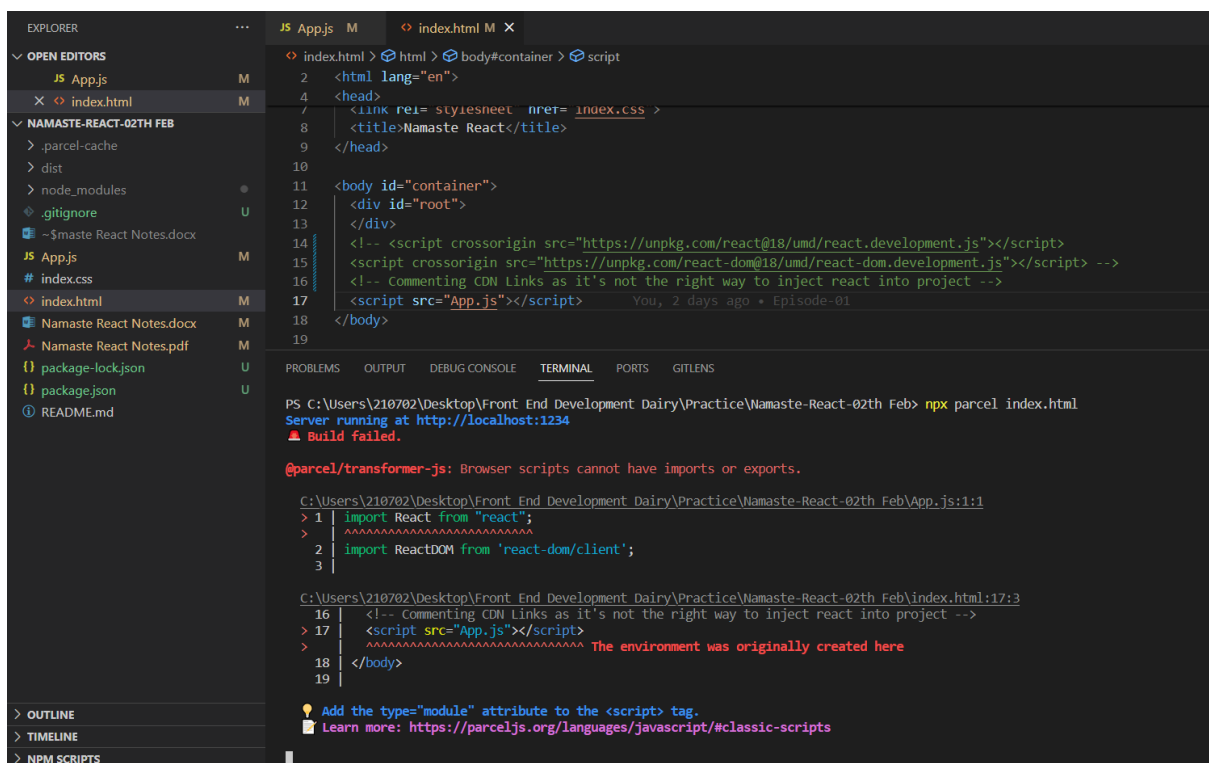
4. npx create index.html

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.

```
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

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/

```
# 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",
        ▷ Debug
  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     },
```

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"

  💡 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",
        ▷ Debug
  6     "scripts": {
```

Like this

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

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:
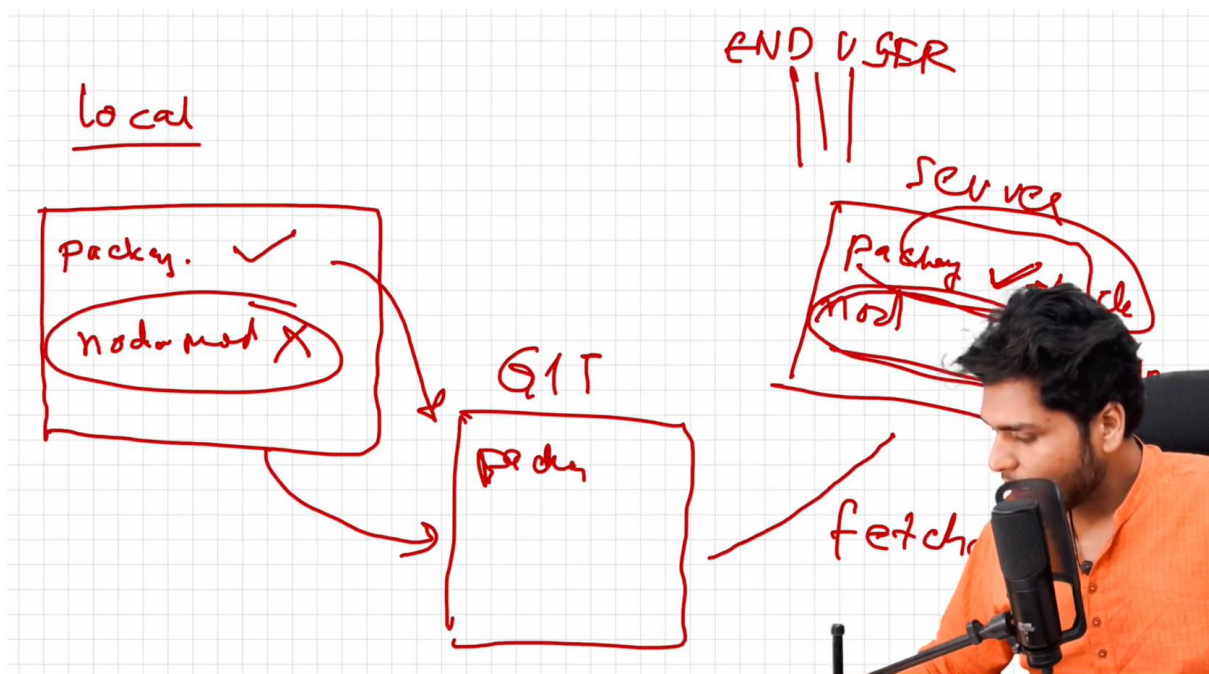
```json
{
  "name": "namaste-react",
  "version": "1.0.0",
  "description": "Namaste React Course",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/AkashDR/Namaste-React.git"
  },
  "keywords": [],
  "author": "Akash D R",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/AkashDR/Namaste-React/issues"
  },
  "homepage": "https://github.com/AkashDR/Namaste-React#readme",
  "devDependencies": {
    "parcel": "^2.11.0",
    "process": "^0.11.10"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "browserlist":[
    "last 10 Chrome version",
    "last 10 Firefox version",
    "last 10 versions"
  ]
}
```

