Corporate Website Documentation

Contents

Technology used

- **Base Technology** – HTML5, CSS3, Javascript (ES5)
- **CSS Preprocessor** – SASS (.scss files)
- **Javascript framework**
  i. jQuery (version 3.2.1)
  ii. VueJS (non cli version ; version 2.x)
- **Javascript libraries**
  i. Jquery.dlmenu.js (for responsive menu; for more info on this, find jquery.dlmenu.js under "Folder Structure and Files > Assets Folder > JS Folder > Folders > Vendor > Common" section)
  ii. Owl-carousel.min.js (for carousel; for more info on this, find owl-carousel.js under "Folder Structure and Files > Assets Folder > JS Folder > Folders > Vendor" section)
  iii. GSAP (mainly for animation; for more info on this, go to "Folder Structure and Files > Assets Folder > JS Folder > Folders > Vendor > GSAP" section)
- **Development and build technologies**
  i. Npm scripts
  ii. Node JS
  iii. Selenium

  For more info on this, go to "Development & Build" section.

Browser Compatibility

This website is compatible in all browsers and IE 11 and above.

Actually entire website is supported in IE10 but because of onetrust cookie consent script, which disables jquery code in IE 10.

In non-supported browsers, website shows browser compatibility message.

The code for browser compatibility is written in assets/js/compatibility-detector.js.

For more info about the logic of compatibility-detector.js, open that js file and read comments.

Responsive

This website is made completely responsive for all resolution minimum from 280px to infinite (best upto).
For your info, smallest browser resolution is 320px and highest browser resolution is 2560px (used in iMac mostly).

Folder Structure and Files

Each and everything resides in "en" folder, which has following folders.

Each folder has web.config file which can be upload to live server (to upload files on demo server, go to "Upload" section).

1. Assets Folder

All the assets like css, fonts, images, js and json files are resides in this folder. Following are the files and folders

- **Files**
  - **All-countries-data.json** – this file has data about countries displayed in country dropdown. (for more info about this data, go to "Content/Data Changes > Country Dropdown Data" section)

- **CSS Folder**
  - **dlmenu.css** – for responsive menu and child menu e.g. Blog Page sub section, News sub section (use dlmenu.min.css for linking, link is given in style.scss)
  - **map.css** – for map used in footprint page (for more info about footprint map, go to "Footprint" section)
  - **owl-carousel.min.css** – for carousel (used in tabs in individuals home page, each blog article page, each government solution page and each executive board member page)
  - **style.scss** – all website styles are created in this file (for more info about styles go to "Styles" section)

- **Fonts Folder**
  Font file format used are .eot, .svg, .ttf, .woff. Each folder has font-face.css which contains @font-face declaration (use font-face.min.css for linking, link is given in style.scss).

  - **Icons Folder** – All the icons given in website uses this font. (for more info about icons go to "Icons" section)
  - **Inter-UI Folder** – This font is used for textual content in entire website. It has 2 font "InterUI-Regular" and "InterUI-Bold" for regular and bold font respectively.

- **Images Folder**
  It has all the image files even .svg image files are here. Following are the files and folders

  - **Files**
    - **all-browser-logos.png** – this image is used in showing browser compatibility message (for more info about browser compatibility, go to "Browser Compatibility" section).
    - **all-flags.png** – this image is used to show the flag of countries (used in dropdown; for more info about flags, go to "Flags" section).
    - **bg-map.png** – this image is used for showing map in background (used in governments home page and in milestones page)
    - **vfs-global-logo.svg** – this image is used to show logo of VFS Global (given in Header of all pages)
    - **vfs-global.logo.png** – this image is used to show logo in browser compatibility message, since IE8 doesn't support SVG.
    - **vfs-200-million-logo.svg** – this image is used to show 200 million applications celebration logo

  - **Folders**
    - **About** – All the images of "About" section from both Individuals and Governments section are here.
      - **Executive-Board** – This folder is inside About folder, since Executive Board is in About section of Governments. All the image of executive board members are here. All the files stored here must be of resolution 450 * 450 and of format .jpg.
    - **Banners** – All the blue colored banner images are here.
    - **Blog** – All the images used in "Blog" section under Individuals section are here.

      - **Sub Section Folder** – Sub sections of blog are "Visa Application Guide", "About", "Featured", "People & Places" and their respective folders are "visa-application-guide", "about", "featured", "people-places".
        Images used in content of each article page of each section are stored here and main article image is stored resolution wise.
        - **Resolution Folder** – Each and every sub section of blog has 3 resolution folders i.e. "extra-large", "large", "medium". These are folder where main image of article is stored.
          i. **Extra-large** – All the images stored here are used for top banner in article page. Image resolution must be 1500 * 640
          ii. **Large** – All the images stored here are used for showing in article in Individuals home page.
          iii. **Medium** – All the images stored here are used in article box (e.g. see any sub section page like blog-visa-application-guide.html; to see html page in browser, go to "Run" section).
    - **Certification** – certification images used in footer are stored here.
    - **Footprint** – Images required for footprint page are stored here.
    - **Government-services** – Images shown in "Government Services" tab under Individuals home page are stored here. All images must be of maximum width 58 and maximum height 38.
    - **Icons** – favicon (shortcut icon) and icons used in plain textual content are stored here.
    - **News** – All the images used in "News" section under Governments section are here.
      - **CSR** – All the images used in "Social Responsibility" under "News" section are stored here.

- **Events** - All the images used in “Events” under “News” section are stored here.

  Image resolution in both “CSR” and “Events” must be of 440 width and height must be no more than 330.

- **Solutions** – All the images of “Governments > Solutions” are stored here. All the files stored here must be of resolution 1500 * 640 and of format .jpg.
  (Note: Service Icon of each Governments Solutions are not an images those are font icons, to know more about icons go to “Icons” section).
- **Tourism-services** - Images shown in “Tourism Services” tab under Individuals home page are stored here. All images must be of maximum width 58 and maximum height 38.

- **Js Folder**

  All the java script files are here. Following are the files and folders. Logic of each and every code is written in comment in respective js files. (to do any changes in any js file and view in browser, go to “Run, Development, Build & Upload” section).

  To know more about structure of website from javascript perspective, go to “Javascript Structure”

  - **Files**
    - **Compatibility-detector.js** – code for browser compatibility is written here. It is linked in each and every html file. (linked compatibility-detector.min.js)
    - **Config.js** – configuration of website and even some data that are to be shown in multiple pages to be taken from here (for more info read comments in config.js file)
    - **Main.js** – initializes vfs modules (written in vfs-global.js) and vue application (written in vue-app.js). Loading of One trust cookie consent js file is wriiten here. (for more info read comments in main.js file)
    - **Vfs-global.js** – all the modules required throughout the website are written here. Modules are exported in window object that means you can see list of modules and sub modules in console by typing vfs. (for more info read comments in vfs-global.js file)
    - **Vue-app.js** – all the code related to Vue is written here. (for more info read comments in vue-app.js file)
    - **Map.js** – js code for footprint map is written here. (linked map.min.js)
    - **Script.min.js** – this file is linked in all pages which has code of all 4 files like vfs-global.js, vue-app.js, config.js and main.js

  - **Folders**

    - **Vendor**
      All the 3rd party plugins and libraries are here. Following are the files and folders.

      - **Files**

- **owl.carousel.js** – for carousel (used in tabs in individuals home page, each blog article page, each government solution page and each executive board member page).
  (linked owl-carousel.min.js)
  Important Note: Don't replace it with original Owl Carousel library from plugin's website since we have changed code of it to solve error for autoWidth enabled carousel used in tab buttons.
- **gsap.min.js** – it is auto generated from "GSAP" folder (for more info about how it is generated automatically, go to "Build > npm run build-all > Minification Process" section)
- **common.min.js** – it is auto generated from "common" folder (for more info about how it is generated automatically, go to "Build > npm run build-all > Minification Process" section)

- **Folders**

  - **Common**
    All the libraries here are linked in all html files, and common.min.js is getting generated automatically (for more info about how it is generated automatically, go to "Build > npm run build-all > Minification Process" section). Following are the files and folders.

    - **Files**
      - **jquery-3.2.1.min.js** – jQuery plugin
      - **vue.min.js** – VueJS plugin, this is automatically replaced during development. (for more info about why and how it is getting replaced, go to "Build > npm run build-all > Minification Process" section)
      - **modernizr.custom.js** – this plugin is used for responsive menu

    - **Folders**
      - **Jquery-plugins** – all the jquery dependent plugins are stored here. Following are the files and folders.

        - **Files**
          - **Jquery.dlmenu.js** – this plugin is used for responsive menu and child menu e.g. Blog Page sub section, News sub section

      - **Vue-resource** – VueJS file both minified and unminified are here.
        When website is in development mode unminified file is used and in live website minified version is used. (for more info about why and how it is getting replaced, go to "Build > npm run build-all > Minification Process" section)

  - **GSAP**
    GSAP is a js tool for high performance animation (for more info about GSAP visit https://greensock.com/gsap). Following are the files and folders.

- **Files**
  - **TweenMax.min.js** – main library for animation (used in 200 million logo animation and milestones, for more info go to vue-app.js million-200-logo and milestones component)

- **Folders**
  - **Utilities** – all the GSAP utility tools are here. Following are the files and folders.

    - **Files**
      - **Draggable.min.js** – this library is used to drag element horizontally, vertically or rotationally. It is used in milestones page.

2. General Folder

This folder has all html files which are common to both "Individuals" and "Governments" sections.

This folder has sitemap.xml file which has urls of all html files resides in this folder except footprint-map.html file since it's an iframe which is called inside footprint.html page. Sitemap.xml is being automatically generated (for more info about how it is generated automatically, go to "Build > npm run build-all > Custom Build" section)

Also it has robots.txt file which has link to sitemap.xml file.

3. Individuals Folder

This folder has all html files which are only for "Individuals" section.

This folder has sitemap.xml file which has urls of all html files resides in this folder. Sitemap.xml is being automatically generated (for more info about how it is generated automatically, go to "Build > npm run build-all > Custom Build" section)

Also it has robots.txt file which has link to sitemap.xml file.

4. Governments Folder

This folder has all html files which are only for "Individuals" section.

This folder has sitemap.xml file which has urls of all html files resides in this folder. Sitemap.xml is being automatically generated (for more info about how it is generated automatically, go to "Build > npm run build-all > Custom Build" section)

Also it has robots.txt file which has link to sitemap.xml file.

5. PDF Folder

This has all the pdf files used in website. Following are the files and folders

- **CSR Folder**
  This has all the pdf files that will be used in CSR (Corporate Social Responsibility) page.

- **Media-releases Folder**
  This has all the pdf files that will be used in Media releases page. This has year wise folder which has pdf files, which will be linked to content of that year in media releases section.

Content/Data Changes

Note: Simple way to update content/data would be find related text in entire folder; then open found file and update.

Data other than listed here are given in config.js and vue-app.js. If you want to know about that data which is not listed here, find keyword of that data in config.js and vue-app.js and read comments.

For Milestones or Footprint related content/data changes are given in "Milestones" and "Footprint" section respectively.

Below are given the list of content/data section with files and location.

1. Config.js

This file resides in assets/js/config.js. Below are contents which are in config.js file.

A. News Sub Section Content

Just find all_content_details in config.js and then find news subsection name. Below are the list of news subsection.

a. **media_releases** – it has link to pdf files which is stored in PDF/media-releases folder. To know more about folder and file structure, go to Media-releases Folder.
b. **news_coverage** – it has external links
c. **events** – it has images which is stored in assets/images/news/events. To know more about images, find **Events** in Images Folder > Folders section.
d. **Awards** – it is plain content without any image.
e. **CSR** – it has images which is stored in assets/images/news/csr. To know more about images, find **CSR** in Images Folder > Folders section.

To know detail about each sub section content, open config.js and read comments.

To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

B. Navigation Menu Content

Just find navigation_page_details in config.js and then find the respective text to update.

To know detail about navigation content, open config.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

Below are some content which are given in navigation_page_details array.

- **Executive board members**

  Html file name of each board member must be given as board-member-(fullname).html e.g. board-member-zubin-jal-karkaria.html

  List of executive board members are given here. Just find executive_board in navigation_page_details.
  For more info about board member Box content, open config.js and read comments.
  Detailed html content of each and every board members are given in respective html page itself.

  Board member images are stored in assets/images/about/executive-board. To know about images of board members, find **Executive-Board** folder in Images Folder > Folders section.

- **Governments solutions**

  Html file name of each government solution must be given as solutions-(solution-name).html e.g. solutions-lidpro.html

  List of governments solutions are given here. Just find solutions_services in navigation_page_details.
  For more info about government solution Box content, open config.js and read comments.
  Detailed html content of each and every Government solutions are given in respective html page itself.

  Banner Images of Government solutions are stored in assets/images/solutions. To know about images of government solutions, find **Solutions** folder in Images Folder > Folders section.

  To know more about solution icons go to "Icons" section.

- **Blogs**

  Html file name of each blog sub section must be given as blog-(sub-section-name).html e.g. blog-featured.html, blog-visa-application-guide.html, etc.

  Html file name of each article (post) must be given as article-(article-name).html e.g. article-simple-visa-tips-for-a-great-start.html, article-setting-up-your-finances-abroad.html, etc.

  List of articles (posts) are given here. Just find blog in navigation_page_details.
  For more info about article Box content, open config.js and read comments.
  Detailed html content of each and every articles are given in respective html page itself.

Images of Article are stored in assets/images/blog. To know about images of blog article, find **Blog** folder in Images Folder > Folders section.

2. Vue-app.js

This file resides in assets/js/vue-app.js. Below are contents/data which are in vue-app.js file.

I. Certification and Accreditation Content

Just find certification_content in vue-app.js and then find the respective text to update.

To know detail about this content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

II. Individuals Home Page Tabs Heading and Related 3 Steps Content

Just find service-tabs-3-steps-container-with-content in vue-app.js and then find the respective heading under template to update.

3 Steps Content is given in vue-app.js (just find service-tabs-3-steps-container-with-content details in vue-app.js and then find the all_step_content under data)

To know detail about these contents, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

III. Government Services in Individuals Home Page

Just find government-services component definition in vue-app.js and then find the respective content under data function.

Government services images are stored in assets/images/government-services/. To know about images of Governments services, find **Government-services** folder in Images Folder > Folders section.

To know detail about government services content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

IV. Tourism Services in Individuals Home Page

Just find tourism-services component definition in vue-app.js and then find the respective content under data function.

Tourism services images are stored in assets/images/tourism-services/. To know about images of Tourism services, find **Tourism-services** folder in Images Folder > Folders section.

To know detail about tourism services content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

V. VDash Widget Top line in Individuals Home Page

Just find vdash-widget component definition in vue-app.js and then find the respective content under data function.

To know detail about this content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

VI. Company Profile Content

This content is displayed in below two places.

i. Individuals > About
ii. Governments > About > About VFS Global > Profile

This content is given in vue-app.js (just find company-profile-content in vue-app.js and then find the content under template)

In this content we have linked stats data. (For more info about stats data, go to "Content/Data Changes > Stats Data" section)

To know detail about company profile content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

VII. Contact Media Queries Content

This content is displayed in below two places.

i. Individuals > Contact
ii. Governments > Get in Touch

This content is given in vue-app.js (just find contact-media-queries in vue-app.js and then find the content under data function)

Note: We have obfuscated email link. (For more info about email obfuscation, go to "Content/Data Changes > Email Obfuscation" section)

To know detail about contact media queries content, open vue-app.js and read comments.
To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

VIII. Stats Data

Below are the list of stats which are used in website.

- **Manual**
  These are the list of stats that are given manually in stats object in vue-app.js file (just find stats in vue-app.js file)

  i. **Number of Application Centers** - Key: application_centres
  ii. **Number of Countries of Operation** – Key: operation_countries
  iii. **Number of Continents** – Key: continents
  iv. **Number of Client Governments** – Key: client_governments
  v. **Number of Employees** – Key: employees
  vi. **Number of Biometric Enrolments (in millions)** – Key: biometric_enrolments_in_millions
  vii. **Number of Applications** – Key: applications
  viii. **Date of Stats** – Key: stats_date

- **Computed**
  These are the list of computed stats that are computed programmatically using above data. Which are also given in vue-app.js file (just find applications_str in vue-app.js file)

  i. **Number of Applications in comma separated format** – Key: applications_str (computed from stats.applications)
  ii. **Number of Applications (in millions)** – Key: applications_in_millions (computed from stats.applications)
  iii. **Date of Stats by adding ordinal indicator** – Key: stats_ordinal_indicator_date (computed from stats.stats_date) e.g. if stats_date = 31 July 2019 then computed ordinal indicator date will be 31st July 2019.

We also show stats data in Individuals > About in circles.
To change text or icon there, just go to vue-app.js file and find all_stats_details array. (For more info on this, go to vue-app.js and read comments)

Pl. note these data can be linked in html page inside master-page tag and also in Vue component template using vue interpolation or v-dynamic-html (For more info about v-dynamic-html, go to "Javascript Structure > Vue-app > Directives > dynamic-html").

E.g. Total number of applications are over {{applications_in_millions}} and {{stats.client_governments}} client governments.

These stats data are shown in many places like Government Home Page, Individuals/Government > About Us Page, Milestones page etc.

To know about how to run and update changes in web browser, go to "Run, Development, Build & Upload" section.

3. All-countries-data.json (Country Dropdown Data)

Country dropdown data is used in visa/permits and passport services in individuals home page.
It is given in all-countries-data.json file.

Below are data that are given in all-countries-data.json

A. Visa/Permits and Passport Services

Key of these data are "visa_and_permits" and "passport_services".

This data is given in below format.

Note: Resident country name in an array must be unique. And also visiting country in that resident country must be unique.

```
[{
      "name": "resident country name",
      "visiting": [{
         "name": "visiting country name",
         "url": "website url",
         "contact_url": "url of contact page" // this will be shown in
individuals contact page
      }, {
         …..
      },
      …..]
},
{
   …..
},
…..
]
```

B. Geographic Details

Key of this data is "geographic_details".

This contains data about country flag and country code of each and every resident and visiting countries. See below for description of each

i. **Flag** – it stores the position of flag of current country in all-flags.png sprite. Format of position is [x, y] (for more info about flags, go to "Flags" section).

ii. **Country Code** – it stores the iso-2-letter code of current country. It will be used in preselecting the user's country based on the ip.

Note: Don't add same geographic details for different country name. If you want to, then add entry in geographic_duplicates.

C. Geographic Duplicates

Key of this data is "geographic_duplicates".

This contains the entry of those country names which have duplicate flag and country code.

e.g. "Finland Residence Permit" has same geographic details as "Finland";
"Saudi Arabia & GCC Residents" has same geographic details as "Saudi Arabia, Kingdom of" (Kingdom of is given after country name because in footprint map we want to show "Saudi Arabia, Kingdom of" in "S" letter rather than "K" letter; for more info about footprint map, go to "Footprint" section).

D. Country Aliases

Key of this data is "country_aliases".

This contains the data about alias names of countries.
E.g. "United Arab Emirates" – alias name is "UAE"
"Côte d'Ivoire (Ivory Coast)" – alias name is "Cote d'Ivoire" (character "ô" and "'" is changed to "o" and "'" respectively)

This data is used while user searches the dropdown.
e.g. If user has typed "UAE" he must see "United Arab Emirates" option in dropdown.

4. Email Obfuscation

We have obfuscated email id's in entire website wherever used. Email obfuscation is required. Search google for why it is required. We have tool to obfuscate email ID. Visit below link

http://210.210.25.188/vfsdemowebsites/New_Template/Encoder-script/

Run, Development, Build & Upload

1. Run

To run this website in browser, we need localhost server.

Install "200 ok" chrome app which is a tool that will create localhost server. After installing run the app and choose "en" folder.

2. Development & Build

This website uses “npm scripts” and custom node script for development and build purpose.

The node scripts are in “_build” folder under “en” folder.

Important Note: First you need to install below npm packages to make “npm scripts” workable.

Make sure you run below command as it is without editing. Since some packages we require to install globally

```
i. npm install --global cpy-cli
ii. npm install --global uglify-js
iii. npm install --global rimraf
iv. npm install --global npm-run-all
v. npm install --global ncat
vi. npm install --global chokidar
vii. npm install --save-dev cli-progress
viii. npm install --save-dev pretty
ix. npm install --save-dev selenium-webdriver
```

Also, after installing all the above packages we need to download selenium driver for chrome http://chromedriver.chromium.org/downloads.

To allow selenium webdriver workable from nodejs, you need to add folder path of driver in PATH variable (For windows, Right Click Computer > Properties > Advanced System Settings > Advanced Tab > Environment Variables > Under System Variables section > In Variable column find Path).

