1) Before the actual QC elements lets go over some of the supporting elements that allow the user to functional use the site.

   Mongo API

   Description

   Configuration and status information about each Plants data structure is stored as mongo documents in mongodb. The API allows us to pull information such as:

   - List of Plant names
   - Current snapshot of the MySql db for any Plant(Data Windows)
   - Obtain last modification dates
   - Update the Mongo document when changes have been made to the db.
   - We can also pull these document and display them as JSON
   - The API runs as a service on “us5876webp”
   - The Mongo DB itself runs locally on 902 where the PSAMD app is hosted

   IP21 API

   Description

   Our local MySql db is seeded with about 2 years of data from IP21.

   IP21 API allows us to download in real-time requested data that is not in our local datastore.

   - The API runs as a service on “us5876webp”

   Batch Downloader

   Description

   The Batch Downloader is an R script that runs nightly as a cronjob

   - It runs on 902
   - It waits until after 12am and downloads the previous days data in our local db
   - Currently the Plant and Tags to download are hard coded
   - They should be updated to pull from MongoDB

2) List of Plants to choose from

   Description

   This list is generated by making a call to the Mongo API.

   A call to API path /getPlants will return a list of plant names

3) Data windows of currently selected Plant

   Description

   On initial page load a default Plant will be selected.

   The corresponding Data Snapshot for that Plant will be displayed as JSON

   There is an observer that waits for changes to the list and makes a live call to Mongo API

4) Toggling between Plants in the list should update the displayed Data Window

   Description

   Selecting a different Plant from the list should cause the corresponding JSON to update

   There is an observer that waits for changes to the list and makes a live call to Mongo API

   Our current prod db is seeded with about 2 years of data and gets updated with and additional day every night.

5) History Date Window

   Description

   This is an free text window that displays the latest dates downloaded from IP21 to local MySql database. Although it is a free text window it has most date validations

-

6) Generated Dygraph

   Description

   Submitting valid historical dates will pull data from the system and generate a Dygraph based off the data range of the submitted dates.

   - The Dygraph start and end dates should always match both the submitted dates.
   - The data range should also be either a data window itself in the Json section or covered by an existing window.
   - The Dygraph range window should still be able to smoothly zoom and scroll

7) Metrics

   Description

   5 text boxes below the Dygraph,M1-M5 measure the time of the submission and display.

   - They should always display either zero or a time in ms
   - They should refresh appropriately when on new submission

8) M1 Metric

   Description

   Is the total time it takes the site to run from the time the submission button is pushed to the output of the Dygraph and the updates of M2-M5.

   - M1 should always total M2-M5 (however there is a 5-7ms consistent discrepancy)
   - All these actions are contained in a “observeEvent(input$submitTags,{....})
   - There are many calls made to functions outside of this object this is the router of sorts.
   - This function has an “onQuit()” that can be used to with confidence that the event that was captured and the subsequent code are complete.
   - I use that “onQuit()” to call a method to tally the total time and to remove the message that appears while the code is running.

9) M2 Metric

   Description

   M2 represents the time it takes to pull missing data in the local mysql database from IP21
   When a date range is submitted the system uses the mongo information for the selected Plant to determine if calls need to be made to IP21 to fill any gaps.

   - The IP21 calls are only made to pull data that doesn't already exist in the mysql db.
   - M2 covers the entire pull of IP21 data, including multiple calls to fill gaps.
   - It does not cover saving the data locally.
   - The dates sent to the IP21 API to pull data will be displayed in the On-demand section below the Mysql query.

10) M3 Metric

   Description

   M3 covers the time it take to receive raw text data from IP21.

   - It converts it to a dataframe
   - Sets the column names
   - Convert the string dates to POSIXct dates
   - A few other things to get the data formatted correctly for MySql
   - Then it saves the datafame to the local MySql db

11) M4 Metric

   Description

   Covers the time it takes to pull data locally from MySql

12) M5

Description

M5 covers the time it takes the R code, RenderDygraph method to run

- This is independent of page display time based on HTML and Javascript Rendering

13) Mongo Document and Json Representation

Description

Any time and update (IP21 download) is made to the MySql db, the Mongodb document that represents that Plants database should be updated as well.

- The Mongo API allows updating a Plants Data array (windows) as well as last modified dates.

Windowing

Description

The local MySql is seeded with about 2 years of IP21 data.

That is recorded in the mongo document as on array element with a start, end and modified date.

1) Submitting a time range that is covered by the current data
    - The start and end dates are represented a an array element in the Json
    - Should not alter the json array elements or the modified dates
    - M2 and M3 metrics should have zero values
    - The Dygraph should display the correct start and end dates as well as smooth zooming and scrolling.
2) Submitting a time range that extends an existing window one-day
    - This should make a call to IP21 API with start and end dates for just the extra day
    - The date range sent to IP21 API should be recorded in the On-demand section
    - Should alter the dates for that window in the json appropriately and update the modified dates
    - The M2 and M3 metrics should have non-zero values
    - The Dygraph should display the correct start and end dates as well as smooth zooming and scrolling.
3) Submitting a time range that does not overlap with any data windows represented by the json.
    - This should make a call to IP21 API for the requested start and end dates
    - The date range submitted to the IP21 API should be recorded in the On-demand section
    - The json should be updated with an addition array element that represents the start, end, and modification dates for this IP21 call
    - M2 and M3 metrics should have non-zero values
    - The Dygraph should display the correct start and end dates as well as smooth zooming and scrolling.
4) Submitting a time range that overlaps with one existing window
    - This should make a call for the non-overlapping data in the requested window
    - The existing window in the json array will have it start, end, and modified times updated
    - The date range submitted to the IP21 API should be recorded in the On-demand section
    - M2 and M3 metrics should have non-zero values
    - The Dygraph should display the correct start and end dates as well as smooth zooming and scrolling.

5) Submitting a time range that overlaps multiple windows

- This should cause multiple calls to IP21 API
- To fill the range between two windows
- To fill the gap between the requested start date and the last windows start date
- To fill the gap between the requested end date and the first windows end date
- The date range submitted to the IP21 API should be recorded in the On-demand section
- M2 and M3 metrics should have non-zero values
- The Dygraph should display the correct start and end dates as well as smooth zooming and scrolling.