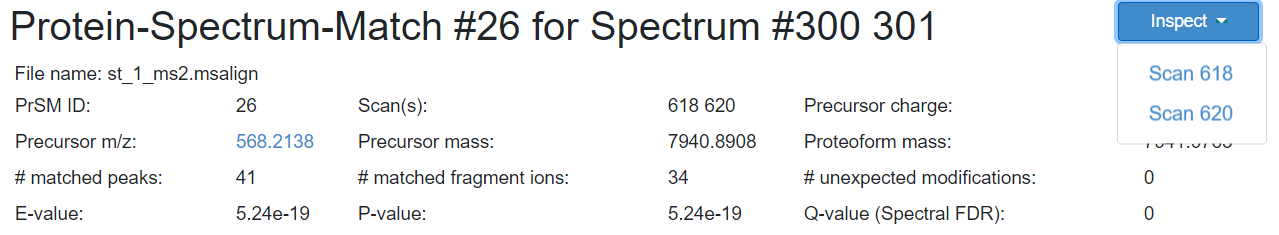
**prsm.html**

When a HTML is being launched in this application, All JavaScript codes inside **$(document).ready(function(){})** wait till all elements of html is loaded on to browser.

Prsm.html Consists of majorly 4 blocks.

* PRSM Information
* SVG of Amino Acid Sequence
* Spectrum and MonoMass Graphs
* MonoMass Table

**1.PRSM Information:**



* First line in the above image consists of prsm Id number and the spectrum numbers corresponding to the prsm. This data is retrieved from the prsm data file directly
* Remaining data is a tabular data retrieved directly from the prsm data file
* Inspect Element Button is used to navigate to inspect page for corresponding spectrum. Please find the detailed flow of Inspect buttons below

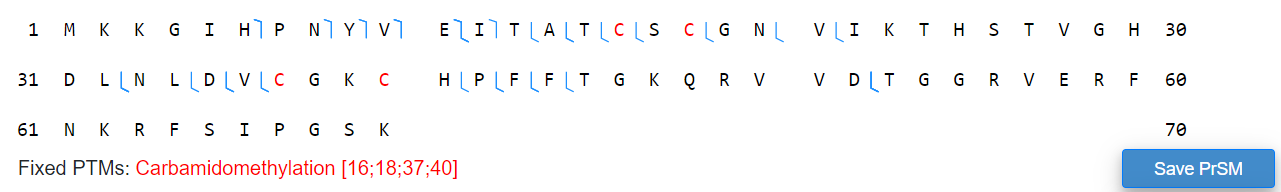
**Code Flow:**

$(document).ready 🡪 BuildUrl() (prsmtohtml.js)

🡪loadDatafromJson2Html() (prsmtohtml.js)

The Above flow and functions retrieve all the information of the PRSM information block

**2.SVG of Amino Acid Sequence:**



This block consists of three major information.

* One to draw an SVG as represented above
* Save prsm button to resize and save the SVG
* show all the fixed and variable PTMs under the SVG

**2.1 Drawing SVG:** The above SVG consists of Amino Acids separated by specific width and the annotations (in blue color) are matched positions. The Margin width, space between acids, block length, row length etc.,. are fixed information retrieved from **parameters()** function in **coordinates\_util.js** file

**Code Flow:**

$(document).ready 🡪 buildSvg() (drawSvg.js) 🡪getSvgSize() (drawSvg.js) 🡪calibrateCoordinates (coordinates\_util.js)

🡪massShiftBackgroundColor() (drawSvg.js)

🡪skippedAcidNotification() (drawSvg.js)

