# Episode-10 | Jo Dikhta he Vo Bikta he

Please make sure to follow along with the whole ”**Namaste React**" series, starting from Episode-1 and continuing through each subsequent episode. The notes are designed to provide detailed explanations of each concept along with examples to ensure thorough understanding. Each episode builds upon the knowledge gained from the previous ones, so starting from the beginning will give you a comprehensive understanding of React  dvelopment.

I've got a quick tip for you. To get the most out of these notes, it's a good idea to watch **Episode-10** first. Understanding what ”**Akshay**” shares in the video will make these notes way easier to understand.

## Q ) Explore all the ways of writing CSS.

> **Using CSS  CSS can be added to HTML documents in `3 ways` :**

```
1. Inline
2. Internal
3. External
```

1 `Inline` - by using the `style attribute` inside HTML elements.

```
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
```

2 `Internal` - by using a `<style>` element in the section.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {background-color: powderblue;}
      h1   {color: blue;}
      p    {color: red;}
    </style>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

3 `External` - by using a `<link>` element to link to an external CSS file.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

## Q: How do we configure `tailwindcss` ?

> Configuring Tailwind CSS involves a few simple steps. Tailwind CSS is often configured using a configuration file where you can customize various settings, such as colors, fonts, breakpoints, and more. Here's a step-by-step guide:

**Step 1️⃣ Create a new project (if not already done)**

Ensure you have a new or existing project where you want to use Tailwind CSS.

**Step 2️⃣ Install Tailwind CSS**

You can install Tailwind CSS using npm or yarn. Open your terminal or command prompt and navigate to your project's root directory. Run one of the following commands: **Using npm:**

```
npm install tailwindcss
```

###  Step 3 Create a Configuration File

Create a configuration file for Tailwind CSS. You can generate a basic configuration file using the following command:

```
npx tailwindcss init
```

This command creates a tailwind.config.js file in your project's root directory.

###  Step 4 Customize the Configuration Optional)

Open the generated tailwind.config.js file, and you can customize various aspects of Tailwind CSS according to your project's needs. This file includes options for colors, fonts, spacing, breakpoints, and more.

For example, you can customize the colors in the tailwind.config.js file like this:

```
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#3490dc',
```

```
        secondary: '#ffed4a',
        // ...add more custom colors as needed
      },
    },
  },
  // ...other configurations
};
```

### Step 5️⃣ Create CSS File

Create a CSS file where you will import Tailwind CSS and any additional styles. Typically, this file is named styles.css or similar. Import Tailwind CSS using the @import directive.

```
/* styles.css */
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

/* Add your custom styles here */
```

### Step 6️⃣ Build Your Styles

Include your CSS file in your HTML or import it in your JavaScript file if you are using a bundler like Webpack.

### Step 7️⃣ Use Tailwind CSS Classes in HTML

Now, you can start using Tailwind CSS classes in your HTML files to apply styles. For example:

```
<div class="bg-primary text-white p-4">
  This is a primary-colored box with white text and padding.
</div>
```

### Step 8️⃣ Build Your Project

Depending on your setup, you might need to build your project to apply the Tailwind CSS styles. If you're using a bundler like Webpack, make sure to run the appropriate build command.

For example, with npm:

```
npm run build or npm start
```

That's it! We've successfully configured and started using Tailwind CSS in your project. Remember to consult the official documentation for more detailed information and advanced configurations.

## Q ) In `tailwind.config.js` , what does all the keys mean (content, theme, extend, plugins)?

> In tailwind.config.js, the various keys serve different purposes and allow you to customize and configure different aspects of Tailwind CSS. Here's an overview of what each key typically represents:

### 1. content Key:

`Purpose` Specifies the files that Tailwind CSS should analyze to generate its utility classes. Usage:

```
module.exports = {
  content: [
    './src/**/*.html',
    './src/**/*.js',
    // Add more file paths as needed
  ],
  // ...other configurations
}
```

The content key helps Tailwind CSS identify which files to process and extract utility classes from. It is particularly useful when working with frameworks like React or Vue.

### 2. theme Key:

Purpose: Defines the default styles and configurations for various aspects of Tailwind CSS, such as colors, spacing, fonts, and more. Usage:

```
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#3490dc',
        secondary: '#ffed4a',
        // ...add more custom colors
      },
    },
    // ...other theme configurations
  },
  // ...other configurations
};
```

The theme key allows you to customize default styles and extend or override the default configuration provided by Tailwind CSS. It is where you can define your

project-specific design system.

### 3. extend Key

Purpose: Extends or overrides the default configuration provided by Tailwind CSS. Usage:

```
module.exports = {
  extend: {
    colors: {
      primary: '#3490dc',
      secondary: '#ffed4a',
      // ...add more custom colors
    },
    // ...other extensions
  },
  // ...other configurations
};
```

The extend key is often used to add new styles or extend existing ones. It is especially useful for adding project-specific utility classes or modifying existing ones.

