# Decoupling Patterns, Services and creating an enterprise level editorial experience

## Who we are and Why we are here?

## Saurabh Chugh

- Started Drupal journey in 2010  with Drupal 6 , long journey with Drupal 7
- https://www.drupal.org/u/saurabh-chugh
- Travelling, Photography , Make Friends and Learn
- Interested areas: E-Commerce, LMS

ACQUIA
CERTIFIED
DEVELOPER
Drupal 7

ACQUIA
CERTIFIED
DEVELOPER
Drupal 8

ACQUIA
CERTIFIED
FRONT END SPECIALIST
Drupal 8

## Surendra Singh

- Working with Tata Consultancy Services
- Started Drupal journey in 2013  with Drupal 7 , long journey with Drupal 7
- Finally in :Love: of Drupal 8 https://www.drupal.org/u/surendrasingh1
- Love Drupal Camps and Cons, Travelling, Foodie
- Interested areas: LMS and Decouple

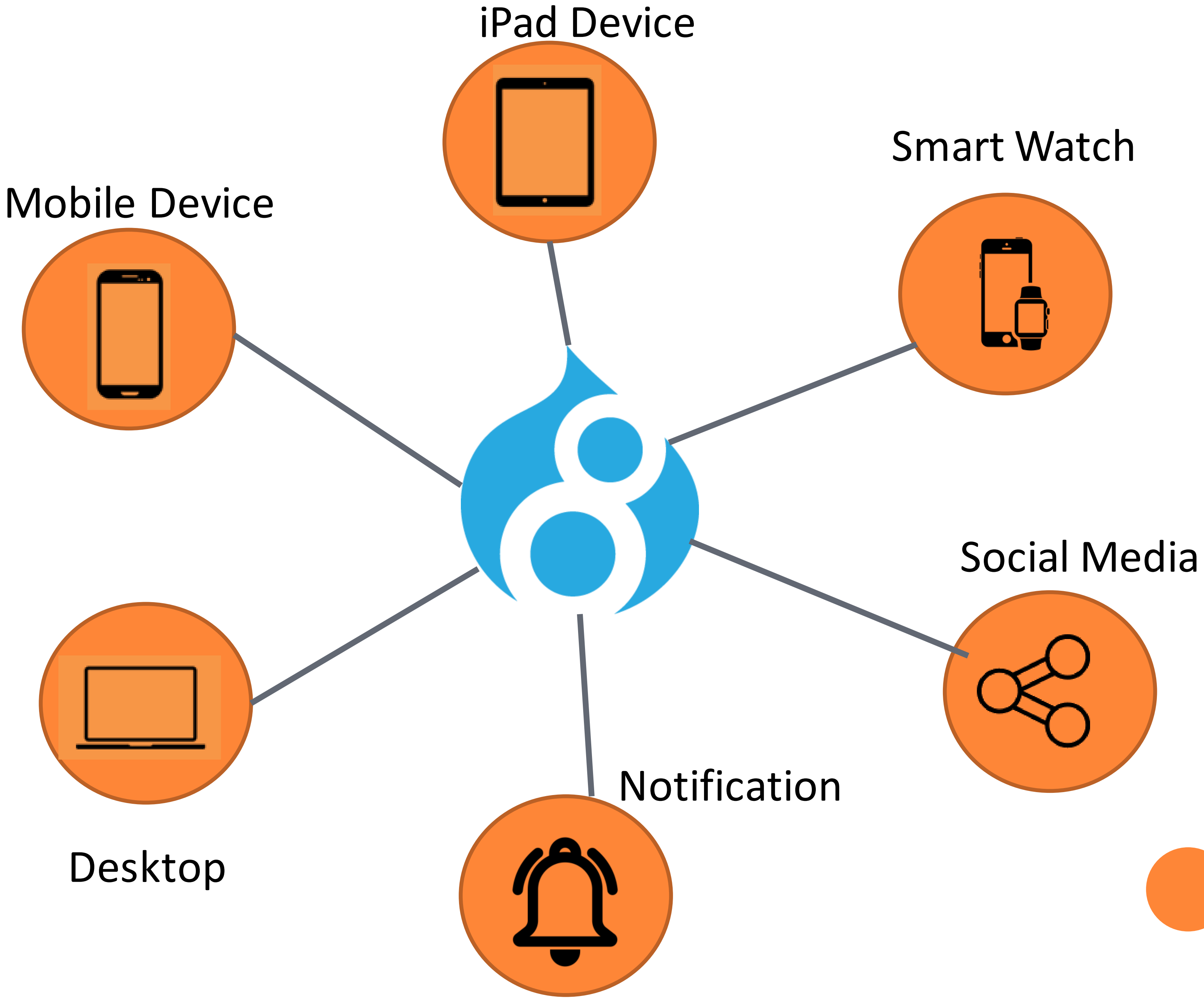
## AGENDA

- Decoupled
- Benefits of Decoupling
- Decoupled VS Progressive Decoupled
- Deciding Path for Decoupled Drupal 8
- JSON API
- React
- Editorial Pain Points
- Empowering the editors / Content Creators
- Editorial tools
- QNA

## Exposing Drupal

## DECOUPLED

- Decoupled is about Multichannel
- Create Once, Publish Everywhere
- CMS prepares content for presentation and pushes it into a delivery environment
- Separating applications backend layer from it’s presentation layer
- Decoupling is a process in which the front-end developer gaining the full control on logic of presenting the content without having the knowledge of how the content gets created and stored at Backend.