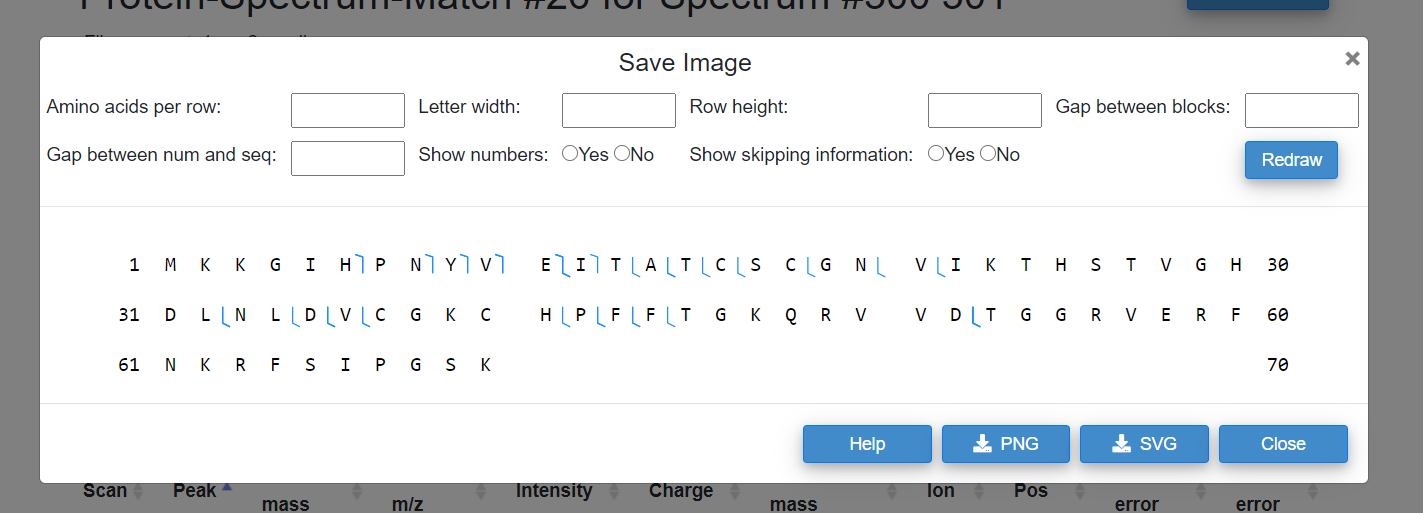
🡪getNumValues() (drawSvg.js)

🡪drawAnnoOfStartEndPosition() (drawSvg.js)

🡪addColorToFixedPtms() (drawSvg.js)

🡪buttonsAndAlerts() (drawSvg.js)

**2.2 Save Prsm Button:** On Click of the save Prsm button, new window pops up and allows the user to enter the size and width parameters. On click of resize, executes all the above **Drawing SVG** code flow and on click of the download buttons on the pop-up window, allows us to download the SVG.



**Code Flow:**

On Click of Resize button 🡪 buttonsAndAlerts() (prsmtohtml.js) 🡪 d3.select('#resize').on("click", function(){})

Inside the on click function, it gets data from the input fields and executes all the functions from above **Drawing SVG** code flow.

**2.2.1 Download as SVG/PNG:** On Click of the download buttons (PNG/SVG) from the above image. It invokes on click action function and pops up a small window to provide name for the image and on click of save, saves the image.

**Code Flow:**

On Click of download SVG/PNG 🡪 buttonsAndAlerts() (prsmtohtml.js) 🡪 d3.select('#saveImage').on("click",function(){}) (prsmtohtml.js) 🡪 popupnamewindow() (prsmtohtml.js) 🡪 $("#saveimage").click(function(){}) (saveImage.js)

The above flow is a single flow where all the functions are called in its predecessor.

**2.3 Fixed/Variable PTMs:**

Fixed or Variable PTMs are the data retrieved directly from prsm data file. Data is been retrieved and placed in HTML using occurence\_ptm() from prsmtohtml.js.

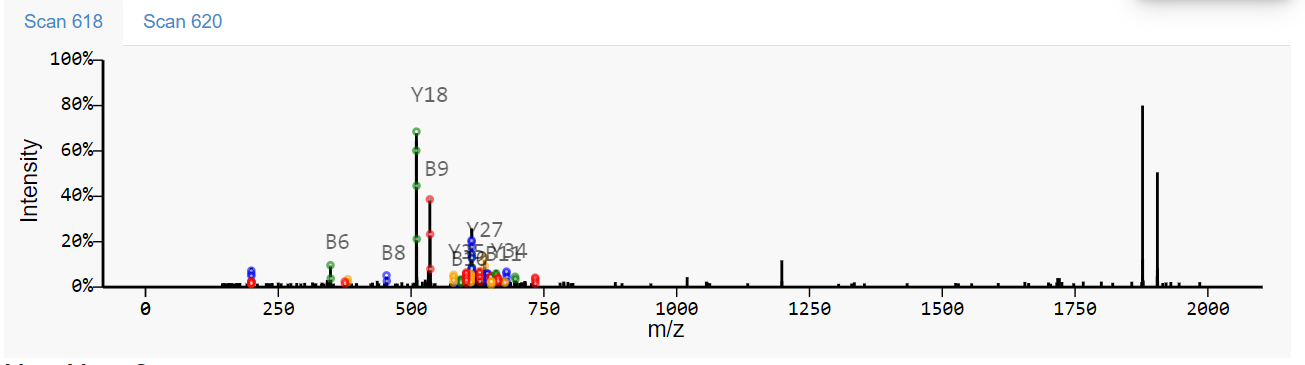
**Code Flow:**

$(document).ready(function(){}) 🡪 occurence\_ptm(prsm) (prsmtohtml.js)

**3.Spectrum and MonoMass Graphs:**

This block consists of 2 graphs. One is a spectrum graph for which the data is been retrieved from corresponding spectrum data and mono m/z is used. Second graph is a Mono Mass graph where mono mass is used, and data is retrieved from prsm file. Along with the mono mass graph, error plot is also been drawn using the error data from prsm file.