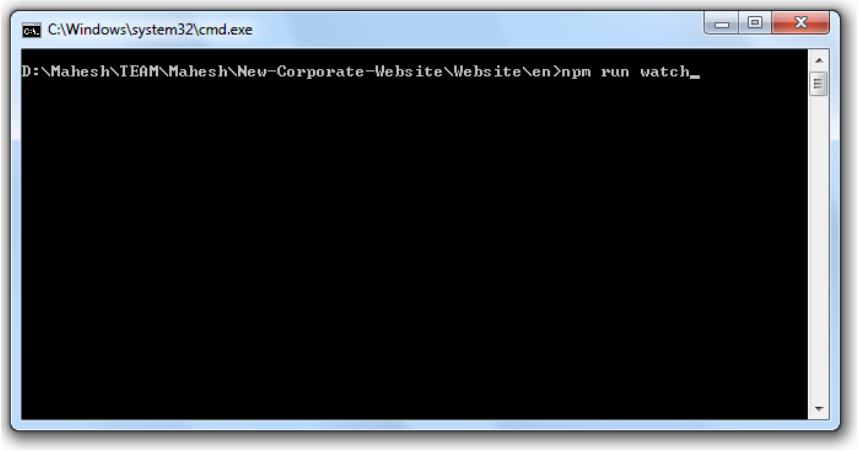
We need Selenium Webdriver for build purpose. For more info on why we need selenium webdriver, go to “Build > npm run build-all > Custom Build” section.

Important Note: Whatever changes you do, pl. check responsive also.

A. Development

- **JS / JSON**
  If you are changing in one of the below files, then you need to run “npm run watch” in command window in “en” folder

  i. compatibility-detector.js
  ii. vfs-global.js
  iii. config.js
  iv. all-countries-data.json
  v. vue-app.js
  vi. main.js
  vii. map.js



It will start website for development purpose. When you edit in any of the above listed files, after avgerage 4-5 seconds you can refresh the browser to see changes. (For more info on how to run website in browser, go to "Run" section)

What does "npm run watch" do?

When you run "watch" command, it copies above unminified js files to minified js file (which is linked to all the html files) for faster development purpose.

After copying all unminified files to their appropriate location, it starts running chokidar, which watches for the file changes and on change of any file it again copies unminified files.

For detailed info, go to package.json and under that understand the commands given in "scripts" object.

- **Style.scss**
  In all html files style.min.css is linked so we need conversion tool, if you are using visual studio code editor then you can install "Easy Sass" extension. This extension convert .scss files to .css and .min.css in same folder path on save of .scss file. (For more info about this extension, visit https://marketplace.visualstudio.com/items?itemName=spook.easysass)

  (for more info about styles go to "Styles" section)

- **Other files**
  If you are changing in html files then you don't need to run any command or don't need any tools.

  For below files, you need to do minification manually.

  i. **font-face.css**
  Both the font-face file of "InterUI" and "Icons" font are linked in style.scss (but font-face.min.css is linked instead of font-face.css)

  So an easy way is,
  During development purpose, in style.scss change link from font-face.min.css to font-face.css.

After Development completes, minify font-face.css online and change link in style.scss back to font-face.min.css

ii. owl.carousel.js

This file is not linked in any html pages instead owl.carousel.min.js file is linked.
So during development, you need to change owl carousel link from .min.js to .js in one of the html file.
After development completes minify js online and change html file owl carousel link back to the .min.js

iii. jquery.dlmenu.js

This file is not linked directly, instead all html file has link to vendor/common.min.js, which has minified code of third party plugins that are required in all html pages.
So after changes in jquery.dlmenu.js file run "npm run build" command. (for more info on build, go to "Build > npm run build" section)

iv. dlmenu.css

It is linked in style.scss (but dlmenu.min.css is linked instead of dlmenu.css)

So an easy way is,
During development purpose, in style.scss change link from dlmenu.min.css to dlmenu.css.
After Development completes, minify dlmenu.css online and change link in style.scss back to dlmenu.min.css

B. Build

If you do changes in any js file or if you have added new html pages, and want to upload, then you need to run build command in command window.

There are 2 build commands "build" and "build-all" depending on what changes you did.

- **npm run build-all**
  This command is a complete build command, no matter whatever changes you have did in website this command will make website ready to upload. (for more info on how to upload, go to "Upload" section)

  Note: While running this command, there must be localhost server created for "en" folder. (To know about how to create localhost server go to "Run" section)

  What does "npm run build-all" do?

  It minifies all javascript files and then does custom build. See below for more info.

I. Minification Process

It uses unglify-js package to minify javascript files. (For more info on how to install uglify-js package, go to top of "Development & Build" section)

All the below minification process runs parallel (means **not** one after another)

i. It minifies and creates vendor/common.min.js from all the third party libraries which are used in all html pages. But for this it needs vue.js minified file, so first it copies vue.min.js from vendor/common/vue-resource/vue.min.js to vendor/common/vue.min.js (it copies using npm scripts which uses "cpy" package) and then it begins minification. (for more info on structure VueJS used in website, go to "Javascript Structure > Vue-app.js" section)
ii. It minifies and creates vendor/gsap.min.js from all GSAP libraries.
iii. It minifies and creates vendor/owl-carousel.min.js from vendor/owl-carousel.js
iv. It minifies and creates script.min.js from all the custom script used in all the html pages like vfs-global.js, config.js, all-countries-data.js (this is being generated from all-countries-data.json file from node script of name create-data-js.js), vue-app.js and main.js.
v. It minifies and creates map.min.js from map.js which is used in footprint map (for more info about footprint map, go to "Footprint" section)
vi. It minifies and creates compatibility-detector.min.js from compatibility-detector.js. It is used to check browser compatibility (For more info about browser compatibility, go to "Browser Compatibility" section)

II. Custom Build

After All the minification done, it runs custom build (which is node script written in _build/index.js) which does following process.

i. Generates 3 sitemap.xml files each for Individuals, Governments and General Folder. Each sitemap.xml file has list of html pages resides in those folders.
ii. Modifies all blog article html pages for Open Graph Protocol, so that each article pages can have proper sharing option to facebook, twiiter and linkedin. Since we need blog article data from config.js file for this process; we have used selenium webdriver.
Note: Sharing option will work properly only on live server.

- **npm run build**

This command does the short job of what "build-all" command does. (click here to see what build all command does)

It does all the minification of javascript files, but skips the custom build process as it takes time. So it you have not added/deleted/renamed any html page and also not changed blog article data in config.js then you can run this command.

For more info on which javascript files are getting minified, go to "npm run build-all > Minification Process" section.

3. Upload

Only the following folders are used on server side

i. assets
ii. Individuals
iii. Governments
iv. General
v. PDF

Don't upload any other folders on server side.

Important Note: If you upload any of the above entire folder on demo/testing server, then delete **web.config** files from all the 5 folders from server, since web.config files are written for live server only.

What to upload and when?

i. If you have done edits in any javascript file then upload entire JS folder.
ii. If you have run "npm run build-all" command then upload all the three folder (Individuals, Governments, General), since it modifies **sitemap.xml** file. (For more info on when to run build-all command, go to "Build > npm run build-all" section)
iii. If you done edits in style.scss then upload additional style.css and style.min.css files.