iPad Device
Smart Watch
Mobile Device
Social Media
Notification
Desktop

## BENEFITS OF DECOUPLING

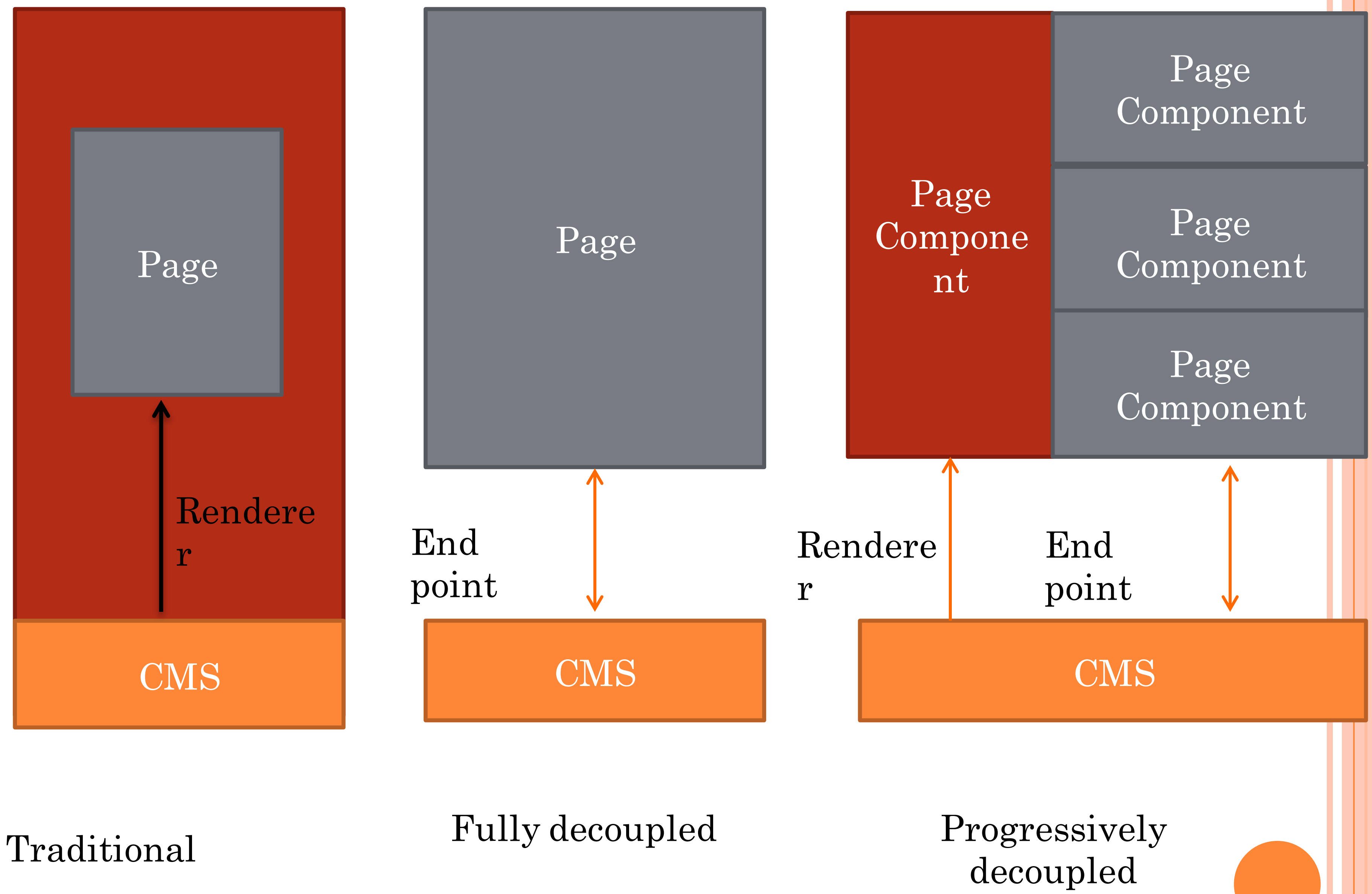
- Future-proofs your website implementation, letting you redesign the site without re-implementing the CMS itself
- Sets frontend developers free from the conventions and structures of the backend. Headless development not only eliminates “div-itis”, but it also gives frontend specialists full control over the user experience using their native tools
- Speeds up the site by shifting display logic to the client-side and streamlining the backend. An application focused on delivering content can be much more responsive than one that assembles completely formatted responses

## Decoupled vs Progressively Decoupled

## Decoupled vs Progressively Decoupled

- JS Framework (react/angular) and Drupal can be used together in two different ways: fully decoupled , also known as headless; or progressively decoupled.

- Find out the answer, which approach is used in your application
  - If the JavaScript that makes up your React application being added to an existing Drupal page? If answer is **YES** then we are already working in Progressive decupled architecture.
  - If the answer is **NO**, than you're using Drupal in a decoupled manner, wherein Drupal acts solely as the data store for your application and plays no part in determining what the user experience consists of JavaScript framework (React/Angular)

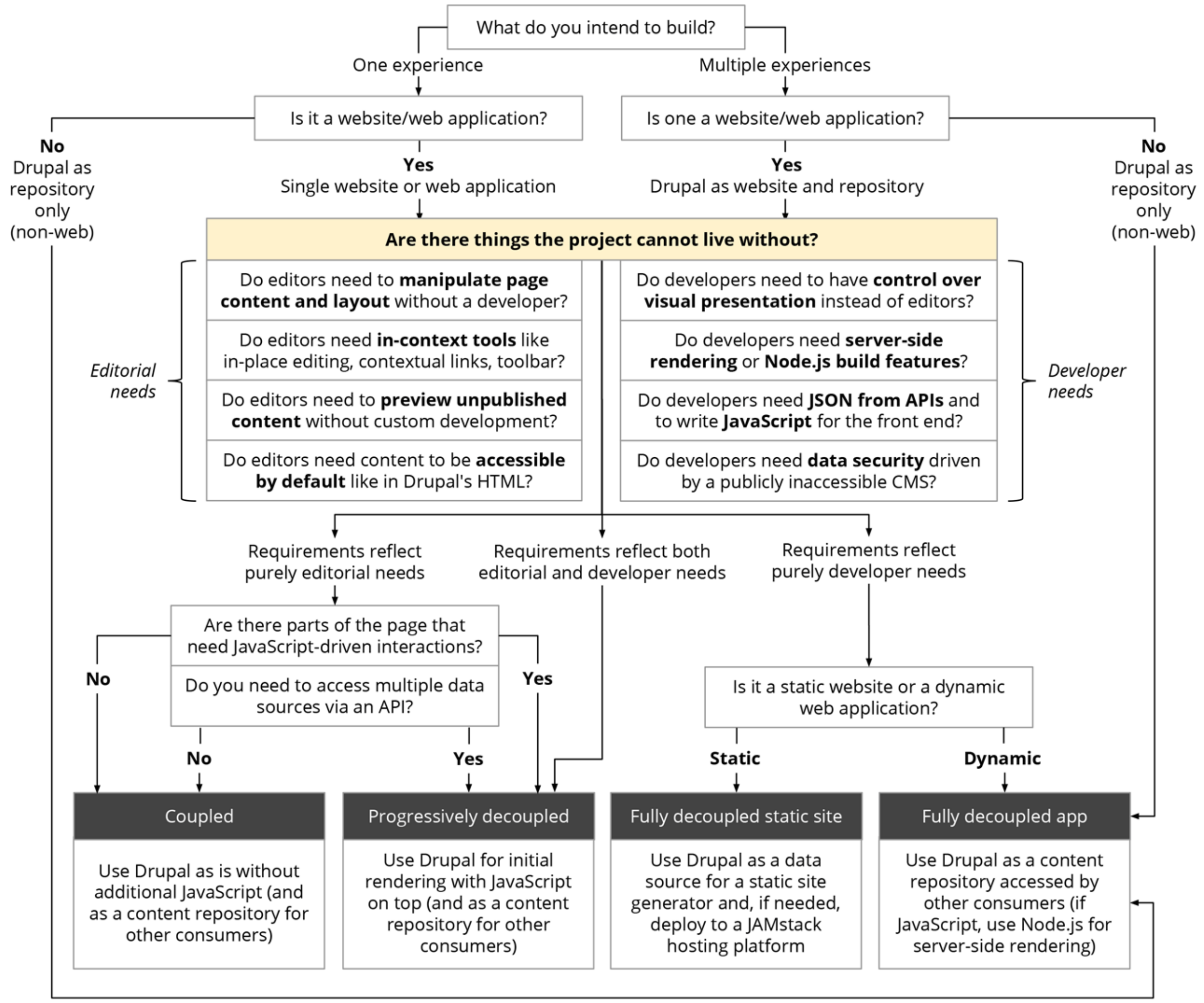


Under progressive decoupling, the CMS renderer still outputs the skeleton of the page.