### 4. plugins Key

Purpose: Allows you to use or define custom plugins to extend or modify Tailwind CSS functionality. Usage:

```
module.exports = {
  plugins: [
    require('@tailwindcss/forms'), // Example plugin
    // ...add more plugins as needed
  ],
```

```
  // ...other configurations
};
```

The plugins key lets you incorporate third-party plugins or create your own custom plugins. Plugins can add new features, styles, or utilities to Tailwind CSS.

These keys provide a flexible and powerful way to configure Tailwind CSS based on your project's requirements. They allow you to control the content, define styles, extend default configurations, and enhance functionality through plugins. Remember to consult the official Tailwind CSS documentation for detailed information on each configuration option and best practices.

## Q ) Why do we have `.postcssrc` file?

> The `.postcssrc` file, often named postcss.config.js, is a configuration file for PostCSS. PostCSS is a tool for transforming styles with JavaScript plugins, and it is commonly used in conjunction with build tools like webpack or parcel for processing and optimizing CSS.

Here are the primary reasons why you might have a .postcssrc file:

- **Plugin Configuration:**

The main purpose of the .postcssrc file is to configure the plugins that PostCSS should use during the CSS transformation process. These plugins can handle tasks such as autoprefixing, minification, and syntax enhancements.

- **Custom Configuration:**

You may need a .postcssrc file if you want to customize the behavior of PostCSS beyond the default settings provided by the build tool (e.g., webpack). This allows you to have fine-grained control over the PostCSS transformations.

- **Presets and Options:**

PostCSS plugins often come with various options and presets that you can configure based on your project's needs. The .postcssrc file is a convenient place to define these options and presets.

- **Maintainability:**

Separating the PostCSS configuration into its own file makes the build configuration more maintainable and organized. It allows you to centralize PostCSS-related settings and keep them distinct from other build tool configurations.

- **Sharing Configurations:**

Having a dedicated configuration file makes it easier to share and reuse PostCSS configurations across different projects. It can be particularly useful in larger development ecosystems where consistent styles and build processes are desired.

Example .postcssrc file:

```
// postcss.config.js

module.exports = {
  plugins: {
    // Example plugins with options
    'autoprefixer': {},
    'postcss-preset-env': {
      stage: 3,
      features: {
        'nesting-rules': true,
      },
    },
    'cssnano': {
      preset: 'default',
    },
  },
};
```

In this example, the .postcssrc file configures three PostCSS plugins: autoprefixer for adding vendor prefixes, postcss-preset-env for enabling future CSS features, and cssnano for minification. The options provided for each plugin customize their behavior.

Remember that the specific configuration options and plugins you include in your .postcssrc file will depend on your project's requirements and the PostCSS features we want to leverage.

# Self- Notes-

There is multiple way to use CSS-

- Normal,
- styled (https://styled-components.com/),
- sass
- material-ui (https://mui.com/material-ui/getting-started/) – already prebuild component you just import on your app and build the component.
- Bootstrap
- Chakra UI
- Ant Design
- Tailwind (https://tailwindcss.com/)