Flags

Flags of all countries that are used in country dropdown is the sprite image. The path of that sprite image is assets/images/all-flags.png.

How to add new flag?

First create flag of that country of size 28 * 20 and add that flag image to sprite image after last flag of last row (if there is no space after that last flag then increase space below i.e. canvas size; and then add flag image).

Since it is sprite image, we need to store position of newly added flag. Position of each flag is stored inside geographic_details in assets/all-countries-data.json file.

The format of geographic_details is given below.

```
{
	"Country Name": {
		"flag": [x, y],
		"country_code": "…"
	},
	…
}
```

In `"flag": [x, y]`
x is x-coordinate, y is y-coordinate and both is number which starts from 0
e.g. First flag is of Algeria so position is [0, 0]
Second flag in first row is of Australia so position is [1, 0]
India flag position is [9, 1]

To know more about geographic_details or country_code, go to "Country Dropdown Data > Geographic Details" section.

Icons

This website uses fonts for monocolor icons like font-awesome do.

The font files and their font face declaration file is stored in assets/fonts/icons folder.

The icons are displayed by giving class name (class name can be given to any tag but generally 'i' tag).
We also have 'icon' vue component that are used in some other components like icon-heading-wrapper and icon-details
(For more info on VueJS structure used in website, go to "Javascript Structure > Vue-app.js")
(For more info on 'icon' vue component, open vue-app.js file and find icon component and read comments)

The List of icon class names are listed in font-face.css file in icons folder under fonts folder.

There is convention of governments solution icon i.e. use icon-solution-(solution-name).
E.g. icon-solution-fast-pass, icon-solution-internet-kiosk, etc.

How to add/change icon?

First create svg file of an icon. (You can use Illustrator software to draw or modify icon; and then export to svg)

Important Note: There are some restrictions for creating svg as icon for font. See below.

Restrictions in svg

i. SVG icon must be of mono color. That means all the drawing elements of the svg file must have same fill color.
ii. SVG icon must not have stroke. If you don't want to remove stroke then use Illustrator to convert stroke to path. (Open SVG in illustrator, then select element that has stroke and then go to Menu > Object > Path > Outline Stroke; that's it, stroke is now converted to path)
iii. Name of you icon svg file must be same as you want to use in your website by class (without "icon-" prefix). e.g. for svg file name must be "test-name.svg"
iv. It will be better if entire icon is in one path tag. You can use illustrator to do that. (In illustrator select all drawing elements; Right Click > Make Compound Path. If icon shape is changing then use "Window > Path Finder" and use your idea to combine shapes).
v. It will be better if there are no width and height attribute to svg tag.

Set Viewbox in svg

After creating proper svg icon file, give perfect square viewbox to it. To do this, see below

i. Open svg file in browser

ii. Run below code in console (this code will calculate the perfect box data to give in svg's viewbox attribute)

```
var svg = document.getElementsByTagName('svg')[0],

main_el = document.getElementsByTagName('g')[0] ||
document.getElementsByTagName('path')[0],
main_el_bounding_rect,
viewbox_arr,
viewbox_size_diff_half;

svg.removeAttribute('viewBox');

main_el_bounding_rect = main_el.getBoundingClientRect();

viewbox_arr = [
                main_el_bounding_rect.left,
                main_el_bounding_rect.top,
                main_el_bounding_rect.width,
                main_el_bounding_rect.height
];

viewbox_size_diff_half = Math.abs(viewbox_arr[2] - viewbox_arr[3]) / 2;

if( viewbox_arr[2] > viewbox_arr[3] )
{
viewbox_arr[1] -= viewbox_size_diff_half;
viewbox_arr[3] = viewbox_arr[2];
}
else
{
viewbox_arr[0] -= viewbox_size_diff_half;
viewbox_arr[2] = viewbox_arr[3];
}

console.log(viewbox_arr[0] + ' ' + viewbox_arr[1] + ' ' + viewbox_arr[2] + ' ' +
viewbox_arr[3]);
```

iii. After running above code; it will print viewbox attribute value; copy it and paste it in svg file's viewbox attribute

Now your svg file is prefect to add it in icon font. To do that we will use icomoon app which third party app, this is free of cost and license to make font from svg. See below steps to do this

Create font from SVG File

i. Open **icomoon app** in browser (https://icomoon.io/app/)

ii. Click on **Import Icons** from top left, and select website's icon svg font file (**assets/fonts/icons/vfs-icons.svg**)

iii. Click **Yes** from alert which says to use metrics and metadata on exporting font. (this will be used by icomoon app to give class name same as given in svg on exporting to font)
Important Note: If you clicked No, no icon will be displayed in website, since class name will not be given by icomoon app as same as we have given in website.

iv. You will see **vfs-icons** section on top in icomoon app.

v. If you want, you can delete icon by selecting "Remove" tool (on the right side of "Import Icons" button)
To replace icon delete icon which is to be replaced and then add new icon.
Note: To replace icon, make sure that new SVG file name is same as the old icon.

After Deleting icon, again select "Select" tool.

vi. **Drag your new svg icon file** in vfs-icons section in app.
After dragging you will see new icon. (If icon is not showing proper, then you may have not followed the steps written in "Restrictions in svg" and "Set Viewbox in svg")

vii. Now **select all the icons** in vfs-icons section.
You need to select all the icon even if you have added only one, since this app generates fonts for those icons which you have selected.
To multi-select, first mouse-press the first icon and drag mouse to last icon. By doing this you will get idea of how to multi-select.

viii. After selecting all icons in vfs-icons section, click on "**Generate Font**" button which is on the bottom-right part of screen.

ix. **Find out your newly added icon.**
If your new icon is not getting blue on mouse hover that means icon is not mono color. Please make it mono color and again follow the steps.
Text after the icon is the name of icon which will be used to give class.

x. **Change settings**
Click on gear icon, which is after "Download" button in the bottom-right part of the screen.

Set below values in field in popup form.
Font Name: vfs-icons
Class Prefix: icon-
Select Support IE8 checkbox.

xi. Close settings popup by clicking outside and the click on "**Download**" button.

xii. **Zip file** will be downloaded, **extract it**, **copy font files** from font folder to icon font folder.
Note: before replacing take backup of old vfs-icons font files.

xiii. Open style.css file from extracted folder, copy and paste entire code in icons/font-face.css file.

xiv. In icons/font-face.css file, in font-face declaration, remove 'fonts/' from all url given in all src.

xv. After completing modification in font-face.css, **minify it online** and paste minified code in icons/font-face.min.css.

DONE. You can use class name of your new icon anywhere in the website.
e.g.

If you are unable to see icon in website, you may have not properly followed steps for create font from svg file.

Grid System

Since this website doesn't use bootstrap, it has custom script for grid system.

In vfs-global.js there is module of name dynamic_columns.

What is dynamic_columns?

Number of columns applied to element changes over the resolution.

In html, if we have provided

```
<div data-column-details="test_column_details" class="col_type_normal">
      <div>Test</div>
      <div>Hello</div>
      <div>Hey</div>
      <div>Bye</div>
</div>
```

Script will find all elements which has data-column-details attribute and the value of it will be used to find configuration of dynamic columns, dynamic column configuration is given in config.js.