## PROGRESSIVELY DECOUPLED

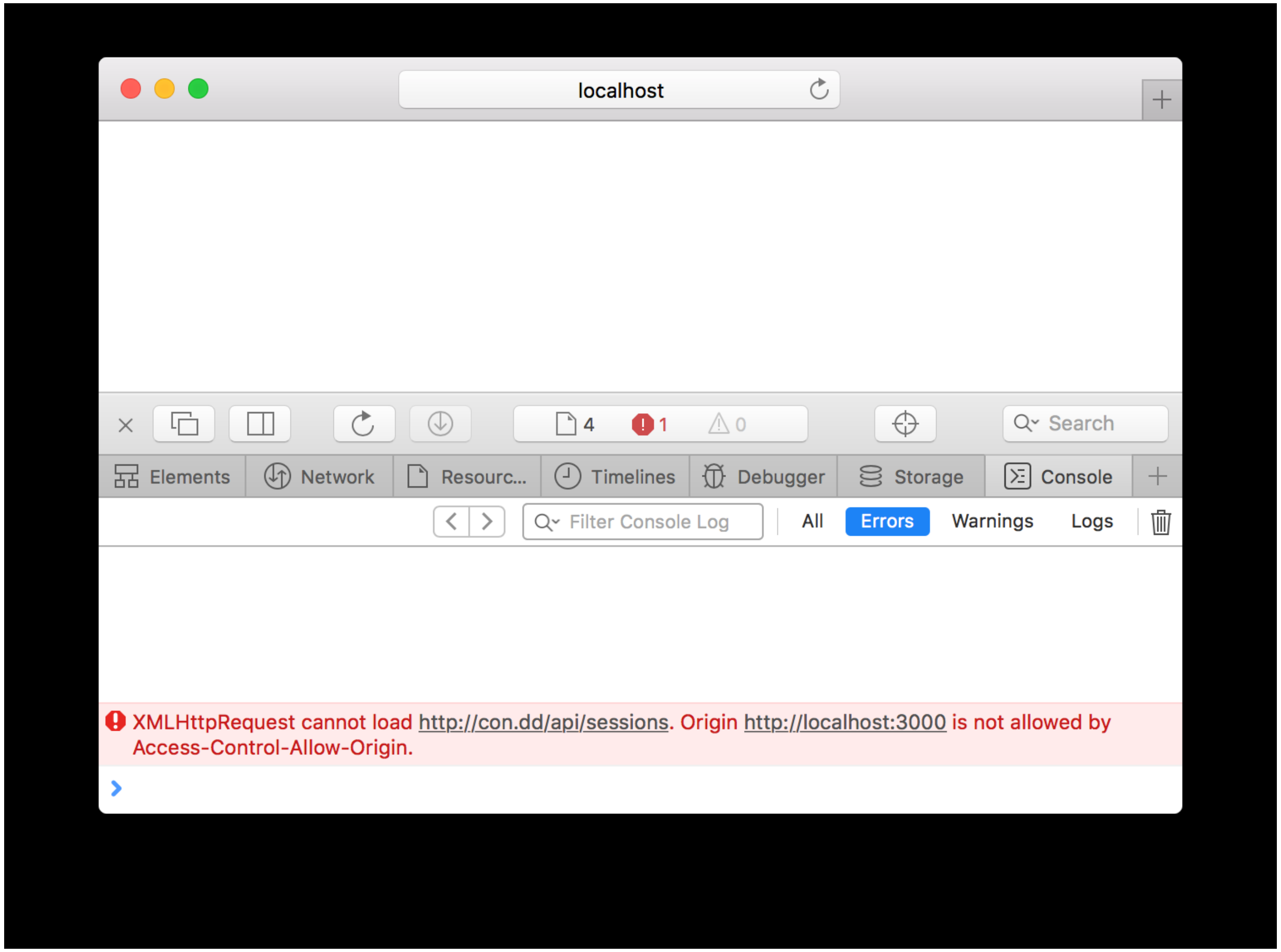
- If the JavaScript frame work (react/angular) and Drupal theme used together to present the content then this is called the progressive decoupled
- Drupal theme, is responsible for outputting the primary user experience, and React is used to enhance it
- Using power of Drupal CMS + Javascript framework
- Rather than decupling entire page, a part of page or it’s blocks were decoupled as Progressive decoupled Drupal which able to offer best of both Drupal 8 and Front End JS frameworks

# Deciding Path for Decoupled Drupal 8


What do you intend to build?
One experience
Multiple experiences
Is it a website/web application?
Is one a website/web application?
No
Drupal as repository only (non-web)
Yes
Single website or web application
Yes
Drupal as website and repository
No
Drupal as repository only (non-web)
Are there things the project cannot live without?
Editorial needs
Do editors need to manipulate page content and layout without a developer?
Do editors need in-context tools like in-place editing, contextual links, toolbar?
Do editors need to preview unpublished content without custom development?
Do editors need content to be accessible by default like in Drupal's HTML?
Developer needs
Do developers need to have control over visual presentation instead of editors?
Do developers need server-side rendering or Node.js build features?
Do developers need JSON from APIs and to write JavaScript for the front end?
Do developers need data security driven by a publicly inaccessible CMS?
Requirements reflect purely editorial needs
Requirements reflect both editorial and developer needs
Requirements reflect purely developer needs
Are there parts of the page that need JavaScript-driven interactions?
Do you need to access multiple data sources via an API?
No
Yes
No
Yes
Is it a static website or a dynamic web application?
Static
Dynamic
Coupled
Use Drupal as is without additional JavaScript (and as a content repository for other consumers)
Progressively decoupled
Use Drupal for initial rendering with JavaScript on top (and as a content repository for other consumers)
Fully decoupled static site
Use Drupal as a data source for a static site generator and, if needed, deploy to a JAMstack hosting platform
Fully decoupled app
Use Drupal as a content repository accessed by other consumers (if JavaScript, use Node.js for server-side rendering)

**Reference**: https://dri.es/how-to-decouple-drupal-in-2019

## Cross Origin Resource Sharing (CORS)



```
XMLHttpRequest cannot load http://con.dd/api/sessions. Origin http://localhost:3000 is not allowed by Access-Control-Allow-Origin.
```

## Drupal 8 - CORS

- API that runs in the browser we need to make sure our Drupal back end supports CORS.
- CORS is a security feature that all browsers implement to ensure access control for cross-origin request
- Drupal 8 core provides support for CORS since 8.2, but it's turned off by default.
- To turn it on we will need to edit our sites/default/services.yml file.