**3.1 Spectrum Graph:**



As seen in the above image, if a prsm consists of multiple scan numbers, we have multiple spectrums. However, the code flow remains same to draw a spectrum graph.

**Code Flow:** Html Id for the spectrum graph starts with ms2svg\_ followed by scan number.

$(document).ready 🡪getUniqueScanIdList() (multiscan.js) 🡪 getMonoMassDataList() (multiscan.js) 🡪 promiseLoadDataJS()(multiscan.js)

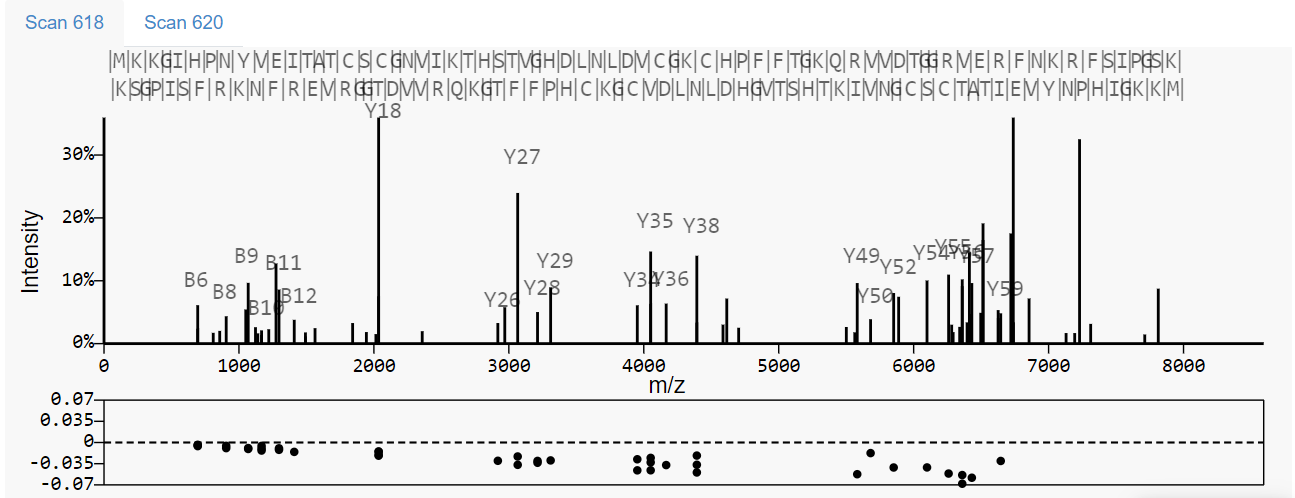
promiseLoadDataJS() (multiscan.js)🡪 getCurrentData() (multiscan.js)

🡪createMs2NavEements() (multiscan.js)

🡪createMultipleSvgs() (generateSpectrumGraphs.js) 🡪generateCorrespondingGraph()

🡪addSpectrum() (./js/spectrum\_graph->invokespectrum.js)

**3.2 Mono Mass Graph:**



This follows the same flow as spectrum graph except the data being retrieved is from different file and contains more data on the graph.

**Code Flow:**

$(document).ready 🡪getUniqueScanIdList() (multiscan.js) 🡪 getMonoMassDataList() (multiscan.js) 🡪 promiseLoadDataJS()(multiscan.js)

promiseLoadDataJS() (multiscan.js)🡪 getCurrentData() (multiscan.js)

🡪createMs2NavEements() (multiscan.js)

🡪createMultipleSvgs() (generateSpectrumGraphs.js) 🡪generateCorrespondingGraph()

If( startOfId == "monoMassSvg")

generateCorrespondingGraph() 🡪 getIonTypeMass() (calculateprefixandsuffixmass.js)

🡪getSequence() (calculateprefixandsuffixmass.js)

🡪 getUnknownMassList() (calculateprefixandsuffixmass.js)