Promises: How to extract data from Promises?

```javascript
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");
  });
```

Check the state of Promise

```
function createOrder(cart) {
  const promise = new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("12345");
      console.log(promise, "promise2");
    }, 5000);
  });
  console.log(promise, "promise1");
  return promise;
}

createOrder(cart)
  .then((res) => {
    console.log(res, "res");
  })
  .catch((err) => {
    console.log(err, "err");
  });
```

```
▼ Promise {<pending>} ⓘ                'promise1'
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "12345"

▼ Promise {<fulfilled>: '12345'} ⓘ   'promise2'
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "12345"

12345 res

>
```

Usage of finally in Promises.

```
function createOrder(cart) {
  const promise = new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("12345");
      console.log(promise, "promise2");
    }, 5000);
```

```
  });
  console.log(promise, "promise1");
  return promise;
}

createOrder(cart)
  .then((res) => {
    console.log(res, "res");
  })
  .catch((err) => {
    console.log(err, "err");
  }).finally(err=>{
    console.log("Just Print - No Matter What")
  });
```

```
▶ Promise {<pending>} 'promise1'

▶ Promise {<fulfilled>: '12345'} 'promise2'

12345 res

Just Print - No Matter What

> |
```

Promise Chaining:

Notice the word return inside then block. You should always return if you want to create Promise chain. Else it would lead to Pyramid kind of structure similar to Callback hell.

```
const cart = ["shoes", "pants", "kurtas"];

function createOrder(cart) {
  return new Promise((resolve, reject) => {
    resolve("Cart Creation Successful");
  });
}

function proceedToPayment(orderId) {
  return new Promise((resolve, reject) => {
    resolve("Payment Successful");
  });
}

createOrder(cart)
```

```
    .then((res) => {
      console.log(res);
      return res;
    })
    .then((res) => {
      console.log(res);
      return proceedToPayment(res);
    })
    .then((res) => {
      console.log(res);
      return proceedToPayment(res);
    })
    .then((res) => {
      console.log(res);
      return proceedToPayment(res);
    })
    .then((res) => {
      console.log(res);
    })
    .catch((err) => {
      console.log(err, "err");
    });
```

```
Cart Creation Successful

Cart Creation Successful

Payment Successful

Payment Successful

Payment Successful
```

>

Catch checks only the errors that come above it. It won't check below it. If we cart creation is failed, we can still move to payment. See Details below.

```
const cart = ["shoes", "pants", "kurtas"];

function createOrder(cart) {
  return new Promise((resolve, reject) => {
    reject("Cart Creation Failed");
  });
}

function proceedToPayment(orderId) {
  return new Promise((resolve, reject) => {
    resolve("Payment Successful");
  });
}
```

```
createOrder(cart)
  .then((res) => {
    console.log(res);
    return res;
  })
  .catch((err) => {
    console.log(err);
    return err
  })
  .then((res) => {
    console.log(res);
    return proceedToPayment(res);
  })
  .then((res) => {
    console.log(res);
    return proceedToPayment(res);
  })
  .catch((err) => {
    console.log(err, "err");
  });
```

Notice first Catch Block above. It catches only failures in Create Cart Function

Cart Creation Failed	index.js:67
Cart Creation Failed	index.js:71
Payment Successful	index.js:75

>

```
47    const cart = ["shoes", "pants", "kurtas"];
48
49    function createOrder(cart) {
50      return new Promise((resolve, reject) => {
51        reject("Cart Creation Failed");
52      });
53    }
54
55    function proceedToPayment(orderId) {
56      return new Promise((resolve, reject) => {
57        reject("Payment Failure");
58      });
59    }
60
61    createOrder(cart)
62      .then((res) => {
63        console.log(res);
64        return res;
65      })
66      .catch((err) => {
67        console.log(err);
68        return err
69      })
70      .then((res) => {
71        console.log(res);
72        return proceedToPayment(res);
73      })
74      .then((res) => {
75        console.log(res);
76        return proceedToPayment(res);
77      })
78      .catch((err) => {
79        console.log(err, "err");
80      });
81
```

Notice now, the Promise failed at both places, but first catch block catched only Cart Failure error.

```
Cart Creation Failed                                                          index.js:67
Cart Creation Failed                                                          index.js:71
Payment Failure err                                                           index.js:79
> |
```

Lecture-03   Laying the Foundation