# Progressively Decoupled React Application

## ReactJS

- Not full MVC, only View.
- Uses "Virtual DOM" for performance.
- Lets you build components that can be injected DOM and updated when state changes.
- Best used with ES6 or ES2015 which compiles to JS

## Modern JS

- Drupal core now using ES6 for JavaScript development.
- ES6 is also known as ECMAScript 6 and ECMAScript 2015.
- ECMAScript (or ES) is a trademarked scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262
- In Drupal 8, jQuery is no longer required.
- The **fetch, promises** are now part of modern versions of JavaScript.

## DECONSTRUCTING ASSIGNMENT

- Destructuring assignment is a special syntax that allows us to “unpack” arrays or objects into a bunch of variables.

```
let options = { title: "Menu", width: 100, height: 200 };
let {title, width, height} = options;
alert(title); // Menu
```

## PROPERTY SHORTHAND

- With ES6, you can use shorthand property names which allow you to write something like this.

```
var o = { s, n }; // This is equivalent to { s: s, n: n }
```

## SPREAD SYNTAX

- The spread syntax is simply three dots: ...
- It allows an iterable to expand in places where 0+ arguments are expected.
- Definitions are tough without context.

```
var mid = [3, 4];
var arr = [1, 2, ...mid, 5, 6];

console.log(arr); //[1, 2, 3, 4, 5, 6]
```

## Arrow functions

- Arrow functions – also called “fat arrow” functions
- Arrow functions are anonymous
- We avoid having to the type function keyword, return keyword and curly brackets.

```
// ES5
var multiplyES5 = function(x, y) { return x * y; };
// ES6
const multiplyES6 = (x, y) => { return x * y };
```

## ES5 - CALLBACK

```
$.ajax({
  url: 'https://dev-decouple/api/article',
  success: function (article) {
    $.ajax({
        url: 'https://dev-decouple/api/node',
        success: function (node) {
        }
        );
       }
       );
```

## ES6 - CALLBACK

- The fetch() function lets you send network requests and get responses. It uses promises to allow for asynchronous requesting and processing of data
- . fetch is easier and cleaner to deal with than the longstanding XHR (XMLHttpRequest)

```
fetch(url, options)
   .then(response => response.json())
   .then(result => console.log('success', result))
   .catch(error => console.log('error', error));
```

## Progressively Decoupled React

Create "Hello, World" React application.

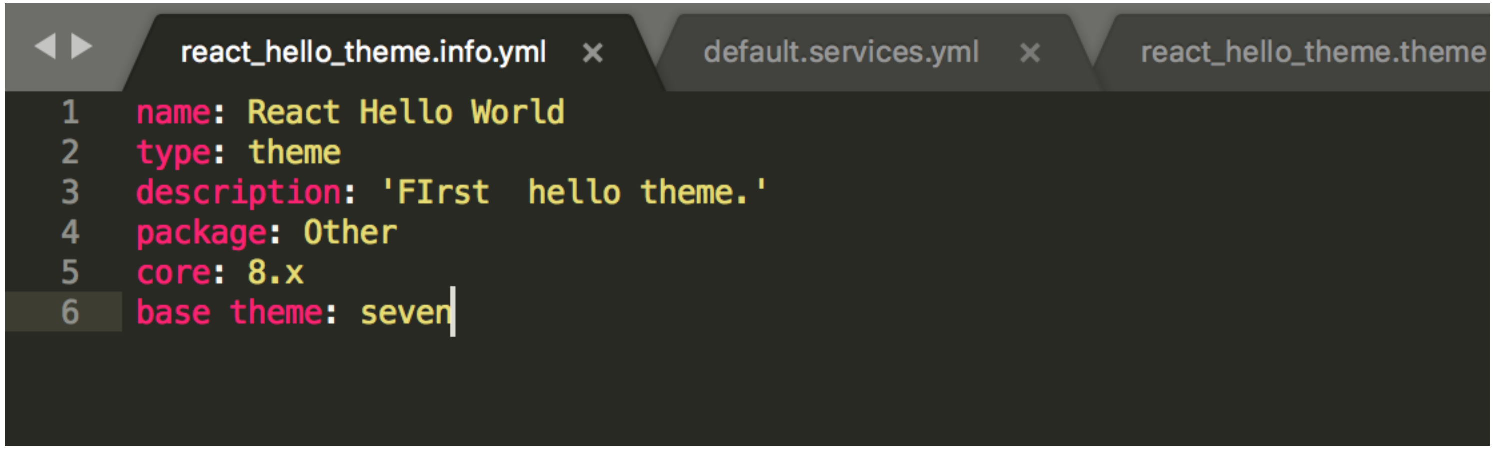
1. **Create a theme**

Using the **Drupal Console** we could create a module using the following instruction:

```
$ drupal generate:theme –theme=“React Hello World" --
machine-name=“react_hello_theme" --theme-path
=“/themes/custom” –description= “First hello theme” --
core="8.x“--base-theme =“seven”
```

## Progressively Decoupled React

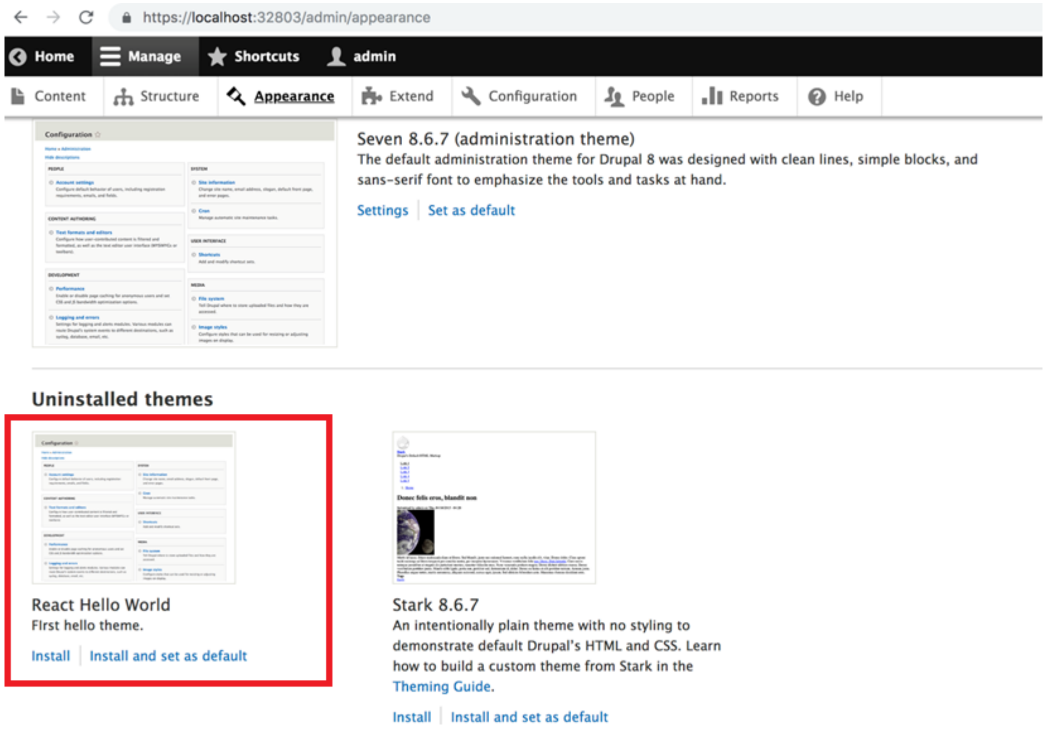
**2. Go to** go to */themes.*


react_hello_theme.info.yml ×
default.services.yml ×
react_hello_theme.theme

```yaml
name: React Hello World
type: theme
description: 'FIrst  hello theme.'
package: Other
core: 8.x
base theme: seven
```

## Progressively Decoupled React

3. Enable the theme **React Hello World in Drupal UI**.



Seven 8.6.7 (administration theme)
The default administration theme for Drupal 8 was designed with clean lines, simple blocks, and sans-serif font to emphasize the tools and tasks at hand.

Settings | Set as default

Uninstalled themes

React Hello World
First hello theme.

Install | Install and set as default

Stark 8.6.7
An intentionally plain theme with no styling to demonstrate default Drupal's HTML and CSS. Learn how to build a custom theme from Stark in the Theming Guide.

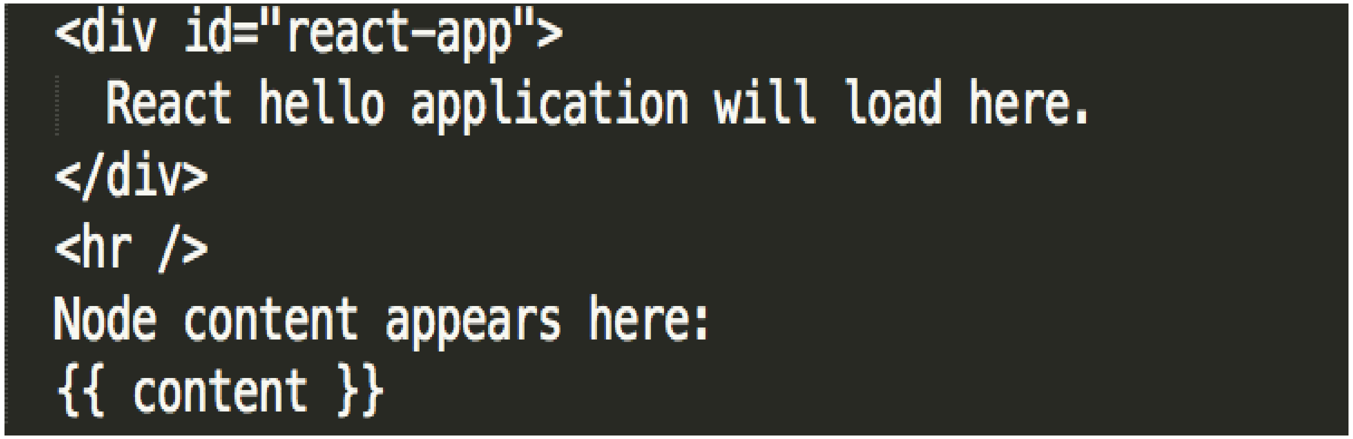
Install | Install and set as default

## Progressively Decoupled React

4. Copy *node.html.twig in templates folder*

5. Add the below code to our theme's Twig file, near the end, near the {{ content }} tag

.



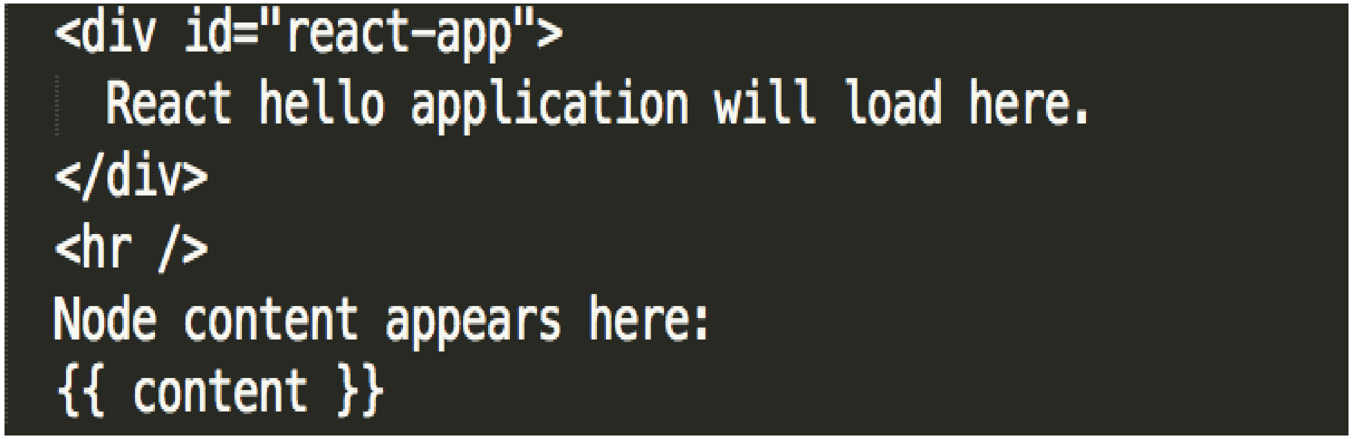
```
<div id="react-app">
  React hello application will load here.
</div>
<hr />
Node content appears here:
{{ content }}
```

## Progressively Decoupled React

4. Copy *node.html.twig in templates folder*

5. Add the below code to our theme's Twig file, near the end, near the {{ content }} tag

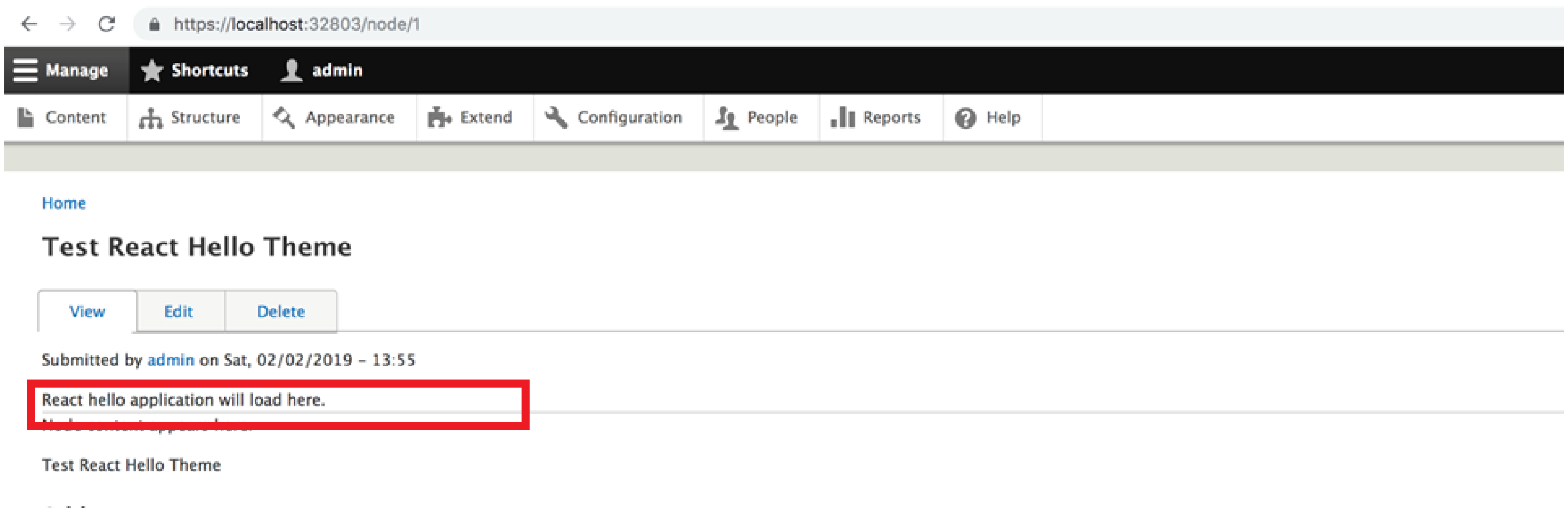
.



```
<div id="react-app">
  React hello application will load here.
</div>
<hr />
Node content appears here:
{{ content }}
```

## Progressively Decoupled React

6. Create some test content and see we have "React hello application will load here"


https://localhost:32803/node/1

Test React Hello Theme

Submitted by admin on Sat, 02/02/2019 - 13:55

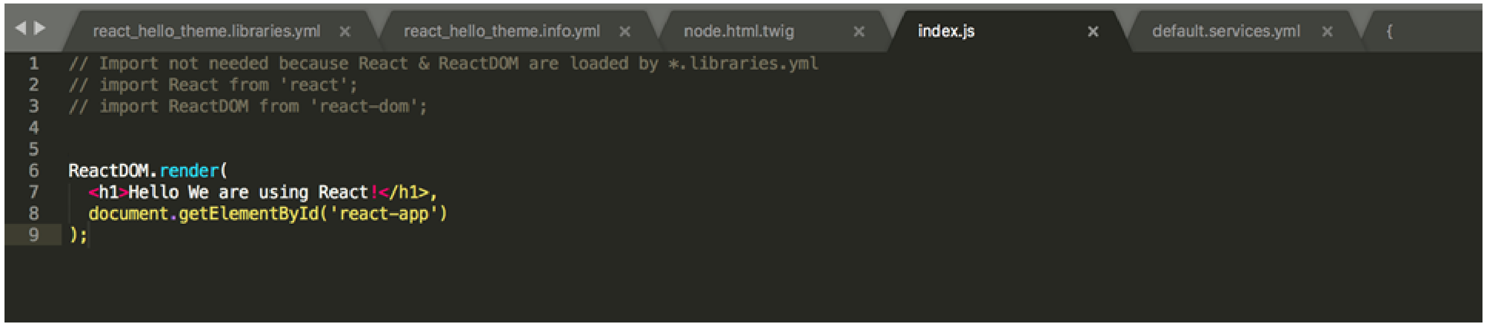
React hello application will load here.

Test React Hello Theme

## Progressively Decoupled React

7. Create a new file called *index.js*: */themes/* react_hello_theme/*js/index.js*.

.



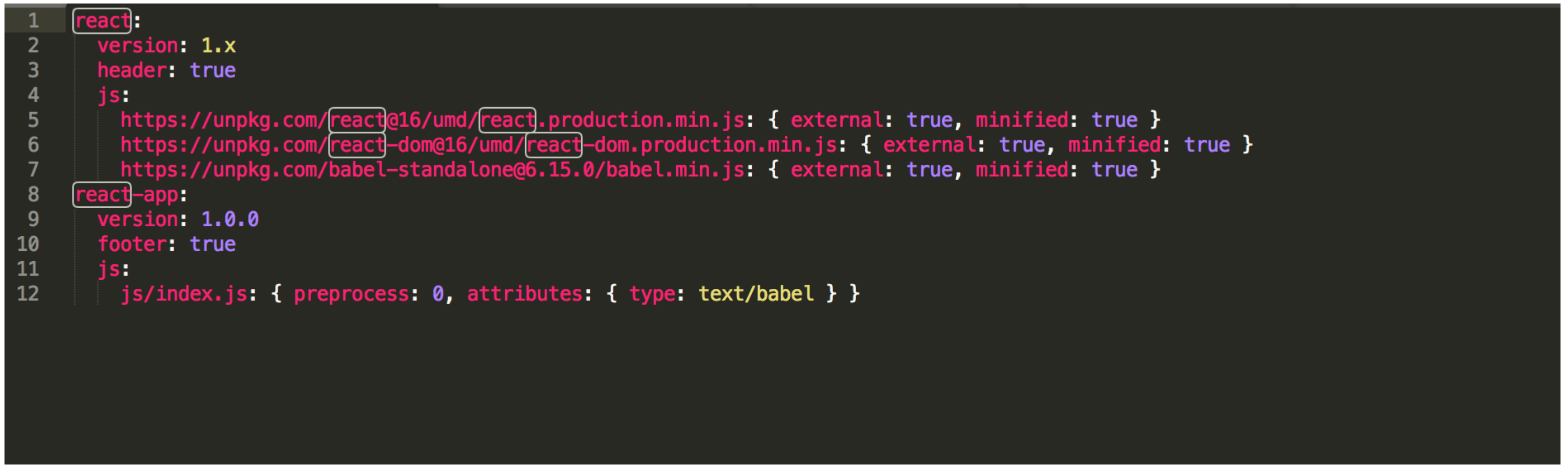
react_hello_theme.libraries.yml × react_hello_theme.info.yml × node.html.twig × index.js × default.services.yml × {

```
// Import not needed because React & ReactDOM are loaded by *.libraries.yml
// import React from 'react';
// import ReactDOM from 'react-dom';


ReactDOM.render(
  <h1>Hello We are using React!</h1>,
  document.getElementById('react-app')
);
```