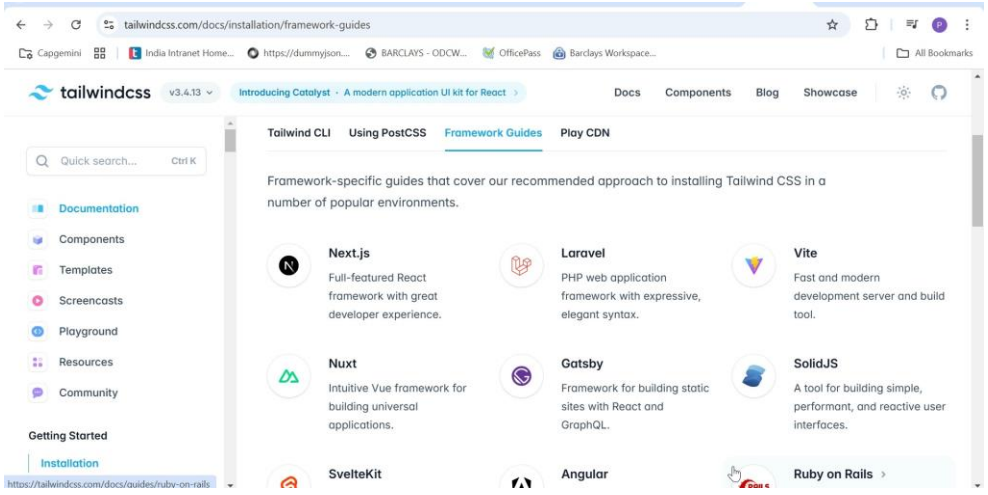
## Tailwind CSS-

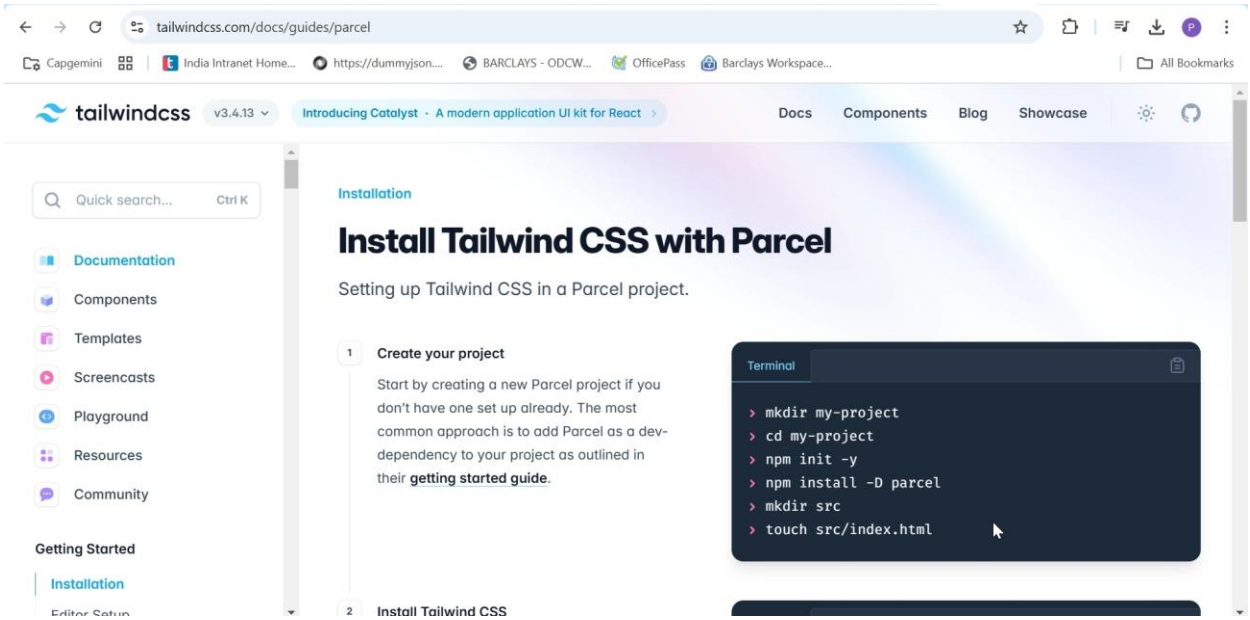
Rapidly build modern websites without ever leaving your HTML.

Means- Tailwind css says you don't have to leave your HTML file or you don't have to leave your JS file you can still style your components without moving and switching between the files.

- Tailwind CSS can use with other frameworks like Angular, JS etc. It's the generic css framework.

Will use Tailwind with parcel because our project use parcel






https://tailwindcss.com/docs/guides/parcel

**Configuration the Tailwind CSS-**

```
npm install -D tailwindcss postcss
npx tailwindcss init
```

It will install tailwindcss and postcss

**PostCSS** - a tool for transforming CSS with JavaScript
basically, if you transfer your css inside JS you have to use postcss
Tailwind css behind the scene use postcss.

Npm tailwind init means you have to invoking, we are executing the tailwind css.
We are just initiating the tailwind css into the repository.
And it will create the tailwind.config.js folder.

Now Configure PostCSS
Create a .postcssrc file in your project root, and enable the tailwindcss plugin.

**postcssrc -**

```
{
  "plugins": {
    "tailwindcss": {}
  }
}
```

You have to tell the postcssrc you are using tailwindcss over here.
So Postcssrc to read tailwind, because you are using parcel, so parcel your bundler use postcssrc is used to understand tailwind. So postcssrc is way to understand what is written inside tailwind.

Do the modification in tailwind.config.js-

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{html,js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

This configuration means that this content attribute takes a array, takes a list of all the files where I can use my tailwind css like it can be use in html file , it can be use in js file , can be use in ts files, use in .jsx file and can be use in .tss file . This is a way we are telling are project that we can use tailwind in any of these files which using these extensions.
We can modify the file as well-
"./src/**/*.{html,js}",

Now we are importing tailwind css into css file , import in index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Configuration is done, Now we can use tailwind in to the project

Tailwind css gives you className automatically for every CSS that want to you write in your app.

Install tailwind vs code extension -> **Tailwind CSS IntelliSense**

Basically it will give you the suggestions.

Ctrl+spacebar – if you are not able to see the suggestions.

Disadvantage – code become ugly.
Advantage- Tailwind CSS is very light weight , suppose in our code I am using just 5 6 class so when parcel will make the bundle of css it will only include the CSS that is required on the webpage.