Then column will be applied according to found configuration.

Script only adds or removes classes and attribute to set column, and grid layout chages according css given in style.scss.

col_type_normal is class that defines space between columns as given in style.scss. To see what css is written open style.scss, find [data-column-details] and read comments.

What will be changed in html when column is applied?

When column is applied; script will add data-column attribute along with column number as it's value.
Script will also add classes like **first_row, last_row, first_in_row and last_in_row**.
first_in_row and last_in_row means first element and last element in that row respectively.
Also **first, second, third, fourth and fifth** classes for that corresponding element in that row.
To see this in website, go to Governments > Solution Page in website and then inspect any box 'li'.

dynamic_column config format

The format of dynamic_column in config.js is given below.

```
dynamic_columns: {
      column_options: {
            test_column_details: {
                  //column number
                  “3”: function() { //if this functions returns true; then column 3 will
be applied to corresponding element
                        //if window width is greater than 768; then column 3 will be
applied to corresponding element
                        //Note: It will be better to use resolution number as same as
given in media_resolutions in style.scss
                        return vfs.details.window_dimension.width > 768;
                  }
            },
            …
      },
      //extra_classes are used if you want to give any custom classes other than
predefined classes
      extra_classes: {
            //below key can be any selector
            ‘[data-column-details=“test_column_details”]’: {
                  all_extra_classes: “class1 class2 class3”, //all the classes given in
column_classes object
                  column_classes: {
                        “1”: “class1”,
                        “2”: “class2”,
                        “3”: “class3”
                  }
            }
      }
}
```

For any example, see dynamic_columns in config.js.

Styles

All the styles of website is written in style.scss (to know how to convert scss to css, go to “Development > Style.scss” section)

If you want to see **theme colors**, then in style.scss, find all variables.

For content this website doesn't use entire screen/wrapper width. Means content only comes within width of 720px in an element. The class for that is '**max_content_width**'.

Methodology for responsive css?

In style.scss file, there are mixins created to make responsive easier and maintainable. See below list of mixins.

i. media
ii. multi_media
iii. add_css_from_map

To know more about any mixin, in style.scss find corresponding mixin definition name and read comments.

There are predefined media responsive resolution widths, which are used in calling above mixins.
Few of those media resolutions are
ipad: 1080px
tablet: 768px
mobile: 520px
you can see more of these in style.scss (find initialization of $media_resolutions)

Methodology for browser prefix css3?

In style.scss file, there is mixin created to make browser prefix css3 short and maintainable.

Mixin of name **css3** is used to give css3 properties.

You can use css3 in responsive mixins by giving "css3-" prefix to css property.

```
e.g. @include multi_media(css3-transform, (
        default: translateX(10px),
        ipad: translateX(7px),
        tablet: translateX(5px)
));
```

Methodology for browser prefix keyframe animation?

In style.scss file, there is mixin created to make browser prefix keyframes short and maintainable.

Mixin of name **keyframes** is used to create keyframe animation.

```
e.g. @include keyframes(fade) {
        0% {
                opacity: 0;
        }
```

```
        100% {
                Opacity: 1;
        }
}
.fade_animation {
        animation: fade 0.5s;
}
```

Methodology for linear gradient?

In style.scss file, there is mixin created to make linear gradient easier and maintainable.

Mixin of name **linear_gradient** is used to create linear-gradient.

e.g. @include linear_gradient(#FFF, to left, rgba(255, 255, 255, 0) 0%, #FFF 50%, rgba(255, 255, 255, 0) 100%);
First argument is fallback color for browser that doesn't support linear-gradient.

Classes for Show/Hide Element resolution wise

There are classes created for showing/hiding element by resolution. The format of class name is .show_from_(media resolution key), .hide_from_(media resolution key)

E.g.
.show_from_ipad: it will hide an element in desktop and will display an element if window width is less than or equal to 'ipad' (1080px) resolution.
.hide_from_tablet: it will display an element in desktop and hide an element if window width is less than or equal to 'tablet' (768px) resolution.
.show_from_big: it will hide an element in desktop and will display an element it window width is greater than or equal to 'big' (1400px) resolution.
Note for show_from_big: we have used greater than or equal to but not less than or equal to since for big media_resolutions key, value is negative (-1400).

Classes for Text center resolution wise

There are classes created for text-center for element by resolution. The format of class name is .text_center_from_(media resolution key)

E.g.
.text_center_from_ipad: it will text-align: center will be applied to an element if window width is less than or equal to 'ipad' (1080px) resolution.

Classes for Hide
 resolution wise

There are classes used to hide br tag inside an element by resolution. This can be used to remove line break in html. The format of class name is
.hide_break_from_(media resolution key)

E.g.
.hide_break_from_mobile: it will give display: none to all br tag inside an element if window width is less than or equal to 'mobile' (520px) resolution.

Classes for Headings

There are many classes created for headings, so that heading styles remain standard across entire website.

e.g. .main_big_heading, .big_heading, .medium_heading, etc.

For full list of heading classes go to style.scss and find "all heading classes and their styles".

Classes for Padding and margin

It has predefined classes for padding and margin.

The format of padding and margin class is
.(space type)(direction letter)_(main space key)
Where,
space type = pad (for padding), mar (for margin)
direction letter = t (for top), b (for bottom), v (for vertical i.e. top and bottom)
main space key = any space key defined in $space_details map variable;

E.g.
.padt_big, marb_small, padv_normal, etc.

To see complete list of all space keys, in style.scss find "all space keys".

More methodologies and predefined classes

There are many more predefined methodologies to write css more short, easier and maintainable. To see that open style.scss and start reading comments from top.

Also there are many more predefined classes. To see that open style.scss and start from body tag styles.

Javascript Structure

Entire website is created in jQuery and VueJS (non cli version)

There are four custom javascript files which resides in assets/js/.

1. Compatibility-detector.js

It detects browser compatibility, also print and displays browser compatibility message.

For more info on implementation of browser compatibility code, open compatibility-detector.js and read comments.

2. Config.js

It has following things

i. Configuration of modules written in vfs-global.js (to see what are configuration, open config.js and read comments)
ii. Some content that are used in multiple pages. (To see, what content are given, then go to "Content/Data Changes > Config.js")

You can see all the configuration in browser console. (Open local/live website in browser, open console; write 'vfs.data')

To see detailed info on each configuration, open config.js and read comments.

3. VFS-global.js

In this file, all the modules and their sub modules are implemented.

All the modules are stored in window.vfs, so that these modules can be used in other script.

You can see all the modules and sub modules in browser console. (Open local/live website in browser, open console; write 'vfs')

E.g. some of modules are vfs.global.init_google_tag_manager, vfs.replace_bold, etc.

Note: vfs.data has all the configuration provided from config.js.

- **Mostly used modules**
  Some of mostly used modules are given below

  Note: There are many more modules other than listed below.

  To see complete list of module, open local/live website in browser, open console; write 'vfs'.
  To see information about any module, open vfs-global.js find corresponding module implementation and read comments.

