**What is Code Splitting in frontend application**

Code spliting used to split our code into various bundle, which can be loaded on demand. With code spliting it makes our website have a faster load time.

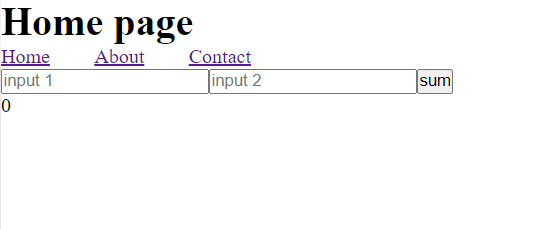
When we want to deploy our web app, we need to bundling our frontend app so it can be download by the client. Imagine if we bundle all the entire our frontend app, and let the user (client) download entire content in it even they don’t really need to check all the content inside. For example if we create an app that have many services like purchase, catalog, etc. and the user accessing our web app only want to explore the catalog parts, but since our frontend app only have one bundle that contain the entire app, the user download all of them only to access catalog.

According to the example, we know that it can make our website become slow when the user try to access it because the client need to download the entire frontend app even they don’t realy use all the content inside. This is why we need to implement a code spliting in our frontend app. With code spliting, we sparate the service into smaller bundle, so client will only download all the resources that they need (when they try to access it). So when the user access our web app at the first time, it will only download the home page, and when they get into the catalog service, its only download the catalog part, and so on.

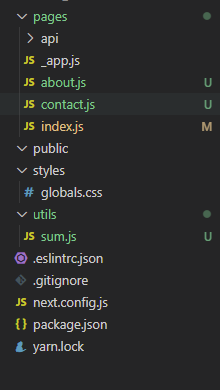
Three general approaches to code splitting:

* Entry Points
* Prevent Duplication
* Dynamic Imports (lazy load)

Currently we going to use dynamic import approach



I built a simple app with this sum functionality.



I put the sum function inside utils/sum.js and export it as a module

Graphical user interface, application

Description automatically generated

As we can see, currently all the entire app is downloaded into the client side

**Dynamic Imports**

Dynamic import is using import() function from ES6. This allows load a module asynchronously and dynamic into a non module environment (client browser).

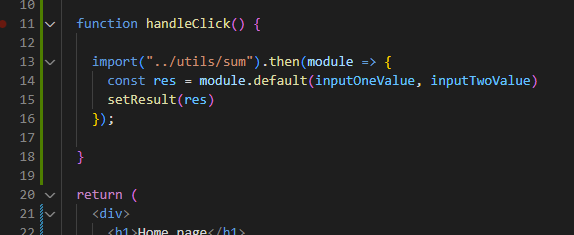
Text

Description automatically generated

Currently, we import all of our module and component on the top of our index.js (Home view) script. Lets take a look the sum module.



We import it on the top of our script that’s why our sum module already there even we don’t use it



Instead of that, we want to import our sum module inside the handleClick. So its only going to be imported when the sum button is clicked 

This is the result

Graphical user interface, application

Description automatically generated

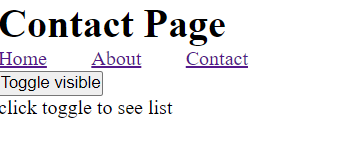
Because the sum and utils is not utilized at all until we click the sum function, its not going to be downloaded

Graphical user interface, application

Description automatically generated

And after we click the sum button,  and run the handleClick function, the utils and sum.js is appearing inside sources. It means that the utils and sum.js is downloaded when we need them.

We can also do the same thing with the component



Inside the contact page, I add the toggle visible button and when its clicked, it will show the contact list. The contact list I put inside sparated component named ContactList.

Text

Description automatically generated

Now inside our code source, use the lazy function to achive the dynamics import for component

Text

Description automatically generated

Wrap it with Suspense element with the fallback event. The fallback will shown when the user try to access ContactList, and need to download the component first (because its lazy load)

A picture containing application

Description automatically generated

As we can see inside the components folder only contains nav component.

A picture containing logo

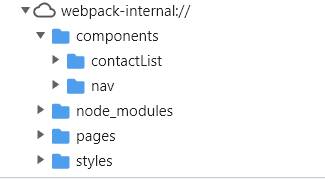
Description automatically generated

We click the toggle visible button, and it will show loading while download the component (how long the Loading will remains is depends on the client internet speed)

A picture containing text

Description automatically generated

After the content is shown



We can see that the contactList component is there

**Responsive Component**

For example we have a web app with mobile first view. It means we develop the app started from the mobile view. We want to separate the mobile and the desktop view, so its detect the client side wether it is mobile or desktop, then we render the component based on what is the client side using. What if we want to add a carousel into our app? Do we need a library to handle this? For the mobile we know that we don’t really need library to make a carrousel to work because its only need to swipe left and right to see the content. Different with the desktop that need a library to make us easier to move the carousel left and right.

As we can see if we don’t implement code splitting¸ the client just download all the entire code source and the module even they don’t really use it because its only need to render a mobile component. With code splitting, the user will only download the resources that need for the mobile app to work, without need to download all entire module that needed for the web app.

**Source**

<https://github.com/Gerardy01/code-splitting>

**References**

<https://www.youtube.com/watch?v=SnV8GSitGVs&ab_channel=WebProgrammingUNPAS>

<https://www.youtube.com/watch?v=JU6sl_yyZqs&ab_channel=WebDevSimplified>