###### File:React-icon.svg - Wikimedia CommonsSitecore – Add missing icons for specific media types – Vincent – Technologist

**ALTUDO HACKATHON 2022**



**Topic**

**JSS IMPLEMENTATION ON   
PORTFOLIO WEBSITE**

|  |  |
| --- | --- |
| **Project Name** | : Personal Dynamic Website |
| **Team Members** | : Chirag Goel {ALT/IND/8033},  Aman Kumar {ALT/IND/8024},  Prince Kumar {ALT/IND/8019}, |
| **Date** | : 07/01/2022 |
|  |  |

**Introduction**

Sitecore is a leading digital experience software used by organisations globally to create seamless, personalised digital experiences. Sitecore is an integrated platform powered by .net CMS, commerce and digital marketing tools.

Sitecore JSS stands for Sitecore JavaScript Services, Sitecore JavaScript Services allow you to develop against the Sitecore Experience Platform in decoupled JavaScript applications.

Sitecore provides a suite of services, APIs, and software development kits (SDKs) that allow you to develop decoupled applications using Sitecore data, transforming Sitecore into a headless content management system (CMS). **Sitecore JSS supports developing React applications with Sitecore.**

In this project we have used Sitecore JSS headless with reactJS to create a dynamic portfolio website that is completely editable from the sitecore itself.

**Working with custom built components**

Sitecore JSS bundles .net code with react based components to create user editable web pages. In our project we have created multiple custom built components in react and binded them with sitecore using props functionality.

Creating components is very useful while building large projects since it helps the developers to reuse code multiple times. One single component can be used in many scenarios to produce desire output.

**Building components**

We used frontend technology ReactJS for creating frontend designs. React is a Javascript library that is used to create modern UI designs. The components created with react library can be consumed by any headless based backend technology.

All the components are created to consume dynamic data. These data will be further provided by our Sitecore.

**Binding components**

When the required components are created the next step is to bind these components with sitecore. Binding of the components is required at both ends from backend and frontend too.

In react components we use props.fields entity to bind the field to the sitecore field.

In sitecore the fields are named and the user inputs the data that data is being received by component as props.

Binding is a very important step because any mistake can result in wrong data collection.

**Deploying components**

Deploying of components are done by building a production build of the project that contains all the necessary files to make sure it works on sitecore.

Production build is done using ----. This gives us a stable, clean and minified version of the all the components. We cannot deploy open or editable code on the sitecore environment since it will not be recognized by sitecore.

**Rendering Parameters and Creating Components**

We in this project have also used rendering parameter for a component named Promo. It has capability to switch the Image on left or right section according to need of the User. And all the components are editable from experience editor like images, text, Links etc.

Moreover all of our components are reusable. We have kept the header navigation as static as these fields only navigate to particular section of the page and moreover these sections are necessary for any portfolio.

**Resultant output**

After developing and deploying we successfully created a webpage that is completely built in sitecore headless using reactJS. This webpage supports dynamic edit from sitecore.

Here are some sample screenshots of the website:









