

The background of the slide features a large, semi-transparent NASA logo. The logo consists of a blue circular field with white stars and a red swoosh. The word "NASA" is written in large, white, serif capital letters across the center of the logo. The entire slide has a light blue background.

# Single Page Application using the NASA API

---

Create a Single Page Application that relies heavily on one or more APIs

<https://sonnerz.github.io/project02-interactive-frontend/>

<https://github.com/Sonnerz/project02-interactive-frontend>

JUNE 20 2018

---

COOLE SITES

Authored by: Sonya Cooley

In Progress

---

# Single Page Application

## Create a Single Page Application that relies heavily on one or more APIs

Provide search results in a manner that is visually appealing for your user (by drawing on the skills you have learned in User-Centric Frontend Development)

## Guidelines and Guideline fulfilment

### 1. Create a Single-Page Application (SPA)

I have created a Single-Page Application (SPA) divided into six distinctive sections

### 2. Incorporate links or buttons to allow your user to navigate the site and reset/control the site functionality

I have provided a 'sticky' navbar at the top of the web page. This means that navigation is always available to the site user. Buttons to search and clear API results are also provided in the content sections.

### 3. Whenever possible, strive to use semantic HTML5 elements to structure your HTML code better.

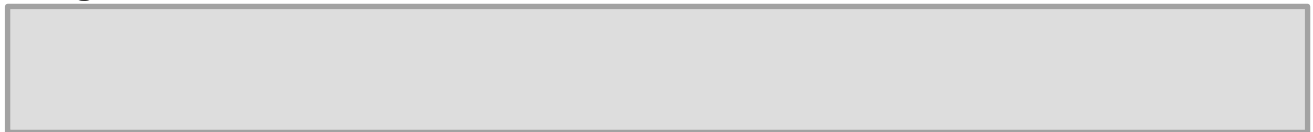
Semantic HTML5 elements are used throughout to structure the html.

### 4. Make sure your site is as responsive as possible. You can test this by checking the site on different screen sizes and browsers.

Please note that if you are building a data dashboard, only your chart containers are expected to be responsive. Charts using D3.js are not responsive as they are designed for desktop or large-screen viewing.

The site has been built to be as responsive as possible. Please see the [HTML/CSS Structure](#)

## Navigation

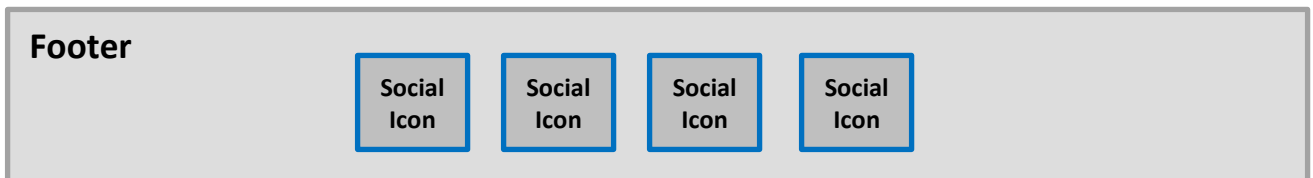


The navigation is responsive to the type of client device a user is using to view the site.

Any view below 992px wide will get a collapsing Navigation Bar. The navigation links will have a hover state making it obvious to users which navigation link they are hovering over. The text will turn the official NASA blue: #0B3D91

The collapsed navigation will have a similar hover state.

## Footer



---

The footer was developed/designed to be full width and contain links to NASA sites including their social media.

## **Content Sections**

The SPA consists of one page – index.html. The page was created using Bootstrap 4. A hero image of Atlantis is the first image users see when they visit the site. The image is owned by the web developer.

The page is divided into six distinct areas, each with its own purpose; welcome, apod, epic, library, data, contact.

### **Welcome section**

The welcome section welcomes the visitor to the site, it invites the visitor to use each section and briefly explains the purpose of that section.

It will be a < header > with a large background image and text in a < p >.

### **APOD – Astronomy Picture of the Day**

APOD, makes a call to the APOD API returning the NASA chosen image/video of the day and a brief explanation provided by a professional astronomer.

The HTML < section > has a title and explanation row filling the width of the page. The image and text are displayed in two columns. If the APOD is a video, it will take up one row while its text will take up another row below the video.

### **See the APOD Wireframe**

### **EPIC**

The visitor has the option to view the most recent EPIC image received or chose a date and image type and send that date and image type to the EPIC API. The API returns images taken by the EPIC camera on that date. The list of images is paged, one image per page.

If the visitor chooses to view the most recent EPIC image then only one image is returned. It will be the last image received from DSCOVR on the previous day.

EPIC is a HTML < section >. The search part is in a div containing descriptive text, an input field for choosing a date, radio choices for natural/enhanced image and a search button.

When the user clicks the Search button, a < div > displays showing the results in two columns. The image in the left column and its text in the right column. Paging buttons also appear with the API results.

**The most recent image API call displays the result in the same way as the Date search option.**

See the EPIC Wireframe

---

## Library – Search NASA Image and Video Library

The first part of the library < section > takes a text query and media type chosen by the site visitor and sends that query and media type to the library API. The API returns results in lots of 100 items. The user can page through the results.

**After the user clicks the Search Button, a div for results appears below the search input area. It shows rows of the search results. Each result has an image in the left column while the text appears in the right column.**

Paging buttons are at the top of the results section.

See the Library Wireframe

## Data

The NASA Facilities dataset is made available by NASA as a json file. This section takes the data and using dc/d3/crossfilter renders pie charts and bar charts.

## Colours

#050215	rgba(5, 2, 21)	Blue	
#0B3D91	rgba(11, 61, 145)	NASA Blue	
#FC3D21	rgba(252, 61, 33)	NASA Red	

## SCSS/CSS

Bootstrap 4 provides the fundamental CSS and JavaScript for the entire site.

However, custom styles have been created and a mainStyles.css file can be found in the assets/css directory.

Custom styles for each content section, and navigation are in their own SCSS file.

mainStyles.css is created from SCSS files.

apod.scss	bootstrap-overrides.scss
data.scss	epic.scss
Global.scss	intro.scss
library.scss	mainStyles.scss
mixins.scss	navbar.scss
variables.scss	

## JavaScript

A JavaScript file has been created for each content section. Each section has its own API call.

NASA prefers that developers sign up for a developer key. This key is used in every API call.

nasaAPI-apod.js	nasaAPI-data.js
nasaAPI-library.js	nasaAPI-epic.js
helperTools.js	

### APOD: nasaAPI-apod.js

This API call uses AJAX.

Returned from API <ul style="list-style-type: none"><li>– copyright</li><li>– date</li><li>– explanation</li><li>– hdurl</li><li>– media_type</li><li>– service_version</li><li>– title</li><li>– url</li></ul>	<pre>{copyright: "Jingyi Zhang", date: "2018-06-19", explanation: "They m e...umans. Almost Hyperspace: Random APOD Generator", hdurl: "https tromatolites_Zhang_1587.jpg", media_type: "image", ...} copyright: "Jingyi Zhang" date: "2018-06-19" explanation: "They may look like round rocks, but they're alive. Mc hdurl: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites media_type: "image" service_version: "v1" title: "Ancients of Sea and Sky" url: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites_2 __proto__: Object</pre>
---	---

<pre>var url = "https://api.nasa.gov/planetary/apod?api_key=pyZKDq8 cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAxZ";  \$.ajax({   url: url,   success: function(result) {     ...   });</pre>	<p>The url provided by NASA takes an API key.</p> <p>An AJAX call is made using the url and the results are written out to the HTML page.</p>
--	---

## EPIC: nasaAPI-epic.js

This API call uses FETCH.

This content section will not work in Internet Explorer. FETCH is not supported by IE.

Returned from API <ul style="list-style-type: none"><li>attitude_quaternions</li><li>caption</li><li>centroid_coordinates</li><li>coords</li><li>date</li><li>dscovr_j2000_position</li><li>identifier</li><li>image</li><li>lunar_j2000_position</li><li>sun_j2000_position</li><li>version</li></ul>	<pre>▼ (12) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ   ▼ 0:     ▶ attitude_quaternions: {q0: 0.009435, q1: 0.614245, q2: -0.772655, q3: 0.160055}     ▶ caption: "This image was taken by the NASA EPIC camera onboard the NOAA DSCOVR spac     ▶ centroid_coordinates: {lat: 12.187576, lon: 156.869864}     ▶ coords: {centroid_coordinates: {...}, dscovr_j2000_position: {...}, lunar_j2000_positic     ▶ date: "2017-04-15 00:50:27"     ▶ dscovr_j2000_position: {x: 1369514.38358, y: 340216.540237, z: 304757.805862}     ▶ identifier: "20170415005515"     ▶ image: "epic_1b_20170415005515"     ▶ lunar_j2000_position: {x: -158260.421883, y: -355114.669844, z: -114884.177393}     ▶ sun_j2000_position: {x: 135998314.92302, y: 58228275.399883, z: 25241919.575161}     ▶ version: "02"     ▶ __proto__: Object   ▶ 1: {identifier: "20170415024318", caption: "This image was taken by the NASA EPIC car</pre>
--	---

<a href="https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h">https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h</a> <a href="https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ">https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ</a>	
getEpicImageByDate()	<ul style="list-style-type: none"><li>Function called from Search button on the HTML page.</li><li>Date and image type taken from visitor input form</li><li>If image type is <b>enhanced</b> then a call is made to EPIC API using <b>FETCH</b> and enhanced url. A JSON Object is returned and parsed.</li><li>If image type is <b>natural</b> then a call is made to EPIC API using <b>FETCH</b> and natural url. A JSON Object is returned and parsed.</li><li>The JSON object is sent to pageTheResult() for slicing into 1 item per page. 1 array item per page.</li><li>The sliced array is sent to getResultItems() for looping and rendering to HTML</li></ul>
getResultItems()	<ul style="list-style-type: none"><li>Data items from API result are rendered to HTML in a forEach loop</li></ul>
pageTheResult()	<ul style="list-style-type: none"><li>Paging the JSON Object to one item per page using JavaScript Array slice() Method</li></ul>
getNumberOfPages()	<ul style="list-style-type: none"><li>Retrieve the total number of items returned from API</li></ul>
nextPage()	<ul style="list-style-type: none"><li>Function called from Next Button</li></ul>
previousPage()	<ul style="list-style-type: none"><li>Function called from Previous Button</li></ul>



API documentation provided by NASA: <a href="https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf">https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf</a> <a href="https://images-api.nasa.gov/search?q={q}">https://images-api.nasa.gov/search?q={q}</a>	
getQueryText()	<ul style="list-style-type: none"> <li>• Function called from Search button on the HTML page.</li> <li>• Query text and media type taken from visitor input form</li> <li>• The url is build using query text and media type</li> <li>• The url is sent to searchNASALibrary()</li> </ul>
searchNASALibrary()	<ul style="list-style-type: none"> <li>• Call made to NASA Library API using XMLHttpRequest()</li> <li>• If result is image then the JSON object and media type is sent to getLibraryResultsDataImage()</li> <li>• If result is audio then the JSON object and media type is sent to getLibraryResultsDataAudio()</li> <li>• If result is video then the JSON object and media type is sent to getLibraryResultsDataVideo()</li> <li>• Each media type required different html rendering hence the 3 different functions</li> </ul>
getLibraryResultsDataImage()	<ul style="list-style-type: none"> <li>• Processes image results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataAudio()	<ul style="list-style-type: none"> <li>• Processes audio results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataVideo()	<ul style="list-style-type: none"> <li>• Processes video results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
Next Button	<ul style="list-style-type: none"> <li>• When the next button is clicked, a click function is evoked to increment the current page by one and call the API again for the next 100 items: searchNASALibrary()</li> </ul>
Previous Button	<ul style="list-style-type: none"> <li>• When the previous button is clicked, a click function is evoked to decrement the current page by one and call the API again for the previous 100 items: searchNASALibrary()</li> </ul>



First Button	<ul style="list-style-type: none"> <li>When the first button is clicked, a click function is evoked to set the current page to one and call the API again for the first 100 items: searchNASALibrary()</li> </ul>
Last Button	<ul style="list-style-type: none"> <li>When the last button is clicked, a click function is evoked to set the current page to the last page and call the API again for the last 100 items: searchNASALibrary()</li> </ul>
checkLibraryResultButtons()	<ul style="list-style-type: none"> <li>Enables/Disables paging buttons if necessary e.g. first button enabled then previous button disabled</li> </ul>

### Data: nasaAPI-data.js

This API call uses Queue().

<a href="https://data.nasa.gov/resource/9g7e-7hzz.json">https://data.nasa.gov/resource/9g7e-7hzz.json</a>	
show_city()	<ul style="list-style-type: none"> <li>Plucks 'center' from the JSON and renders a pie chart illustrating the number of facilities in each city</li> </ul>
show_status_pie()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a pie chart illustrating the status of the facilities: active, inactive, etc</li> </ul>
show_center()	<ul style="list-style-type: none"> <li>Plucks 'city' from the JSON and renders a bar chart illustrating the number of the facilities in each center</li> </ul>
how_status()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a bar chart illustrating the status of the facilities in each center</li> </ul>

### helperTools.js

Common functions used across all the js files

splitDate()	<ul style="list-style-type: none"> <li>This function takes two arguments: varDate (date), monthName (Boolean)</li> <li>This function splits dates into components: 2015-10-31 and returns object: {year:2015, month:10, day:31}</li> <li>monthName = 1 returns month name.</li> <li>monthName = 0 returns month number</li> </ul>
dscovrDistance()	<ul style="list-style-type: none"> <li>This function takes two sets of x y z coordinates</li> <li>Calculation found on <a href="https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php">https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php</a></li> </ul>

---

	<ul style="list-style-type: none"> <li>Calculates the 3D distance between two sets of x y z coordinates</li> </ul>
getNasaCenter()	<ul style="list-style-type: none"> <li>This function takes one arguments: centerName</li> <li>The function takes the NASA center name and returns a web address</li> </ul>
escapeHtml()	<ul style="list-style-type: none"> <li>This function escapes characters in text returned from API</li> </ul>
clearEpicResults()	<ul style="list-style-type: none"> <li>This function clears the page of EPIC results. It's called from a button</li> </ul>
clearLibraryResults()	<ul style="list-style-type: none"> <li>This function clears the page of Library results. It's called from a button</li> </ul>

- 
5. Testing section for further details.
  6. **We advise that you write down user stories and create wireframes/mock-ups before embarking on full-blown development.**

User stories and wireframes are provided in the [Scenarios](#) section and the [Wireframes](#) section.

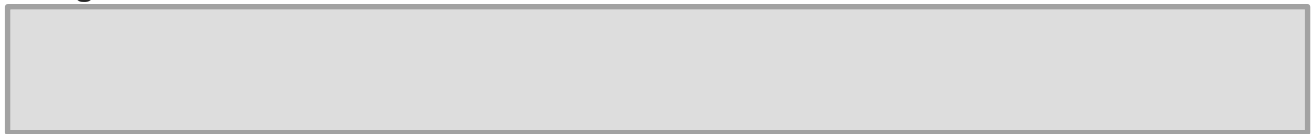
7. **The site can also make use of CSS frameworks such as Bootstrap, just make sure you maintain a clear separation between the library code and your code.**

The page is created using HTML5 and CSS3, provided by the Bootstrap Framework 4.

8. **You should conduct and document tests to ensure that all of your website's functionality works well.**

Testing was undertaken and recorded in the [HTML/CSS Structure](#)

## Navigation

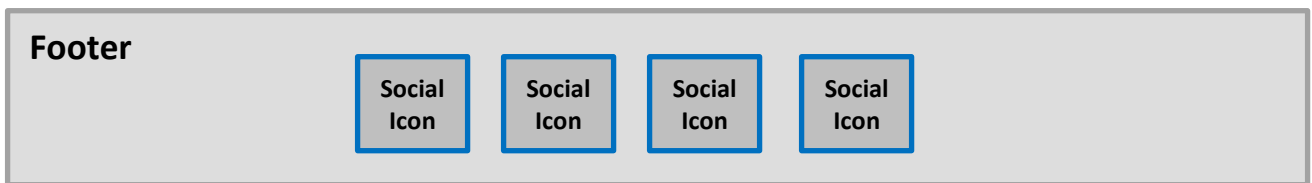


The navigation is responsive to the type of client device a user is using to view the site.

Any view below 992px wide will get a collapsing Navigation Bar. The navigation links will have a hover state making it obvious to users which navigation link they are hovering over. The text will turn the official NASA blue: #0B3D91

The collapsed navigation will have a similar hover state.

## Footer



The footer was developed/designed to be full width and contain links to NASA sites including their social media.

## Content Sections

The SPA consists of one page – index.html. The page was created using Bootstrap 4. A hero image of Atlantis is the first image users see when they visit the site. The image is owned by the web developer.

The page is divided into six distinct areas, each with its own purpose; welcome, apod, epic, library, data, contact.

## Welcome section

The welcome section welcomes the visitor to the site, it invites the visitor to use each section and briefly explains the purpose of that section.

It will be a < header > with a large background image and text in a < p >.

---

## **APOD – Astronomy Picture of the Day**

APOD, makes a call to the APOD API returning the NASA chosen image/video of the day and a brief explanation provided by a professional astronomer.

The HTML < section > has a title and explanation row filling the width of the page. The image and text are displayed in two columns. If the APOD is a video, it will take up one row while its text will take up another row below the video.

**See the APOD Wireframe**

## **EPIC**

The visitor has the option to view the most recent EPIC image received or chose a date and image type and send that date and image type to the EPIC API. The API returns images taken by the EPIC camera on that date. The list of images is paged, one image per page.

If the visitor chooses to view the most recent EPIC image then only one image is returned. It will be the last image received from DSCOVR on the previous day.

EPIC is a HTML < section >. The search part is in a div containing descriptive text, an input field for choosing a date, radio choices for natural/enhanced image and a search button.

When the user clicks the Search button, a < div > displays showing the results in two columns. The image in the left column and its text in the right column. Paging buttons also appear with the API results.

**The most recent image API call displays the result in the same way as the Date search option.**

See the EPIC Wireframe

## **Library – Search NASA Image and Video Library**

The first part of the library < section > takes a text query and media type chosen by the site visitor and sends that query and media type to the library API. The API returns results in lots of 100 items. The user can page through the results.

**After the user clicks the Search Button, a div for results appears below the search input area. It shows rows of the search results. Each result has an image in the left column while the text appears in the right column.**

Paging buttons are at the top of the results section.

See the Library Wireframe

---

## Data

The NASA Facilities dataset is made available by NASA as a json file. This section takes the data and using dc/d3/crossfilter renders pie charts and bar charts.

## Colours

#050215	rgba(5, 2, 21)	Blue	
#0B3D91	rgba(11, 61, 145)	NASA Blue	
#FC3D21	rgba(252, 61, 33)	NASA Red	

## SCSS/CSS

Bootstrap 4 provides the fundamental CSS and JavaScript for the entire site.

However, custom styles have been created and a mainStyles.css file can be found in the assets/css directory.

Custom styles for each content section, and navigation are in their own SCSS file.

mainStyles.css is created from SCSS files.

apod.scss	bootstrap-overrides.scss
data.scss	epic.scss
Global.scss	intro.scss
library.scss	mainStyles.scss
mixins.scss	navbar.scss
variables.scss	

## JavaScript

A JavaScript file has been created for each content section. Each section has its own API call.

NASA prefers that developers sign up for a developer key. This key is used in every API call.

nasaAPI-apod.js	nasaAPI-data.js
nasaAPI-library.js	nasaAPI-epic.js
helperTools.js	

## APOD: nasaAPI-apod.js

This API call uses AJAX.

<p>Returned from API</p> <ul style="list-style-type: none"> <li>– copyright</li> <li>– date</li> <li>– explanation</li> <li>– hdurl</li> <li>– media_type</li> <li>– service_version</li> <li>– title</li> <li>– url</li> </ul>	<pre>{copyright: "Jingyi Zhang", date: "2018-06-19", explanation: "They m ▼ e...umans. Almost Hyperspace: Random APOD Generator", hdurl: "https tromatolites_Zhang_1587.jpg", media_type: "image", ...} ⓘ   copyright: "Jingyi Zhang"   date: "2018-06-19"   explanation: "They may look like round rocks, but they're alive. Mc   hdurl: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites   media_type: "image"   service_version: "v1"   title: "Ancients of Sea and Sky"   url: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites_2   ► __proto__: Object</pre>
---	---

<pre>var url = "https://api.nasa.gov/planetary/apod?api_key=pyZKDq8 cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAxZ";  \$.ajax({   url: url,   success: function(result) {     ...   });</pre>	<p>The url provided by NASA takes an API key.</p> <p>An AJAX call is made using the url and the results are written out to the HTML page.</p>
--	---

## EPIC: nasaAPI-epic.js

This API call uses FETCH.

This content section will not work in Internet Explorer. FETCH is not supported by IE.

Returned from API <ul style="list-style-type: none"><li>attitude_quaternions</li><li>caption</li><li>centroid_coordinates</li><li>coords</li><li>date</li><li>dscovr_j2000_position</li><li>identifier</li><li>image</li><li>lunar_j2000_position</li><li>sun_j2000_position</li><li>version</li></ul>	<pre>▼ (12) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ   ▼ 0:     ▶ attitude_quaternions: {q0: 0.009435, q1: 0.614245, q2: -0.772655, q3: 0.160055}     ▶ caption: "This image was taken by the NASA EPIC camera onboard the NOAA DSCOVR spac     ▶ centroid_coordinates: {lat: 12.187576, lon: 156.869864}     ▶ coords: {centroid_coordinates: {...}, dscovr_j2000_position: {...}, lunar_j2000_positic     ▶ date: "2017-04-15 00:50:27"     ▶ dscovr_j2000_position: {x: 1369514.38358, y: 340216.540237, z: 304757.805862}     ▶ identifier: "20170415005515"     ▶ image: "epic_1b_20170415005515"     ▶ lunar_j2000_position: {x: -158260.421883, y: -355114.669844, z: -114884.177393}     ▶ sun_j2000_position: {x: 135998314.92302, y: 58228275.399883, z: 25241919.575161}     ▶ version: "02"     ▶ __proto__: Object   ▶ 1: {identifier: "20170415024318", caption: "This image was taken by the NASA EPIC car</pre>
--	---

<a href="https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h">https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h</a> <a href="https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ">https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ</a>	
getEpicImageByDate()	<ul style="list-style-type: none"><li>Function called from Search button on the HTML page.</li><li>Date and image type taken from visitor input form</li><li>If image type is <b>enhanced</b> then a call is made to EPIC API using <b>FETCH</b> and enhanced url. A JSON Object is returned and parsed.</li><li>If image type is <b>natural</b> then a call is made to EPIC API using <b>FETCH</b> and natural url. A JSON Object is returned and parsed.</li><li>The JSON object is sent to pageTheResult() for slicing into 1 item per page. 1 array item per page.</li><li>The sliced array is sent to getResultItems() for looping and rendering to HTML</li></ul>
getResultItems()	<ul style="list-style-type: none"><li>Data items from API result are rendered to HTML in a forEach loop</li></ul>
pageTheResult()	<ul style="list-style-type: none"><li>Paging the JSON Object to one item per page using JavaScript Array slice() Method</li></ul>
getNumberOfPages()	<ul style="list-style-type: none"><li>Retrieve the total number of items returned from API</li></ul>
nextPage()	<ul style="list-style-type: none"><li>Function called from Next Button</li></ul>
previousPage()	<ul style="list-style-type: none"><li>Function called from Previous Button</li></ul>

firstPage()	<ul style="list-style-type: none"> <li>Function called from First Button</li> </ul>
lastPage()	<ul style="list-style-type: none"> <li>Function called from Last Button</li> </ul>
check()	<ul style="list-style-type: none"> <li>Enables/Disables paging buttons if necessary e.g. first button enabled then previous button disabled</li> </ul>
getMostRecentEpic()	<ul style="list-style-type: none"> <li>This function retrieves the most recent date of natural colour imagery and displays the last image in the array.</li> </ul>

## Library: nasaAPI-library.js

This API call uses XMLHttpRequest.

Object returned from API	<pre> ▼ {collection: {...}} ⓘ   ▼ collection:     href: "https://images-api.nasa.gov/search?q=orion&amp;page=1&amp;media_type=image"     ▶ items: (100) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}]     ▶ links: [...]     ▶ metadata: {total_hits: 5137}       version: "1.0"     ▶ __proto__: Object     ▶ __proto__: Object         </pre>
The data array inside the items array	<pre> ▼ {collection: {...}} ⓘ   ▼ collection:     href: "https://images-api.nasa.gov/search?q=orion&amp;page=1&amp;media_ty     ▼ items: Array(100)       ▼ 0:         ▼ data: Array(1)           ▼ 0:             center: "KSC"             date_created: "2016-09-20T00:00:00Z"             description: "Tim Goddard, NASA Open Water Recovery Opera             ▶ keywords: (8) ["Orion", "GSDO", "recovery", "JSC", "NBL",               location: "Orion NBL"               media_type: "image"               nasa_id: "JSC-20160920-PH_JNB01_0002"               photographer: "NASA/James Blair"               title: "Orion Neutral Buoyancy Lab (NBL) Activities"             ▶ __proto__: Object             length: 1             ▶ __proto__: Array(0)             href: "https://images-assets.nasa.gov/image/JSC-20160920-PH_J             ▶ links: [...]             ▶ __proto__: Object         </pre>



API documentation provided by NASA: <a href="https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf">https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf</a> <a href="https://images-api.nasa.gov/search?q={q}">https://images-api.nasa.gov/search?q={q}</a>	
getQueryText()	<ul style="list-style-type: none"> <li>• Function called from Search button on the HTML page.</li> <li>• Query text and media type taken from visitor input form</li> <li>• The url is build using query text and media type</li> <li>• The url is sent to searchNASALibrary()</li> </ul>
searchNASALibrary()	<ul style="list-style-type: none"> <li>• Call made to NASA Library API using XMLHttpRequest()</li> <li>• If result is image then the JSON object and media type is sent to getLibraryResultsDataImage()</li> <li>• If result is audio then the JSON object and media type is sent to getLibraryResultsDataAudio()</li> <li>• If result is video then the JSON object and media type is sent to getLibraryResultsDataVideo()</li> <li>• Each media type required different html rendering hence the 3 different functions</li> </ul>
getLibraryResultsDataImage()	<ul style="list-style-type: none"> <li>• Processes image results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataAudio()	<ul style="list-style-type: none"> <li>• Processes audio results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataVideo()	<ul style="list-style-type: none"> <li>• Processes video results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
Next Button	<ul style="list-style-type: none"> <li>• When the next button is clicked, a click function is evoked to increment the current page by one and call the API again for the next 100 items: searchNASALibrary()</li> </ul>
Previous Button	<ul style="list-style-type: none"> <li>• When the previous button is clicked, a click function is evoked to decrement the current page by one and call the API again for the previous 100 items: searchNASALibrary()</li> </ul>

First Button	<ul style="list-style-type: none"> <li>When the first button is clicked, a click function is evoked to set the current page to one and call the API again for the first 100 items: searchNASALibrary()</li> </ul>
Last Button	<ul style="list-style-type: none"> <li>When the last button is clicked, a click function is evoked to set the current page to the last page and call the API again for the last 100 items: searchNASALibrary()</li> </ul>
checkLibraryResultButtons()	<ul style="list-style-type: none"> <li>Enables/Disables paging buttons if necessary e.g. first button enabled then previous button disabled</li> </ul>

### Data: nasaAPI-data.js

This API call uses Queue().

<a href="https://data.nasa.gov/resource/9g7e-7hzz.json">https://data.nasa.gov/resource/9g7e-7hzz.json</a>	
show_city()	<ul style="list-style-type: none"> <li>Plucks 'center' from the JSON and renders a pie chart illustrating the number of facilities in each city</li> </ul>
show_status_pie()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a pie chart illustrating the status of the facilities: active, inactive, etc</li> </ul>
show_center()	<ul style="list-style-type: none"> <li>Plucks 'city' from the JSON and renders a bar chart illustrating the number of the facilities in each center</li> </ul>
how_status()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a bar chart illustrating the status of the facilities in each center</li> </ul>

### helperTools.js

Common functions used across all the js files

splitDate()	<ul style="list-style-type: none"> <li>This function takes two arguments: varDate (date), monthName (Boolean)</li> <li>This function splits dates into components: 2015-10-31 and returns object: {year:2015, month:10, day:31}</li> <li>monthName = 1 returns month name.</li> <li>monthName = 0 returns month number</li> </ul>
dscovrDistance()	<ul style="list-style-type: none"> <li>This function takes two sets of x y z coordinates</li> <li>Calculation found on <a href="https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php">https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php</a></li> </ul>

---

	<ul style="list-style-type: none"> <li>Calculates the 3D distance between two sets of x y z coordinates</li> </ul>
getNasaCenter()	<ul style="list-style-type: none"> <li>This function takes one arguments: centerName</li> <li>The function takes the NASA center name and returns a web address</li> </ul>
escapeHtml()	<ul style="list-style-type: none"> <li>This function escapes characters in text returned from API</li> </ul>
clearEpicResults()	<ul style="list-style-type: none"> <li>This function clears the page of EPIC results. It's called from a button</li> </ul>
clearLibraryResults()	<ul style="list-style-type: none"> <li>This function clears the page of Library results. It's called from a button</li> </ul>

---

9. Testing section below

10. Write a README.md file for your project that

- a. explains what the project does and
- b. the need that it fulfills.
- c. It should also describe the functionality of the project,
- d. as well as the technologies used.
- e. Detail how the project was deployed
- f. and tested and
- g. if some of the work was based on other code, explain what was kept and how it was changed to fit your need.

**A project submitted without a README.md file will FAIL.**

I hope this ReadMe.md file meets all the criteria listed above.

11. Use Git & GitHub for version control. Each new piece of functionality should be in a separate commit.

Git was used for version control and my logs clearly show that a commit was made after each new piece of functionality was added and/or an issue was resolved. See Version Control section below.

12. Deploy the final version of your code to a hosting platform such as GitHub Pages.

GitHub Pages was used to host the final SPA. See Deployment section below.

---

# Strategy Plane

The overall aim of the project was to create a single page application website (SPA) that utilises the NASA API. A selection of API calls was made to illustrate the type of data NASA makes available to the public. The scope is fundamentally determined by the strategy of the site, so it was important we define a good strategy from the outset. This strategy outlines the benefit to the users.

We aimed to:

- determine the value of the website and
- the reason for the websites existence

## Define roles and responsibilities

From here on the **single page application website** will be known as the **SPA**.

For the purposes of this project, Sonya Cooley had full authority, primary responsibility, and full accountability for all aspects of the project.

She had a Mentor available to her throughout the development of the SPA.

## Project Charter

	Objectives
<b>Purpose:</b> What purpose does the website serve?	The SPA will target users interested in NASA, the type of work they do and the topic of Space in general. Casual users may be interested in the images made available by NASA.
<b>Goals:</b> What outcomes does it need to achieve?	<ul style="list-style-type: none"><li>• Successfully access the NASA API</li><li>• Publish the data NASA has made available in an interesting and user-friendly way</li></ul>
<b>Target audience:</b> Whom must the product appeal to and work for?	NASA enthusiasts and casual site visitors looking for interesting and easy to understand data.
<b>Success indicators:</b> How will you know you have achieved project goals?	<ul style="list-style-type: none"><li>• Data is successfully returned from the API</li><li>• Visitors regularly revisit to view new data and new visitor numbers increase</li></ul>
<b>Strategies:</b> What approaches will help to realize the goals?	<ul style="list-style-type: none"><li>• We will take a mobile first approach to Content.</li><li>• Focus content at our target audience</li><li>• Aim to keep the site simple and not over complicated</li></ul>

	<ul style="list-style-type: none"> <li>• Present NASA data in an interesting and user-friendly way</li> </ul>
<b>Tactics:</b> <b>What activities might help to realise the strategies?</b>	<ul style="list-style-type: none"> <li>• Provide a user experience that is accessible and enjoyable for all visitors             <ul style="list-style-type: none"> <li>○ following conventions for design and interaction</li> <li>○ providing clear and consistent navigation</li> <li>○ testing usability with a wide range of clients and industry standard tools</li> </ul> </li> </ul>

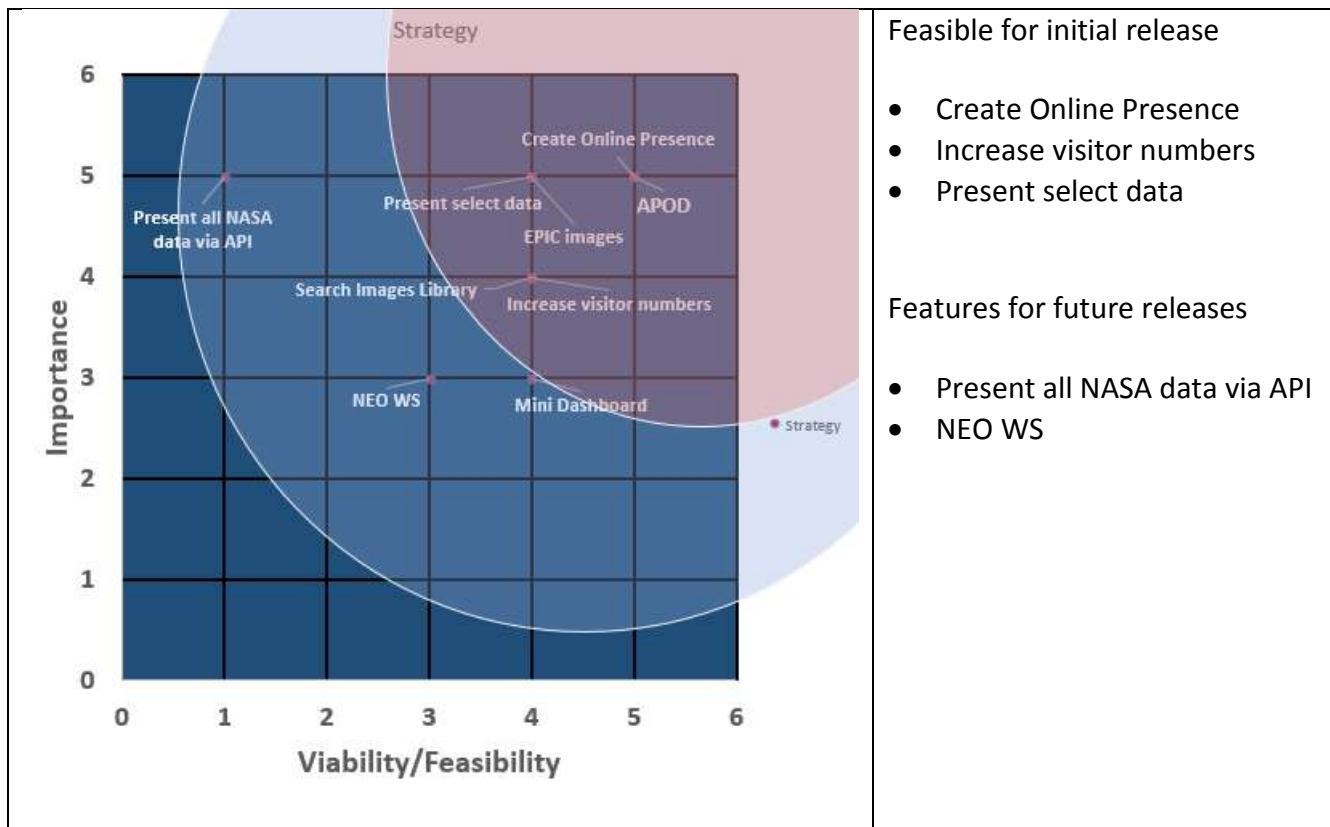
## Website development Roadmap

The UXD will be driven by API data and user needs.

Define	NASA API analysis, Requirements gathering, SEO, Social Media, Content Strategy – Personas, Content inventory returned from API.
Design	Information architecture, Functional & technical requirements, Navigation design, Wireframes, UX/UI, Pages, Branding, style guides, mock-ups.
Develop	Look & feel, Design and Development, Build, Version control, Testing, Deploy

## Strategy Trade-off

Opportunity/Problem	Importance	Viability/Feasibility
Create Online Presence	5	5
Increase visitor numbers	4	4
Present all NASA data via API	5	1
Present select data	5	4
APOD	5	5
EPIC images	5	4
Search Images Library	4	4
NEO WS	3	3
Mini Dashboard	3	4



## Defensive Design

Defensive design for the Web usually focuses on the most common points of failure: forms, search, the address bar and server problems

- It employs validation to check for mistakes before they frustrate the site user,
- It expands available options based on the user's implied intent,
- It protects site visitors from server errors and broken links with informative messages and
- It assists the user before mistakes happen.

For this SPA all API calls in the JS files will have a message to users if the connect fails for any reason. Users will be told there was a network/connection error and to try again later.

If users fail to enter text or a date for inputs before clicking a search button, they will be informed that text/date are required.

If users query text fails to return any search results, users will be informed that there were no results for that search query.

---

# Scope Plane

Our project scope was based on our defined Strategy. We;

- determined the SPA requirements
- determined the SPA key functionality and
- determined what features were to be included in this and possible future product releases

The SPA targeted NASA enthusiasts and casual visitors with NASA data. The site accesses API data and publishes to the SPA. A combination of images, tables, and HTML5 was used to make the data useful and interesting. Semantic HTML was used throughout and the site is responsive to a broad range of devices.

## Note about FETCH

### Coding

The JavaScript was optimised for Chrome. A combination of AJAX, FETCH and XMLHttpRequest were used to make the API calls. These were used to show flexibility.

The downside of this means that the FETCH API call does not work in Internet Explorer.

### Fetch

[https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)

Desktop:

Feature	Chrome	Edge	Firefox(Gecko)	IE	Opera	Safari(WebKit)
Basic support	42	14	39 (39) 34[1] 52 (52)[2]	No support	29 28[1]	10.1
ReadableStream response body	43	14	No support [3]	No support	?	No support

Mobile:

Feature	Android Webview	Chrome for Android	Edge	Firefox Mobile (Gecko)	IE Phone	Opera Mobile	Safari Mobile
Basic support	42	42	(Yes)	(Yes)	No support	No support	10.1
ReadableStream response body	43	43	(Yes)	No support [3]	No support	?	No support

## Scenarios

### NASA enthusiast:

A NASA enthusiast will be a visitor who is aware of NASA. They will be aware of the NASA organisation, the type of work they do and the type of data that is made available.



---

An enthusiast will visit the site with an expectation of being able to view NASA data and search the data for topics/images of interest to them. The enthusiast will most likely use the Search NASA images library.

**Casual visitor:**

A casual visitor will visit the site possibly out of curiosity to see what data is available. They will revisit if the data is presented in a favourable way. Images are a great way to introduce visitors to NASA data without overwhelming them with complex data.

**Student:**

A student may visit the site to access NASA images and videos. They may also have a specific interest in the EPIC project.

## Functional Specifications

The site will provide visitors with access to the NASA data made available via its API.

The site will provide the ability to;

- navigate content
- view NASA API data
  - Astronomy Picture of the Day (APOD)
  - Earth Polychromatic Imaging Camera (EPIC) images – the visitor will input a date and select an image type. This API returns up to 12 images and information on each image. EPIC takes an image of earth every 2 hours approx.
  - Search the NASA Images and Video Library – the visitor will input a text query and select whether to search for images, audio or video. The API returns the results in pages of 100 items.
  - Data – a select NASA dataset will be used to chart and illustrate the type of data available from NASA.
- Welcome and Footer area with further information

The SPA will be optimised for latest version of Chrome, Firefox, Internet Explorer, Safari and Opera and optimised for mobile usage. HTML and CSS will be written using the Mobile-First approach. The mobile-first approach is designing for the smallest screen and working your way up to desktop. Please note that the EPIC data will not be available in Internet Explorer due to the use of FETCH.

---

# Content Requirements

The SPA will follow a standard format, with a structure based on a Bootstrap 4 theme.

The page will have a sticky navbar always available to visitors and an intro section briefly explaining each section.

The first content section will be a welcome to visitors and a brief explanation about each content section.

The next content section will publish the Astronomy Picture of the Day (APOD).

The third content section provides images captured by the EPIC project.

The fourth content section will allow users search the NASA Images and Video library.

The footer will contain links to various NASA sites providing further information on the content sections.

All pages will be created using HTML5 and CSS3, provided by the Bootstrap Framework 4.

## Page structure

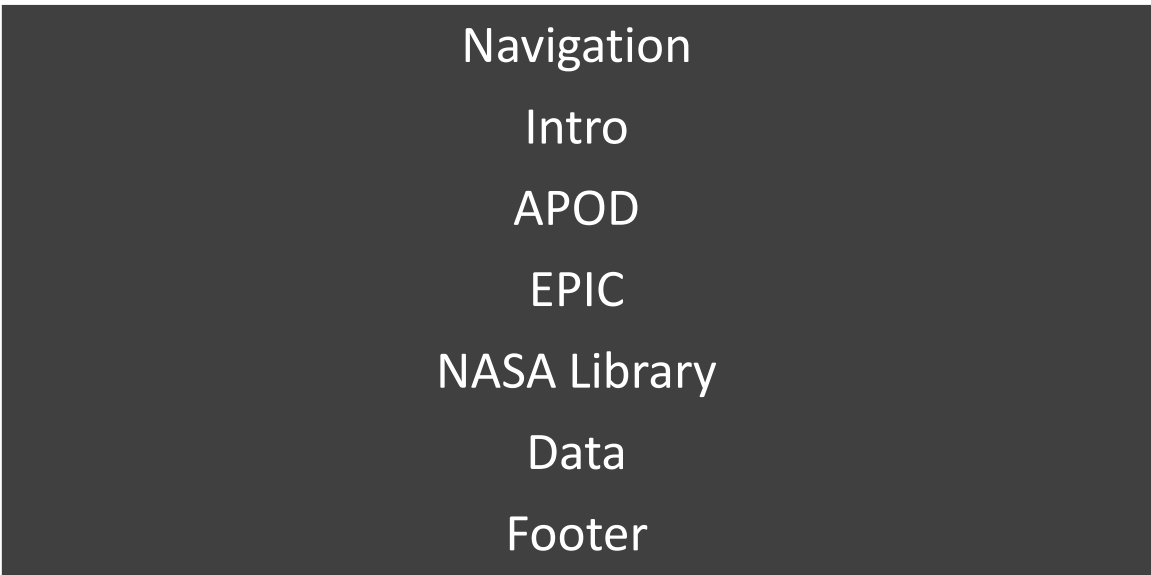


Figure 1 – Page structure

SPA	<p>The SPA will consist of a hero image in the intro section, a navigation bar and a footer with links to other information.</p> <p>The navigation will allow users to navigate up and down the page.</p> <p>The content sections will provide APOD, EPIC images and a search facility.</p>
-----	---

---

## Navigation systems

'Sticky' navbar.

The navbar will be always be available to users at the top of the page regardless of which content section a visitor is viewing.

Description	Response
NASA Images Portal	Takes the user to the welcome/intro section
APOD	Takes the user to the Astronomy Picture of the Day section
EPIC	Takes the user to the DSCOVR's Earth Polychromatic Imaging Camera (EPIC) section
Library	Takes the user to the search facility that allows visitors to search the NASA image and video library
Data	The Data section shows a mini dashboard of NASA data
Contact/Information	Shows where to get further information about NASA, social media, etc.

---

# Structure & Skeleton Plane

Our goal for the Structure plane is to organise the information architecture and interactions for the site. We will keep a consistent, predictable, learnable interface, and, interactions. We will use industry standard technologies to implement expected behaviours when using the site, e.g. tooltips, navigation, including accessibility, etc.

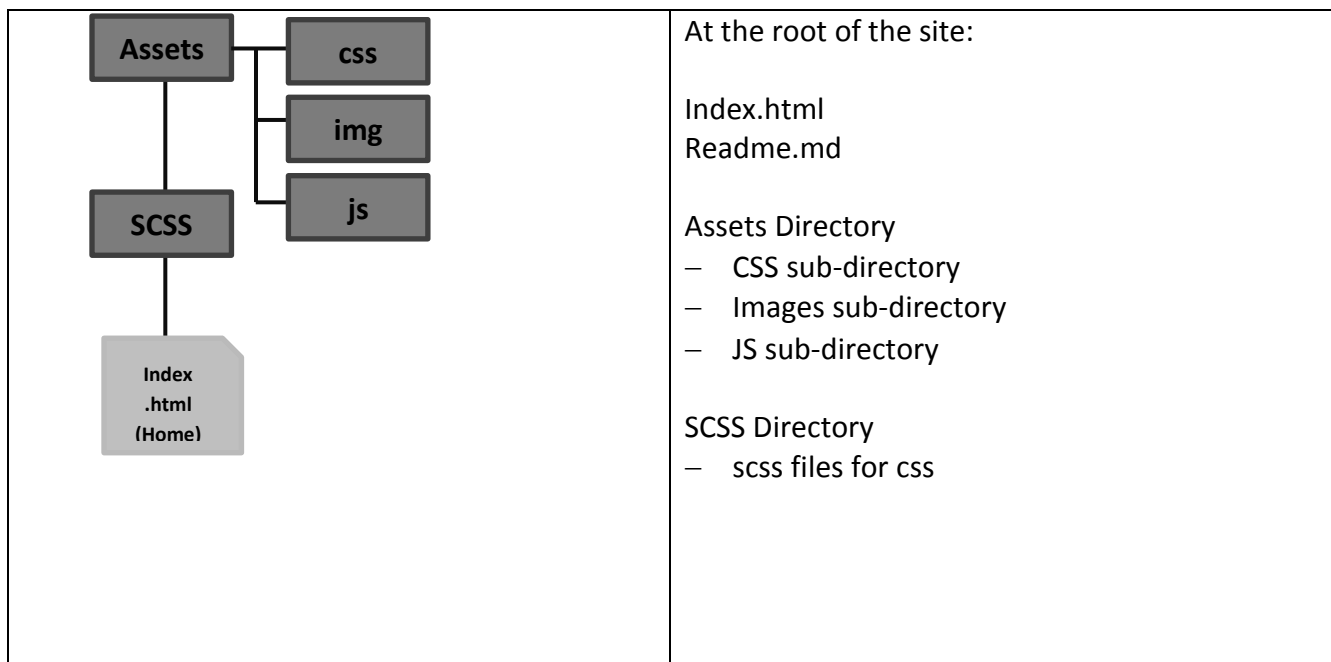
Users will find the navigation at the top of the site in an expected location. The site will be intuitive and follow tried and trusted paths.

The input fields will follow user expectations where feedback is provided if user interactions are unexpected, incomplete or complete.

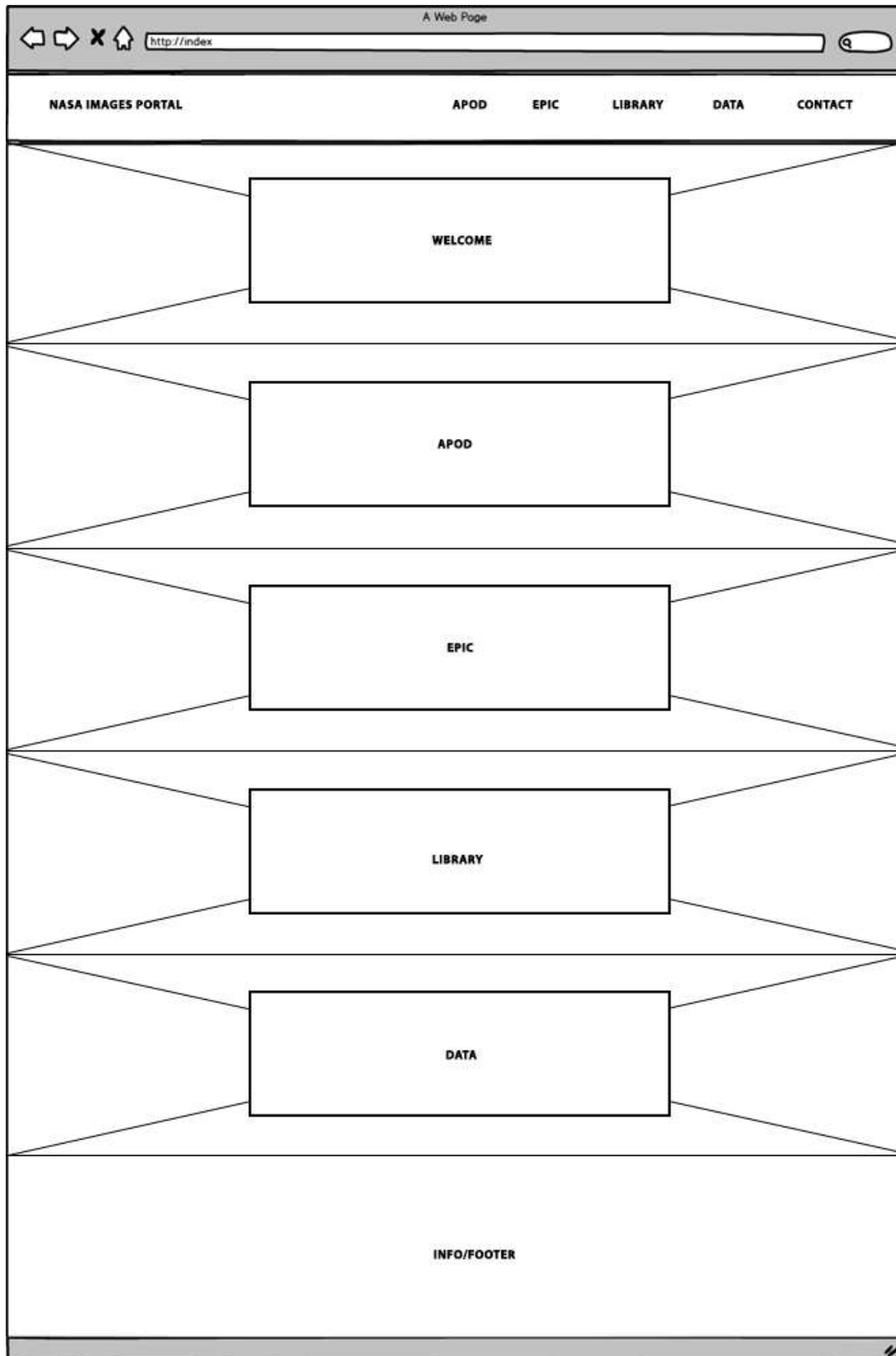
## Information architecture

The SPA will implement content in a single page. Each content section will be clearly labelled and be visually different from the previous content section. Making it obvious to visitors that they are in a different content section.

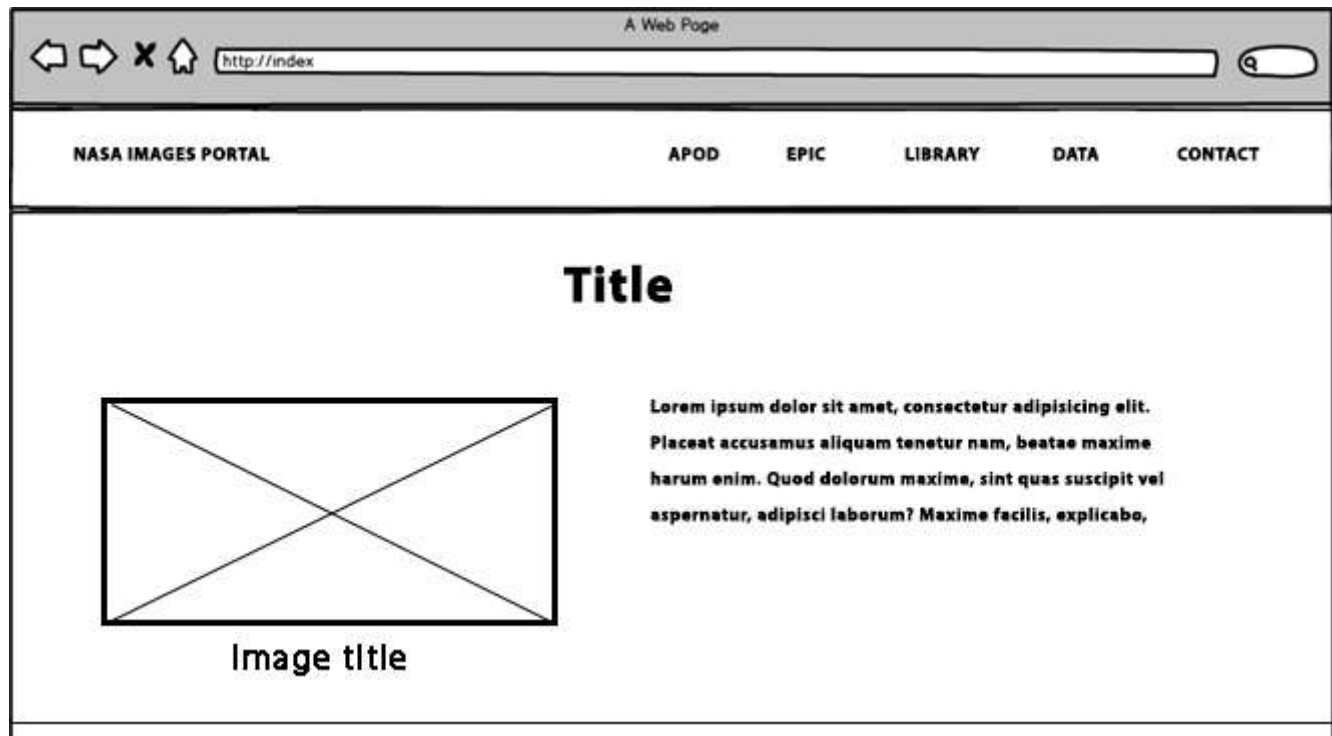
The site directories and files will be organised in the following way;



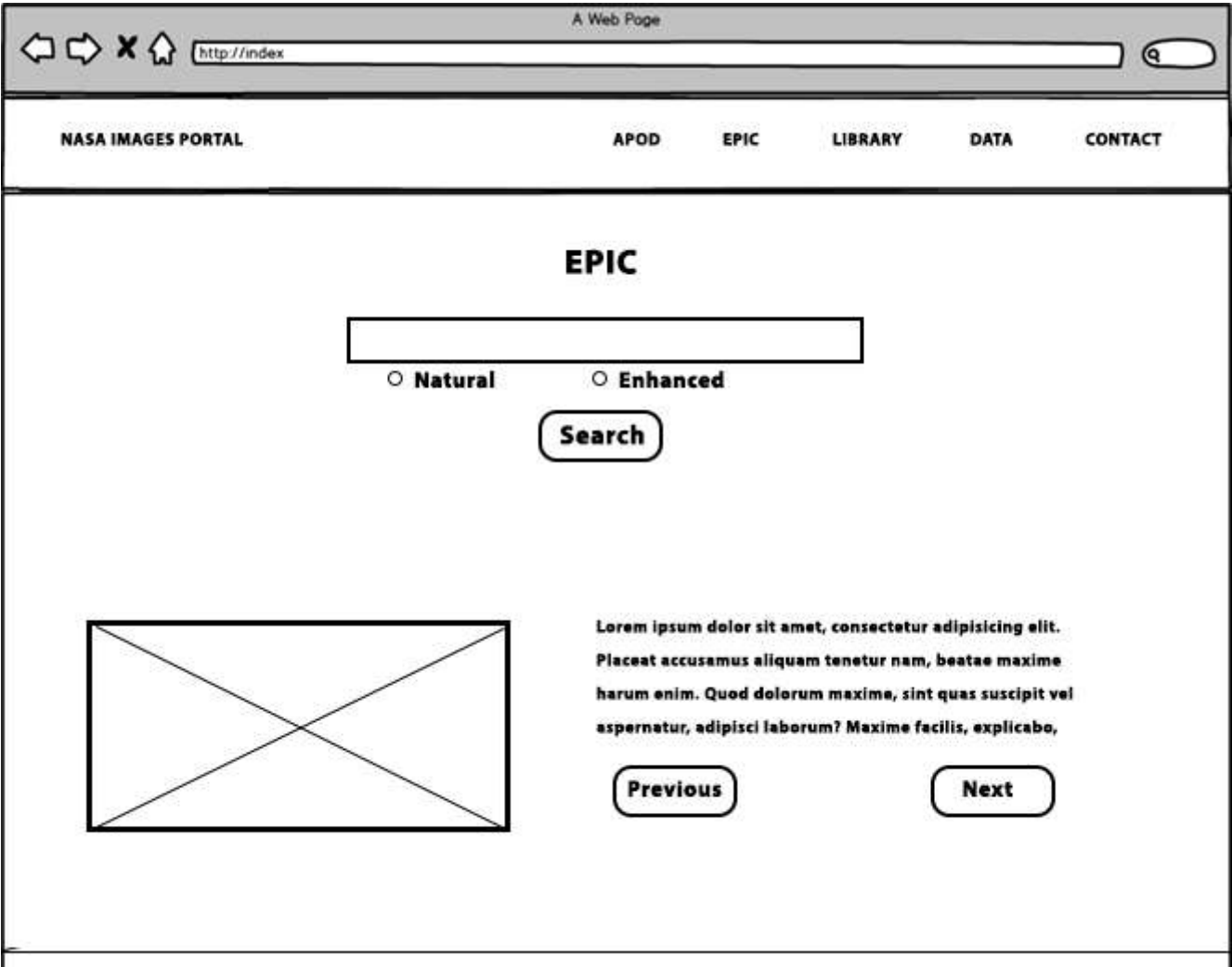
# Wireframes



# APOD








# EPIC



# Library

A Web Page



NASA IMAGES PORTAL

APOD   EPIC   **LIBRARY**   DATA   CONTACT

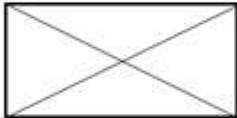
## LIBRARY

☐ Image   ☒ **Video**   ☐ Audio

Search

Previous

Next



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Placeat accusamus aliquam tenetur nam, beatae maxime harum enim. Quod dolorum maxime, sint quas suscipit vel aspernatur, adipisci laborum? Maxime facilis, explicabo,



---

# Surface Plane

## Development Phase

### IDE

The website was developed using the Cloud9 IDE.

Cloud9 IDE is an online integrated development environment, that supports hundreds of programming languages. It enables developers to get started with coding immediately with pre-configured workspaces, collaborate with their peers with collaborative coding features, and web development features like live preview and browser compatibility testing.

### Version Control

Git was used to manage the source code for this project. Git is a version control system for tracking changes in project files.

Project files were committed to Git after each major functional addition, update or implementation of testing results.

Following the initial commit to Git, each major update was followed by a Git add and commit.

**A full Git log can be provided on request or be seen on the GitHub project repository.**

### Readme

A Readme markdown file is provided with the site on GitHub. It explains what the project does and the need that it fulfils.

It also describes the functionality of the site, as well as the technologies used.

The Readme provides information on how the site was deployed and tested and if some of the work was based on other code.

## HTML/CSS Structure

### Navigation

Home	APOD	EPIC	Library	Data	Contact
------	------	------	---------	------	---------

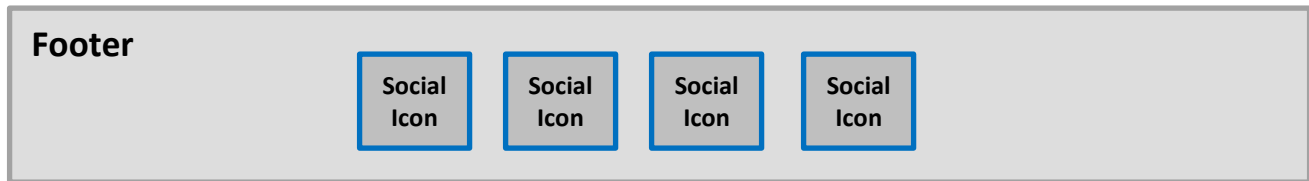
The navigation is responsive to the type of client device a user is using to view the site.

Any view below 992px wide will get a collapsing Navigation Bar. The navigation links will have a hover state making it obvious to users which navigation link they are hovering over. The text will turn the official NASA blue: #0B3D91

The collapsed navigation will have a similar hover state.

---

## Footer



The footer was developed/designed to be full width and contain links to NASA sites including their social media.

## Content Sections

The SPA consists of one page – index.html. The page was created using Bootstrap 4. A hero image of Atlantis is the first image users see when they visit the site. The image is owned by the web developer.

The page is divided into six distinct areas, each with its own purpose; welcome, apod, epic, library, data, contact.

### Welcome section

The welcome section welcomes the visitor to the site, it invites the visitor to use each section and briefly explains the purpose of that section.

It will be a < header > with a large background image and text in a < p >.

### APOD – Astronomy Picture of the Day

APOD, makes a call to the APOD API returning the NASA chosen image/video of the day and a brief explanation provided by a professional astronomer.

The HTML < section > has a title and explanation row filling the width of the page. The image and text are displayed in two columns. If the APOD is a video, it will take up one row while its text will take up another row below the video.

See the APOD Wireframe

### EPIC

The visitor has the option to view the most recent EPIC image received or chose a date and image type and send that date and image type to the EPIC API. The API returns images taken by the EPIC camera on that date. The list of images is paged, one image per page.

If the visitor chooses to view the most recent EPIC image then only one image is returned. It will be the last image received from DSCOVR on the previous day.

EPIC is a HTML < section >. The search part is in a div containing descriptive text, an input field for choosing a date, radio choices for natural/enhanced image and a search button.



When the user clicks the Search button, a < div > displays showing the results in two columns. The image in the left column and its text in the right column. Paging buttons also appear with the API results.

The most recent image API call displays the result in the same way as the Date search option.  
See the EPIC Wireframe

**Library – Search NASA Image and Video Library**

The first part of the library < section > takes a text query and media type chosen by the site visitor and sends that query and media type to the library API. The API returns results in lots of 100 items. The user can page through the results.

After the user clicks the Search Button, a div for results appears below the search input area. It shows rows of the search results. Each result has an image in the left column while the text appears in the right column.

Paging buttons are at the top of the results section.

See the Library Wireframe

**Data**

The NASA Facilities dataset is made available by NASA as a json file. This section takes the data and using dc/d3/crossfilter renders pie charts and bar charts.

**Colours**

#050215	rgba(5, 2, 21)	Blue	
#0B3D91	rgba(11, 61, 145)	NASA Blue	
#FC3D21	rgba(252, 61, 33)	NASA Red	

**SCSS/CSS**

Bootstrap 4 provides the fundamental CSS and JavaScript for the entire site.  
However, custom styles have been created and a mainStyles.css file can be found in the assets/css directory.  
Custom styles for each content section, and navigation are in their own SCSS file.  
mainStyles.css is created from SCSS files.

apod.scss	bootstrap-overrides.scss
data.scss	epic.scss

Global.scss	intro.scss
library.scss	mainStyles.scss
mixins.scss	navbar.scss
variables.scss	

## JavaScript

A JavaScript file has been created for each content section. Each section has its own API call.

NASA prefers that developers sign up for a developer key. This key is used in every API call.

nasaAPI-apod.js	nasaAPI-data.js
nasaAPI-library.js	nasaAPI-epic.js
helperTools.js	

## APOD: nasaAPI-apod.js

This API call uses AJAX.

Returned from API <ul style="list-style-type: none"> <li>– copyright</li> <li>– date</li> <li>– explanation</li> <li>– hdurl</li> <li>– media_type</li> <li>– service_version</li> <li>– title</li> <li>– url</li> </ul>	<pre>{copyright: "Jingyi Zhang", date: "2018-06-19", explanation: "They m ▼ e...umans. Almost Hyperspace: Random APOD Generator", hdurl: "https tromatolites_Zhang_1587.jpg", media_type: "image", ...} ⓘ   copyright: "Jingyi Zhang"   date: "2018-06-19"   explanation: "They may look like round rocks, but they're alive. Mc   hdurl: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites   media_type: "image"   service_version: "v1"   title: "Ancients of Sea and Sky"   url: "https://apod.nasa.gov/apod/image/1806/MilkyWayStromatolites_2   ▶ __proto__: Object</pre>
--	---

<pre>var url = "https://api.nasa.gov/planetary/apod?api_key=pyZKDq8 cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAxZ";  \$.ajax({   url: url,   success: function(result) {     ...   });</pre>	<p>The url provided by NASA takes an API key.</p> <p>An AJAX call is made using the url and the results are written out to the HTML page.</p>
--	---

## EPIC: nasaAPI-epic.js

This API call uses FETCH.

This content section will not work in Internet Explorer. FETCH is not supported by IE.

Returned from API <ul style="list-style-type: none"><li>attitude_quaternions</li><li>caption</li><li>centroid_coordinates</li><li>coords</li><li>date</li><li>dscovr_j2000_position</li><li>identifier</li><li>image</li><li>lunar_j2000_position</li><li>sun_j2000_position</li><li>version</li></ul>	<pre>▼ (12) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ   ▼ 0:     ▶ attitude_quaternions: {q0: 0.009435, q1: 0.614245, q2: -0.772655, q3: 0.160055}     ▶ caption: "This image was taken by the NASA EPIC camera onboard the NOAA DSCOVR spac     ▶ centroid_coordinates: {lat: 12.187576, lon: 156.869864}     ▶ coords: {centroid_coordinates: {...}, dscovr_j2000_position: {...}, lunar_j2000_positic     ▶ date: "2017-04-15 00:50:27"     ▶ dscovr_j2000_position: {x: 1369514.38358, y: 340216.540237, z: 304757.805862}     ▶ identifier: "20170415005515"     ▶ image: "epic_1b_20170415005515"     ▶ lunar_j2000_position: {x: -158260.421883, y: -355114.669844, z: -114884.177393}     ▶ sun_j2000_position: {x: 135998314.92302, y: 58228275.399883, z: 25241919.575161}     ▶ version: "02"     ▶ __proto__: Object   ▶ 1: {identifier: "20170415024318", caption: "This image was taken by the NASA EPIC car</pre>
--	---

<a href="https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h">https://api.nasa.gov/EPIC/api/enhanced/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ h</a> <a href="https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ">https://api.nasa.gov/EPIC/api/natural/date/Date?api_key=pyZKDq8cb4x1dJi0dsodTT9PBoWkQaa5CgxmPAXZ</a>	
getEpicImageByDate()	<ul style="list-style-type: none"><li>Function called from Search button on the HTML page.</li><li>Date and image type taken from visitor input form</li><li>If image type is <b>enhanced</b> then a call is made to EPIC API using <b>FETCH</b> and enhanced url. A JSON Object is returned and parsed.</li><li>If image type is <b>natural</b> then a call is made to EPIC API using <b>FETCH</b> and natural url. A JSON Object is returned and parsed.</li><li>The JSON object is sent to pageTheResult() for slicing into 1 item per page. 1 array item per page.</li><li>The sliced array is sent to getResultItems() for looping and rendering to HTML</li></ul>
getResultItems()	<ul style="list-style-type: none"><li>Data items from API result are rendered to HTML in a forEach loop</li></ul>
pageTheResult()	<ul style="list-style-type: none"><li>Paging the JSON Object to one item per page using JavaScript Array slice() Method</li></ul>
getNumberOfPages()	<ul style="list-style-type: none"><li>Retrieve the total number of items returned from API</li></ul>
nextPage()	<ul style="list-style-type: none"><li>Function called from Next Button</li></ul>
previousPage()	<ul style="list-style-type: none"><li>Function called from Previous Button</li></ul>



API documentation provided by NASA: <a href="https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf">https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf</a> <a href="https://images-api.nasa.gov/search?q={q}">https://images-api.nasa.gov/search?q={q}</a>	
getQueryText()	<ul style="list-style-type: none"> <li>• Function called from Search button on the HTML page.</li> <li>• Query text and media type taken from visitor input form</li> <li>• The url is build using query text and media type</li> <li>• The url is sent to searchNASALibrary()</li> </ul>
searchNASALibrary()	<ul style="list-style-type: none"> <li>• Call made to NASA Library API using XMLHttpRequest()</li> <li>• If result is image then the JSON object and media type is sent to getLibraryResultsDataImage()</li> <li>• If result is audio then the JSON object and media type is sent to getLibraryResultsDataAudio()</li> <li>• If result is video then the JSON object and media type is sent to getLibraryResultsDataVideo()</li> <li>• Each media type required different html rendering hence the 3 different functions</li> </ul>
getLibraryResultsDataImage()	<ul style="list-style-type: none"> <li>• Processes image results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataAudio()	<ul style="list-style-type: none"> <li>• Processes audio results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
getLibraryResultsDataVideo()	<ul style="list-style-type: none"> <li>• Processes video results and renders HTML</li> <li>• A popover is used to view the full description provided in the Results</li> <li>• A Paging url returned in the results is used to make the next page call to the API for the next 100 items</li> </ul>
Next Button	<ul style="list-style-type: none"> <li>• When the next button is clicked, a click function is evoked to increment the current page by one and call the API again for the next 100 items: searchNASALibrary()</li> </ul>
Previous Button	<ul style="list-style-type: none"> <li>• When the previous button is clicked, a click function is evoked to decrement the current page by one and call the API again for the previous 100 items: searchNASALibrary()</li> </ul>

First Button	<ul style="list-style-type: none"> <li>When the first button is clicked, a click function is evoked to set the current page to one and call the API again for the first 100 items: searchNASALibrary()</li> </ul>
Last Button	<ul style="list-style-type: none"> <li>When the last button is clicked, a click function is evoked to set the current page to the last page and call the API again for the last 100 items: searchNASALibrary()</li> </ul>
checkLibraryResultButtons()	<ul style="list-style-type: none"> <li>Enables/Disables paging buttons if necessary e.g. first button enabled then previous button disabled</li> </ul>

### Data: nasaAPI-data.js

This API call uses Queue().

<a href="https://data.nasa.gov/resource/9g7e-7hzz.json">https://data.nasa.gov/resource/9g7e-7hzz.json</a>	
show_city()	<ul style="list-style-type: none"> <li>Plucks 'center' from the JSON and renders a pie chart illustrating the number of facilities in each city</li> </ul>
show_status_pie()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a pie chart illustrating the status of the facilities: active, inactive, etc</li> </ul>
show_center()	<ul style="list-style-type: none"> <li>Plucks 'city' from the JSON and renders a bar chart illustrating the number of the facilities in each center</li> </ul>
how_status()	<ul style="list-style-type: none"> <li>Plucks 'status' from the JSON and renders a bar chart illustrating the status of the facilities in each center</li> </ul>

### helperTools.js

Common functions used across all the js files

splitDate()	<ul style="list-style-type: none"> <li>This function takes two arguments: varDate (date), monthName (Boolean)</li> <li>This function splits dates into components: 2015-10-31 and returns object: {year:2015, month:10, day:31}</li> <li>monthName = 1 returns month name.</li> <li>monthName = 0 returns month number</li> </ul>
dscovrDistance()	<ul style="list-style-type: none"> <li>This function takes two sets of x y z coordinates</li> <li>Calculation found on <a href="https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php">https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php</a></li> </ul>



---

	<ul style="list-style-type: none"> <li>Calculates the 3D distance between two sets of x y z coordinates</li> </ul>
getNasaCenter()	<ul style="list-style-type: none"> <li>This function takes one arguments: centerName</li> <li>The function takes the NASA center name and returns a web address</li> </ul>
escapeHtml()	<ul style="list-style-type: none"> <li>This function escapes characters in text returned from API</li> </ul>
clearEpicResults()	<ul style="list-style-type: none"> <li>This function clears the page of EPIC results. It's called from a button</li> </ul>
clearLibraryResults()	<ul style="list-style-type: none"> <li>This function clears the page of Library results. It's called from a button</li> </ul>

## Testing

The website was tested on an ongoing basis. Chrome and Chrome Developer Tools were the primary browser and tool used for testing. However, the site was also tested using Firefox and Internet Explorer.

- HTML passed validation using the **Markup Validation Service** provided by The World Wide Web Consortium (W3C): <https://validator.w3.org/>
- CSS passed validation using the **CSS Validation Service** provided by The World Wide Web Consortium (W3C): <https://jigsaw.w3.org/css-validator/>

## During development:

- Console.log was used extensively for viewing returned data and testing.
- Div's had vibrant background colours so that the developer was easily able to identify them
- Each change was viewed in a chrome browser and tested using developer tools at full width resolution and using a variety of device emulators; Galaxy SIII, Galaxy 5, Laptop touch screen, iPhone 5/SE, iPhone 6/7/8, iPhone 6/7/8 Plus, iPhone X, iPad.
- Remote debugging using Android, Windows OS and Chrome Dev Tools was used to test each new functionality and new/updated page.
- The SPA was audited using Chrome Dev Tools Audit functionality

## Development/Defensive Design Testing

Testing was carried out continuously while developing the SPA. Each input was tested for empty values and checked for zero results.

As per the Defensive Design Strategy described in the Strategy Plan, all form inputs are checked for empty values. Users are messaged if they click a search button without providing text or a date.

Users are also informed by an on-screen text if their search returns no results.

Users are left in no doubt what is required of them, and if their search was unsuccessful.

### Library:

If the user clicked the search button without entering text to be searched, a message displays informing the visitor that they must enter text to search.

## Search NASA Image and Video Library

Enter a topic to search and choose the media type.

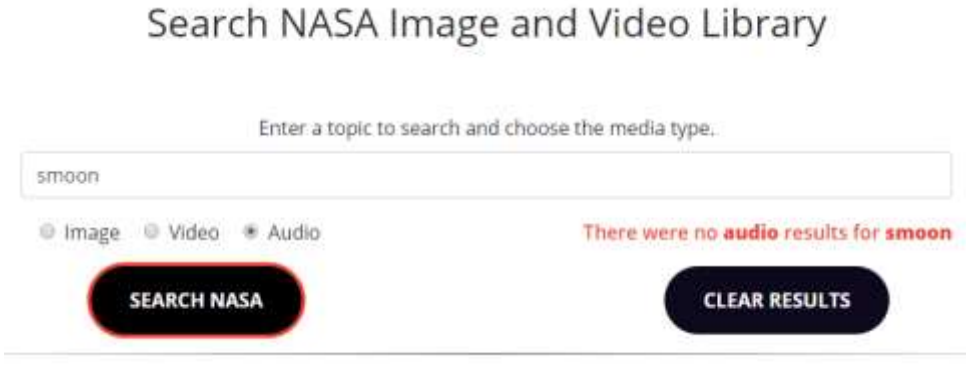
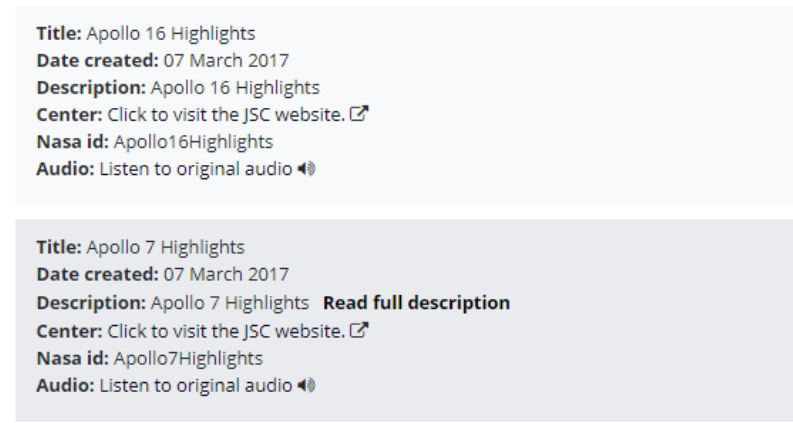
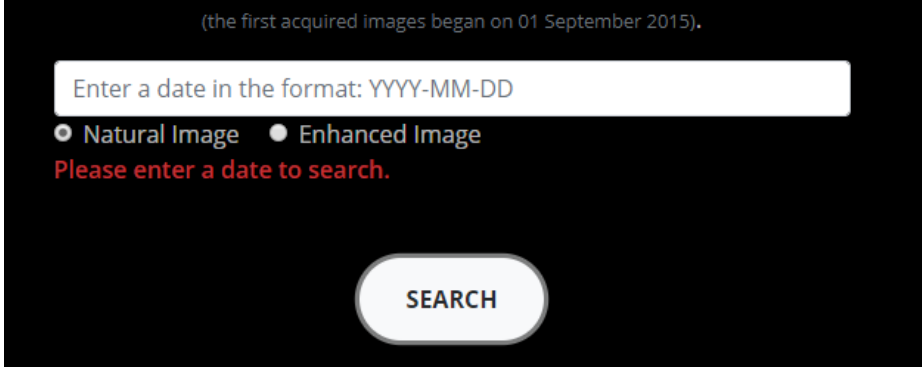
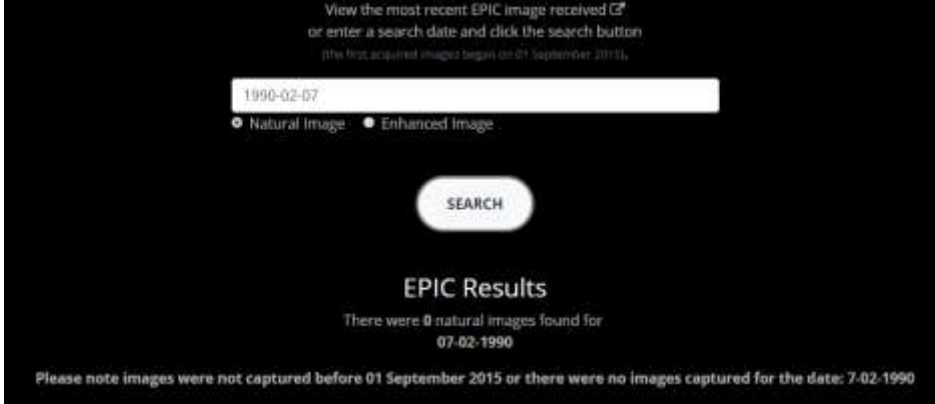
e.g. apollo, moon, europe...

☒ Image ☐ Video ☐ Audio

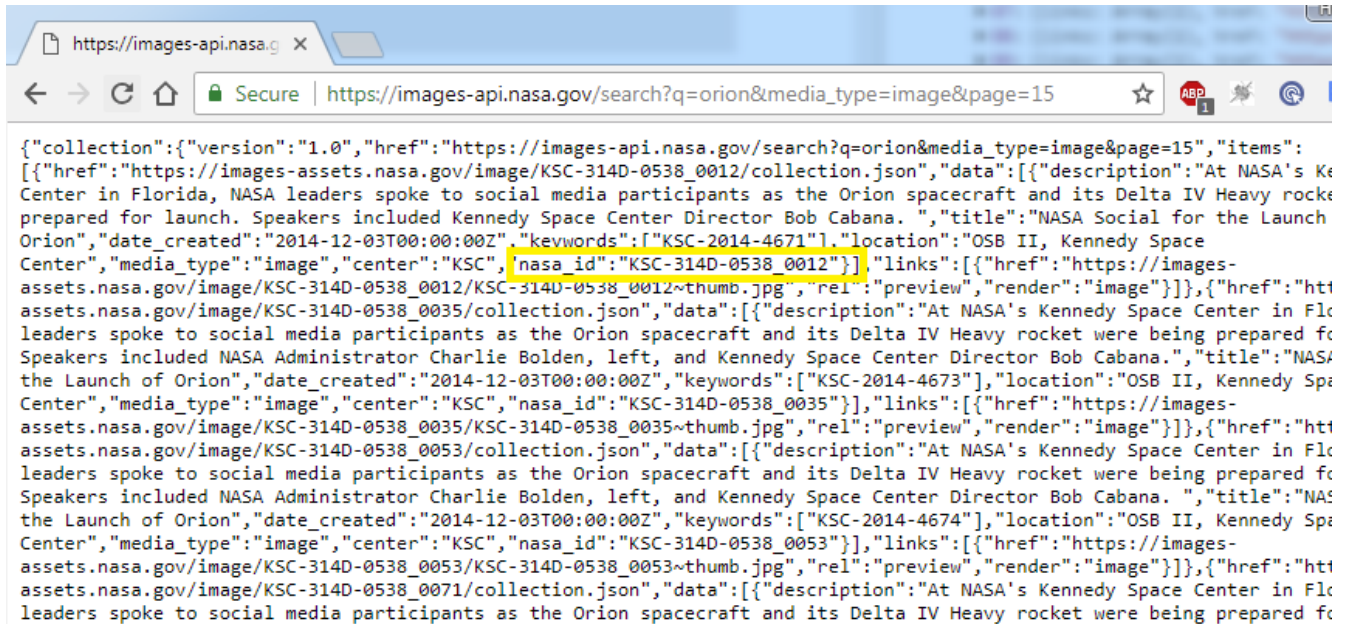
Please enter text to search the NASA Library.

SEARCH NASA

CLEAR RESULTS

<p><b>Library:</b> If the user clicked the search button and no results were returned, a message displays informing the visitor that there were no results for that search query and media type.</p>	
<p><b>Library Results:</b> Links were checked against the source JSON data to ensure that the correct link was being created for its matching data. The <b>Center</b> link was being created from the <code>getNASACenter()</code> function.</p>	
<p><b>EPIC:</b> If the user clicked the search button without entering a date to be searched, a message displays informing the visitor that they must enter a date to search.</p>	
<p><b>EPIC:</b> If the user clicked the search button and no results were returned, a message displays informing the visitor that there were no results for that date and image type.</p>	

To ensure that the Library results were being paged correctly, I viewed the JSON file in a browser noting the first **nasa\_id** item and the last **nasa\_id** item. These values were then used to ensure that the paging was being performed correctly.



## Ongoing Testing

Browser/Test	Chrome	Firefox	IE	Chrome Android-Remote Debugging
Intro section	Passed	Passed	Intro header formatting issue	Passed
Content Display	General formatting issues	General formatting issues	General formatting issues	General formatting issues
Navigation, hover, etc	Passed	Passed	Passed	Passed
Responsive Navigation	Passed	Passed	Passed	Passed
Data Section	Formatting issues	Formatting issues	Formatting issues	Formatting issues
Error messages	Passed	Passed	Passed	Passed
Popover full desc	Not working	Not working	Not working	Not working

Device/Test	Galaxy SIII	Galaxy 5	Laptop touch screen	iPhone 5/SE	iPhone 6/7/8	iPhone 6/7/8 Plus	iPhone X	iPad
Intro section	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Content Display	Formatting issues	Formatting issues	Formatting issues	Formatting issues	Formatting issues	Formatting issues	Formatting issues	Formatting issues

Navigation, hover, etc	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Responsive Navigation	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Styling	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues
Error messages	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Search results	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues	Styling issues
Data section	Not responsive	Not responsive	Passed	Not responsive	Not responsive	Not responsive	Not responsive	Passed
Popover full desc	Not working	Not working	Not working	Not working	Not working	Not working	Not working	Not working

## Usability Testing

During usability testing, it was found that the user would prefer if the page scrolled to the results section after performing a search. This made it more obvious to users on mobile devices that the results appeared after the search input area.

## Final Testing

Browser/Test	Chrome	Firefox	IE	Chrome Android-Remote Debugging
Intro Section	Passed	Passed	Passed	Passed
Content Display	Passed	Passed	Passed	Passed
Navigation, hover, etc	Passed	Passed	Passed	Passed
Responsive Navigation	Passed	Passed	Passed	Passed
Styling	Passed	Passed	Passed	Passed
Data Section	Passed	Passed	Passed	Bar charts hidden
Popover full desc	Passed	Passed	Passed	Passed

Device/Test	Galaxy SIII	Galaxy 5	Laptop touch screen	iPhone 5/SE	iPhone 6/7/8	iPhone 6/7/8 Plus	iPhone X	iPad
Intro section	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Content Display	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Navigation, hover, etc	Not Applicable	Not Applicable	Passed	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Passed

Responsive Navigation	Passed	Passed	Not Applicable	Passed	Passed	Passed	Passed	Not Applicable
Styling	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Error messages	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Search results	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Data section	Bar charts hidden	Bar charts hidden	Passed	Bar charts hidden	Bar charts hidden	Bar charts hidden	Bar charts hidden	Passed
Popover full desc	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed

## Audit

Page/Criteria	Performance	Accessibility	Best Practices	SEO
SPA	64	72	88	89

## Data section

It was decided that because the charts using D3.js are not responsive as they are designed for desktop or large-screen viewing, these two bar charts can only be viewed in the desktop version of the SPA.

## Deployment

GitHub is used to host the code and publish the pages.

A new repository was created in GitHub called: **interactive-front-end-project**

After a final Git Add and Git commit

```
$git add .
```

```
$git commit -m "Final commit"
```

The pages were pushed to the new GitHub repository

```
$ git remote add origin https://github.com/Sonnerz/project02-interactive-front-end-project
```

```
$ git push -u origin master
```

```
$Username
```

```
$Password
```

Under the **Settings – GitHub Pages** of the new repository, the master branch of the code is published to the url: <https://sonnerz.github.io/project02-interactive-front-end-project>

# External Help

Site	url	Resource
Stack Overflow	<a href="http://stackoverflow.com">http://stackoverflow.com</a>	CSS issues and in particular the pseudo class solution for setting the opacity of the background images.
Stack Overflow/Slack	<a href="http://stackoverflow.com">http://stackoverflow.com</a>	Popovers would not work after results were displayed
w3schools	<a href="https://www.w3schools.com/">https://www.w3schools.com/</a>	Multiple css styles were researched here
Bootstrap Date Picker	<a href="https://bootstrap-datepicker.readthedocs.io">https://bootstrap-datepicker.readthedocs.io</a>	This site provided the full solution for the date picker used in the EPIC date input
Slack - Simen Daehlin		Helped with the initial Fetch to the NEO API Suggested using FETCH
Slack - Chris Zelinski		Helped with understanding CORS Helped with paging of library results. My paging code was in the wrong location.
Astronomy calcs	<a href="https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php">https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php</a>	Distance between coordinates to work out distances
Pagination on EPIC images	<a href="http://www.thatsoftwaredude.com/content/6125/how-to-paginate-through-a-collection-in-javascript">http://www.thatsoftwaredude.com/content/6125/how-to-paginate-through-a-collection-in-javascript</a>	Learned how to paginate an object using array.slice when no pagination is supplied by the API
NASA API	<a href="https://api.nasa.gov/api.html#EPIC">https://api.nasa.gov/api.html#EPIC</a>	Official NASA API documentation

---

Bootstrap	<a href="https://getbootstrap.com/docs/4.1/getting-started/introduction/">https://getbootstrap.com/docs/4.1/getting-started/introduction/</a>	Official Bootstrap documentation for version 4.1
-----------	---	--