  I. global_variables

  This module has below global variables that are used in many other places in script.

  i. **$window** – it contains jquery object of window i.e. $(window)

ii. **$document** – it contains jquery object of document i.e. $(document)
iii. **$html** – it contains jquery object of html i.e. $('html')
iv. **$body** – it contains jquery object of body i.e. $('body')
v. **$html_body** – it contains jquery object of html, body i.e. $('html, body')
vi. **$head** – it contains jquery object of head i.e. $('head')

Important Note: Wherever you want to use above jquery object use this global_variables module only to reduce processing time of creating jquery object. See below example.

If you want to see implementation of this, open vfs-global.js and find "vfs.global_variables = ".

II. prototypes

This module has all the class whose instances are created and used in other parts of script.

Note: class definition is implemented in ES5 syntax, so you will not see any class keyword.

Below is the list of class definition given in prototypes module.

i. **Callback** – This is used to achieve built in concept of callback for custom modules.
ii. **Property** – This is used to create watchable property.

To see more info about above class definitions, then open vfs-global.js, find "vfs.prototypes = ".

III. details

This module has details about several sections which are used in other parts of script.

See below the list of details section

i. **page** – this contains details about current page such as page name, type, hash, etc.
ii. **window_dimension** – this contains details of window_dimension and also has callback for when window_dimension is changed
iii. **browser** – this contains details of browser such as browser name, version, support_transitions, support_canvas, etc.

To see more info about above details, then open vfs-global.js, find "vfs.details = ".

IV. utilities

This module has all the service functions that we need in all other modules. It's a biggest module and has many sub modules.

To see complete list of all the modules and sub modules, run live/local website in browser, open console, and write 'vfs.utilities'.

To see more info about any sub module, then go to vfs-global.js, find 'vfs.utilities', under that find corresponding module name and read comments.

- **Jquery plugin modules**

There are also jquery plugin modules. That means this website has implementation of it's own jquery plugin instead of third party jquery plugin.

Each jquery plugin module created by calling create_plugin_module. (to know more about this, open vfs-global.js, find “create_plugin_module:” and read comments)

To see information about any jquery module, open vfs-global.js find corresponding module implementation and read comments.

I. autocomplete

This plugin is used for creating both editable and non-editable dropdown.

E.g.

i. Country dropdown in Individuals Home Page is created from this plugin.
ii. Language dropdown (this is currently not in website, but can be added in future)

II. accordion

This plugin is used for creating accordion

E.g. See ‘Commitment’ section under ‘about.html’ page under ‘Governments’ section.

4. Vue-app.js

All the VueJS related code is written here.

Below is the list of what vue-app.js consists of.

A. Configuration

Below are the details provided in configuration.

I. app_selector

Value is selector, which will be used in ‘el’ in creating vue application. Currently ‘master-page’ is provided.

II. data_to_copy

Value is of below format. This will copy data from other sources.

```
{
    from_vfs_data: [], //provide array of key from vfs.data that you want to copy in vue application
    from_window: [‘master_page_data’, ‘page_data’] //provide array of key from window that you want to copy
}
```

III. app_mixins

Value is array of object. This will be provided in ‘mixins’ property in creating vue application. To see what details given in app_mixins, open vue-app.js and find ‘app_mixins’.

IV. all_mixins

Value is in below format. This is created for components so that if any component want to use any mixin provided here, then add mixin key in mixins array.

```
{
        mixin_key: mixin object,
        ….
}
```

E.g.

```
{
        'Comp-A': {
            a.  mixins: ['mixin_key']
        }
}
```

To see what mixins are provided, open vue-app.js and find all_mixins.

V. global_components

Value is in below format. Component configuration given here will be registered globally in Vue.

```
{
    1. 'component-name': {
            a.  … //component options
    2. }
}
```

To see all the components, open vue-app.js file, find global_components and erad comments.

- **master-page**

  master-page is an app selector and component also.

  This website uses concept of Master Page from .NET.

  Content of all html pages must be in 'master-page'.

  There are many master page configured, which are being used by providing type attribute in master-page.

  e.g <master-page type="basic">…</master-page> will use 'master-basic' component

  To see more master page components, open vue-app.js and find 'global_components'; all the components whose name starts with 'master-' are master page components.

B. Directives

Below is the list of directives that are created.

I. dynamic-html

This is the dynamic version of v-html.

What it does is, if value is passed as null then it will not set any html nor removes it otherwise if value is passed as non-null value then html will be set. To remove html, pass empty string ('') in value.

To see arguments of it or to see implementation of it, open vue-app.js and find 'all_directives' and under that find 'dynamic-html'.

Important Note: Use **v-dynamic-html** instead of v-html in all the places.

II. animate-height-on-change

This will be used to animate height of an element on 'updated' hook of component.

To see arguments of it or to see implementation of it, open vue-app.js and find 'all_directives' and under that find 'animate-height-on-change'.

C. Data Generation Code (generate_data)

This has implementation of following type of data generation. This data are placed in creating vue application.

I. Copied Data (function name: copy_data)

This is the type of data to be copied from other sources. The data to copy will be taken from configuration. See "Configuration > data_to_copy" for info on configuration.

For any info on implementation, open vue-app.js, find "generate_data" and under that find "copy_data".

II. Section Index (function name: set_section_index)

This is the section_index which has the value 0 or 1, if 'Individuals' or 'Government' is active respectively.

For any info on implementation, open vue-app.js, find "generate_data" and under that find "set_section_index".

III. Navigation Related Data (function name: navigation_related_data, set_additional_data_from_navigation)

This is the data which is related to navigation.

For any info on implementation, open vue-app.js, find "generate_data" and under that find "navigation_related_data" or "set_additional_data_from_navigation".

D. Vue Plugins

Below is the list of vue plugins that are created.

I. import_data

This plugin is used to import the data from parent component or from root component.

import_from_parent/import_from_root can be provided in component configuration options.

To see detailed implementation of this, open vue-app.js file, find 'all_plugin_details' and under that find 'import_data'.

- **import_from_parent**
  imports data from parent component or from any parent component in hierarchy.

  E.g.

  i. import_from_parent: [‘a’, ‘b’]
  This will import ‘a’ and ‘b’ from parent component. Means in child component I can access using this.a and this.b

  ii. import_from_parent: {
  a: 2
  }
  This will import ‘a’ from parent of parent component (since level is given as 2).

Note: If data in parent component changes then child component’s that data will also be changed (since in child component uses computed data for import)

- **import_from_root**
  Import data from root application.

  E.g.

  Import_from_root: [‘x’, ‘y’]

  This will import ‘x’ and ‘y’ from root application. Means in child component I can access using this.x and this.y

  Note: If data in root application changes then child component’s that data will also be changed (since in child component uses computed data for import)

II. outside_events

This plugin is used to trigger outer events (other than vue events/hooks) in component.

outside_events can be provided in component configuration options.

To see detailed implementation of this, open vue-app.js file, find ‘all_plugin_details’ and under that find ‘outside_events’.

Currently, it has 2 outer events

i. loaded (for window loaded)
ii. resized (for window resized).

E.g.

```
‘Comp-A’: {
    Outside_events: {
        loaded: function() {
        },
```

```
            resized: function() {
            }
        }
    }
```

III. get_text_from_html

This plugin adds custom function to vue prototype to make function available in all components.

This adds $get_text_from_html function which converts html to text.

To see detailed implementation of this, open vue-app.js file, find 'all_plugin_details' and under that find 'get_text_from_html'.

IV. get_slot_content

This plugin adds custom function to vue prototype to make function available in all components.

This adds $get_slot_content function which get contents of slot of a given slot name.

To see detailed implementation of this, open vue-app.js file, find 'all_plugin_details' and under that find 'get_text_from_html'.

V. add_and_remove_animation_class

This plugin adds custom function to vue prototype to make function available in all components.

This adds $add_and_remove_animation_class function which adds class and then sets animation-end event to remove class after animation completes.

To see detailed implementation of this, open vue-app.js file, find 'all_plugin_details' and under that find 'add_and_remove_animation_class'.

E. Export Vue Hooks to vue-app Module

It has implementation of exporting events to vue-app module, so that other script can attach callback event to vue hooks.

It exports 2 events created and mounted.

To see detailed implementation of this, open vue-app.js file, find 'all_events'.

5. Main.js

It does the following thing.

i. Initializes all modules implemented in vfs-global.js
ii. Initializes vue application implemented in vue-app.js
iii. Loads one trust content script.
iv. Add callback events in some modules and in vue app.
v. Add window resize event through vfs.details.window_dimension

To see more info on implementation, open main.js and read comments.

Milestones

This page uses gsap library for below purposes

i. To create rotator wheel UI
ii. To create ruler type UI
iii. And For animating both of above

Content for this is written in “milestones-outer-container” component in vue-app.js file (find milestones-outer-container component definition in vue-app.js)

The format of content is written in vue-app.js file. (open vue-app.js file and read comments)

The logic of this component is written inside it’s private component ‘milestones’. (find “milestones: {” inside vue-app.js).

Styles of this component are written inside ‘.milestones_container’ selector block.

Footprint

This page shows detailed data about missions, countries of operation, cities of operation, visa application center addresses, website links, etc.

There are two html pages used for this.

i. **footprint.html** – this is the page which shows up in the url.
ii. **footprint-map.html** – this is the page which is called by iframe in footprint.html page.

So, all the resources like css, js and images are different for footprint and are used in footprint-map.html page. Below are the resources that are used for footprint.

i. inter-ui font
ii. icon font
iii. map.css
iv. jquery plugin
v. map.js (linked map.min.js which is automatically generated for development and build purpose)

Sub heading of Footprint page is given in footprint.html page.

All the headings are given in footprint-map.html page, so to change any heading just open footprint-map.html page and find related heading.

We have stored all the data in footprint-map.html itself. So to change data you just change in html itself. To see why we have given data in html page, click here.

Structure of data given in html

Note: Keep footprint-map.html page open while reading below, it will make you understand better.

i. **Container** - All the data are given in ".map_container" div.
ii. **Countries of operation** - ".map" div inside ".map_container" has all the countries of operation.
iii. **Country of operation Data Container** – inside ".map_container" div there are ".box_content" div with id of format "LightBox(number)" has all the data about country of operation like for particular country of operation what are the missions, what are the cities of operation, what are addresses of VAC and website link.
iv. **Countries of operation Data Link** – Country of operation and it's data is linked through unique lightbox number which are given to '.map > .(country_name) a[onclick]' element and id of ".box_content"
v. **Mission Countries** – inside ".box_content" div there is ".tabs_wrapper > ul.tabs" element which has list of all the mission countries for particular country of operation.
vi. **Mission Country Data Container** – inside ".box_content" div there is ".tab-content-inner > . country_map_container > ul.tab_content_list" has all the data about mission country like what are the cities of operation, what are addresses of VAC and website link.
vii. **Mission Country Data Link** – Mission country and it's data is linked through unique id which are given to ".tabs_wrapper > ul.tabs > li > a[rel]" element and ".tab-content-inner > .country_map_container > ul.tab_content_list > li[id]" element.
viii. **Cities of operation** – inside "ul.tab_content_list > li" element there is "ul.city_list" element which has list of all cities of operation for particular mission country of a particular operation country.
ix. **Cities of operation Data** – each 'li' of "ul.city_list" element has data about VAC address and website link.

All the positions for below type of data on map is set in map.css by adding new class and giving left and top.

i. positions of operation countries on world map
ii. positions of cities of operation on country map

All the images used in this page are in 'assets/images/footprint' folder.
All the below type of images must be of '**.gif**' format.

i. **Country flag** – Each flag image must be of resolution 24 * 16.
ii. **Country map** – Each country map image must be of same color. Each country map image's width must not be greater than 375px. New map can be downloaded from https://www.amcharts.com/svg-maps/ (This will download svg file, convert it to gif in illustrator or online)

By understanding above section you will get an idea of how to add/edit/delete data.

Important Note: While deleting country of operation or mission country, don't just delete it's entry from list also delete it's data container. See below for more info

i. While deleting country of operation delete from ".map" div and also delete corresponding ".box_content" div.
ii. While deleting mission country delete 'li' from "ul.tabs' and also delete 'li' from 'ul.tab_content_list'.

Map.js

This js file doesn't have any data/content. It just has functioning code. (to change in map.js file, see "Development & Build" section.)

Some of most important functions are listed below

I. **generate_all_data**
This function generates json based data from html by using jquery through looping elements.
This data will be used for creating responsive view and also for creating all mission countries.

II. **init_responsive_view**
This function generates the responsive view.

In responsive we have shown editable dropdown. We have copied the required code for editable dropdown from vfs-global.js to map.js and also copied required css of editable dropdown from style.css to map.css.

Since it's an iframe and doesn't use vfs-global.js so not able to use parent window's code as it is written for root frame only.

III. **init_all_visiting_list**
This function generates list of all visiting countries (shown below world map).

IV. **set_iframe_height**
This function sets height of parent iframe element. Whenever body's height changes call this function to change iframe height also.

V. **animate_parent_frame_scroll**
This function is used to animate parent frame's scrollbar. Example of usage is on selecting vising country it scrolls to world map.

Why we have given data in html?

In old website design, footprint was there and given all the data in html itself. We are unable to create new page from scratch because of below reasons

i. There were many errors like different 'country of operation / mission country names in dropdown and map', 'multiple city names for same country', 'different website links in dropdown and map' etc.
Because of these errors, we can't create proper structured data that combines dropdown and map.
ii. There are also many same addresses which are added in different format (e.g. A-101,… and 201-A, …).
Because of this we are unable to create compact data.
iii. Because of different country names we are unable to link flag (all-flags.png) which is used in individuals home page.
iv. We also don't have svg map to make map image sharp. We need svg map image also for creating zoomable map UI.
v. We don't have latitude and longitude based position of each country and cities, also we don't have boundary position of each operation country which we need to place city marker.

Because of above reasons, we have just modified map.css and map.js and used old website's footprint page.