## PROGRESSIVELY DECOUPLED REACT

8. Create a new file in the theme folder (*.*libraries.yml*): /*themes*/react_hello_theme/react_hello_theme.*libraries.yml*.



```yaml
react:
  version: 1.x
  header: true
  js:
    https://unpkg.com/react@16/umd/react.production.min.js: { external: true, minified: true }
    https://unpkg.com/react-dom@16/umd/react-dom.production.min.js: { external: true, minified: true }
    https://unpkg.com/babel-standalone@6.15.0/babel.min.js: { external: true, minified: true }
react-app:
  version: 1.0.0
  footer: true
  js:
    js/index.js: { preprocess: 0, attributes: { type: text/babel } }
```

## Decoupled React Application

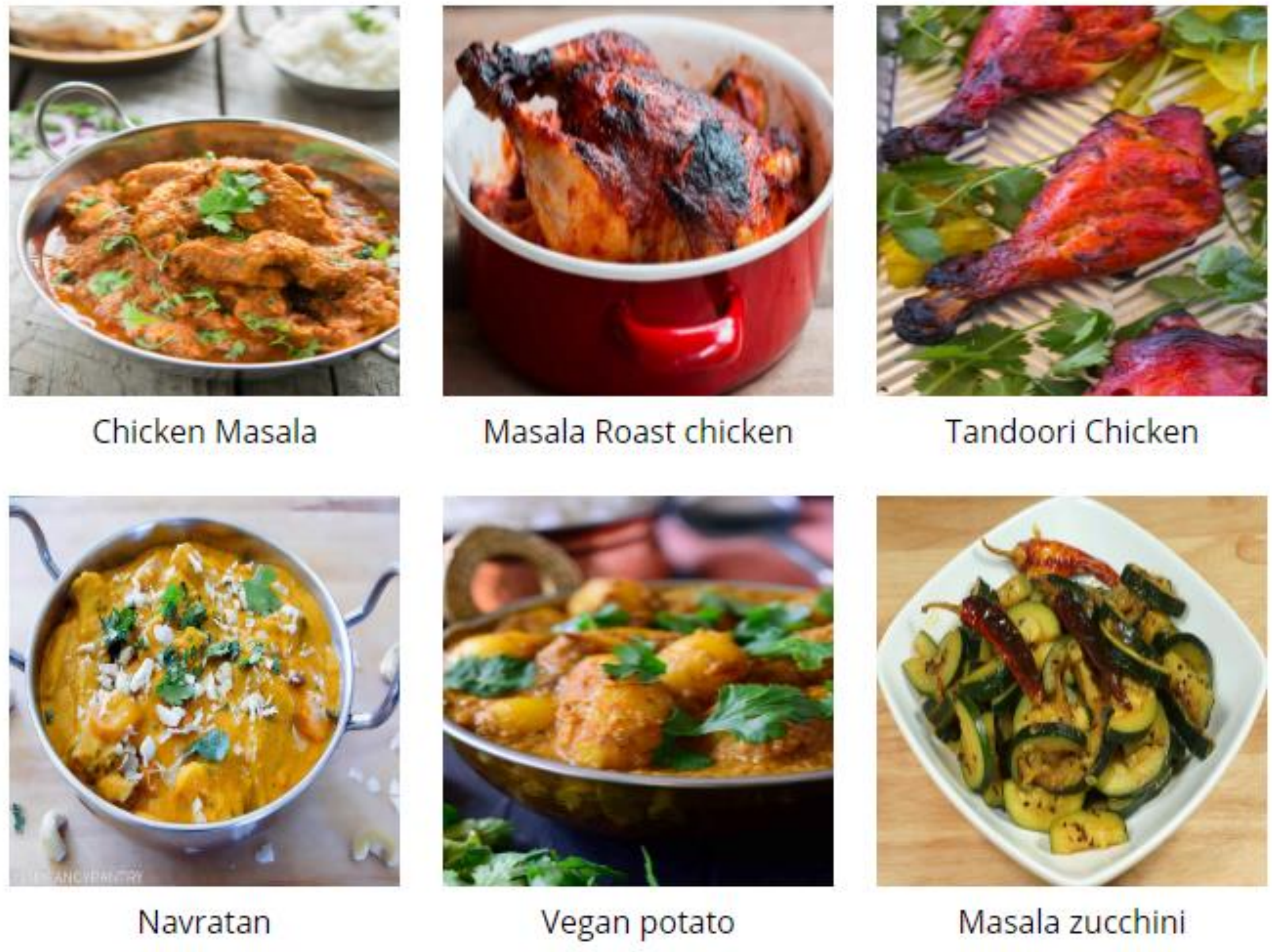
## JSON API

- Install the JSON API module
  https://www.drupal.org/project/jsonapi
- The JSON API module is a zero configuration module.
- JSON API exposes all the bundles for all the entity types via a modern REST specification.
- Drupal's entity access respect for all the operations.
- Request to the entry point in JSON API you will get a list of links indicating where all the resources are situated

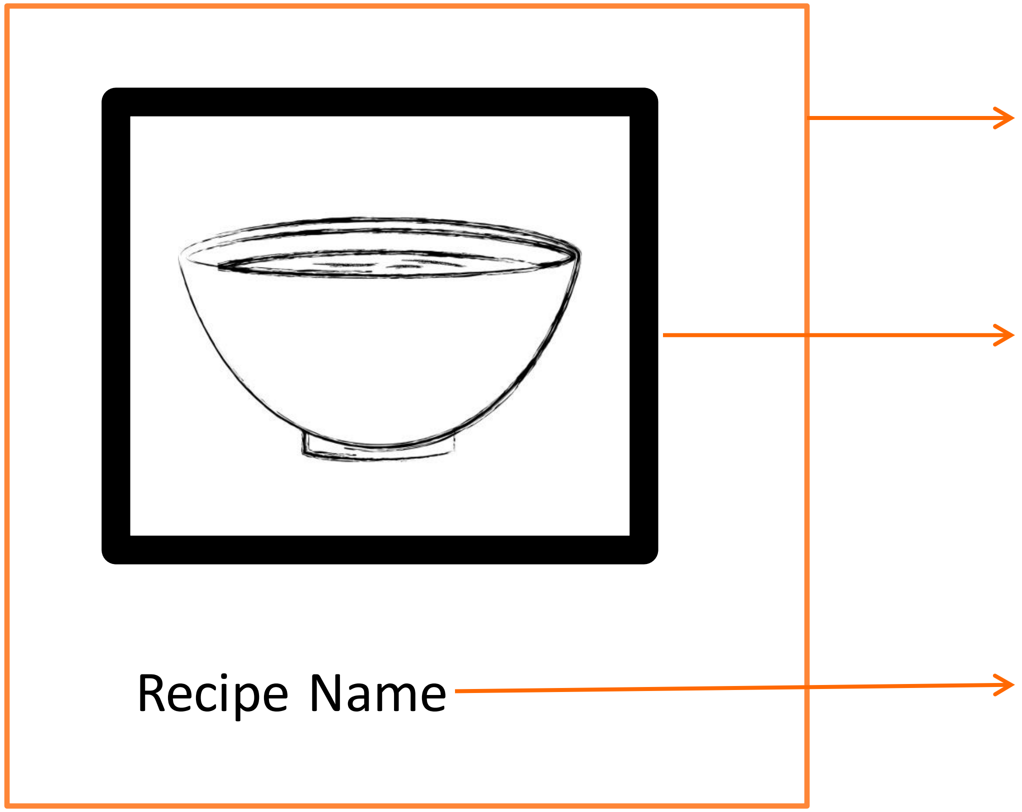
  https://example.org/jsonapi

  It is called the entry point because it is the place where we start to learn what other resources are available.

# Decoupled React application



# Component Planning



Recipe Card Container which contain the link to actual recipe page

Image Component which will hold the recipe picture

Title Component which will hold the recipe's name

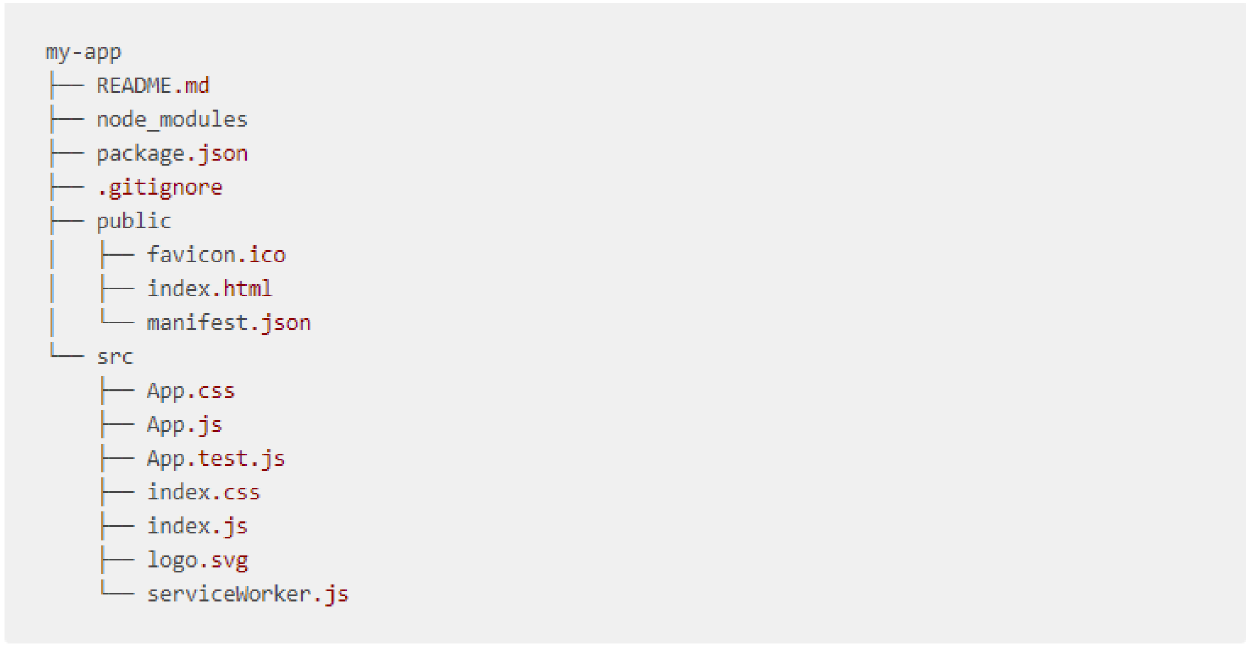
## Fully Decoupled React Application

- Use **create-react-app** to scaffold a new React project

```
npx create-react-app my-app
yarn create react-app my-app
```

- It will generate the initial project structure

## Fully Decoupled React Application



```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

For the project to build, **these files must exist with exact filenames**:

- public/index.html is the page template;
- src/index.js is the JavaScript entry point.

You can delete or rename the other files

## EDITORIAL PAIN POINTS

- Page refreshes
- Less features are available for the content editing and creation as compared to other editorial tools.
- Less interactive and slow as compared to other editorial tools

## Empowering Editorial UI

## EMPOWERING THE EDITORIAL UI

- Editorial tools (openstory.io, Gutenberg, Glazed) provides the enrich features to editor for live editing of content.
- Provides more flexibility and ease for content creation.
- Happy Editors leads to Better Content and as in today's world content is Everything.
- Live URL's for demo
  - https://drupalgutenberg.org/demo
  - https://www.sooperthemes.com/drupal-modules/glazed-builder

## Editorial tools

## EDITORIAL TOOLS

- Gutenberg
  - Great for content authors
  - Drupal Gutenberg brings the powerful admin features of Gutenberg Editor to Drupal.
  - Scaleable, high performance
  - Reusable blocks created, saved and reused within the editor (no code needed)
  - Core Drupal blocks support
  - Live demo URL : https://drupalgutenberg.org/demo
  - FAQ : https://www.drupal.org/docs/8/modules/gutenberg/faq

## EDITORIAL TOOLS

- Glazed Builder
  - Drupal Drag and Drop Page Builder
  - Create Content 5x Faster With Glazed Builder
  - Video URL : https://youtu.be/4Lu1UZiO_ZM
  - https://www.sooperthemes.com/drupal-modules/glazed-builder

## EDITORIAL TOOLS

- Openstory.io
  - friendly content creation & media management interface
  - It’s an open source editorial interface for Drupal and Wordpress
  - Decoupled interface specially created to simplify content creation!
  - Created with Angular and NodeJS
  - https://demo.openstory.io/

# Q&A

## Thank You