🡪 getPrefixMassList() (calculateprefixandsuffixmass.js)

🡪 getIonTypeMass() (calculateprefixandsuffixmass.js)

🡪 getSuffixMassList() (calculateprefixandsuffixmass.js)

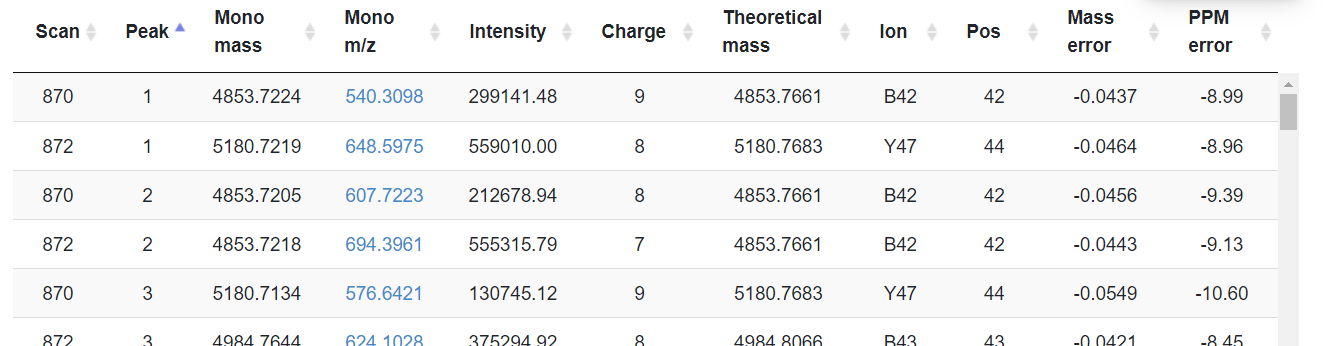
🡪 json2ErrorDataList() (prsmDataUtil.js)

🡪 getAbsoluteMaxValfromList() (generateSpectrumGraphs.js)

🡪 addSpectrum() (./js/spectrum\_graph->invokespectrum.js)

Put all the data into Graph features object and pass it to invokespectrum

**4.MonoMass Table:**



This block consists of a tabular form with information from the prsm data file. The tabular form consists of information of Mono Mass, Mono m/z, Intensity, Charge, Theoretical Mass, Ion, Pos, Mass error, PPM error for each Scan id.

On click of Mono m/z value the spectrum and mono mass graphs zooms to the mono m/z and mono mass locations, respectively.

**Code flow:**

$(document).ready 🡪 createTableElements() (prsmtohtml.js)

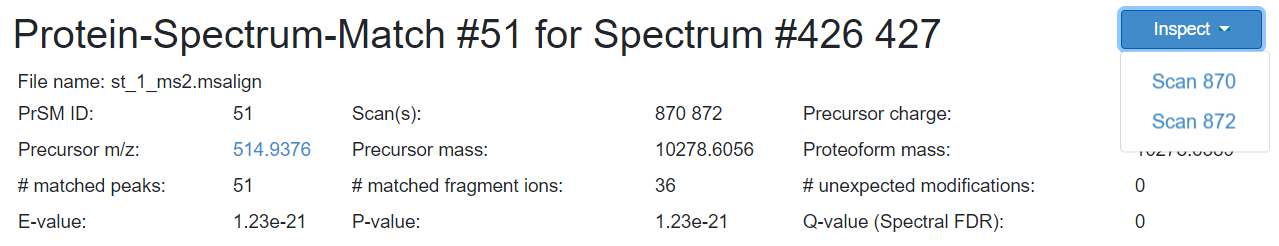
When the table is being created, the mono m/z column is a hyperlink column with a class name **peakRows.** Based on the class name **peakRows**, on click actions of the mono m/z is determined.

The on click action code is defined in

graphOnClickActions() (generateSpectrumGraphs.js) 🡪$(".peakRows").click(function() {} 🡪 generateCorrespondingGraph()

Inside the on click function takes the value of the **mono m/z** and pass it to the generateCorrespondingGraph() as one of the input parameters. The graphs are generated again by taking the **mono m/z** values as the center point.

**On Click of Inspect Button:**



On click of inspect button, provides a dropdown with all available scan numbers. On click of the scan number, opens a new page called Visual Inspection.

Before opening Visual inspection, the prsm page stores all the information of the current prsm needed for inspection.

**Code Flow:**

$(document).ready 🡪setDropDownItemsForInspectButton() (topview.js)

🡪onClickToInspect() (topview.js) 🡪 onclickTopView() (topview.js) 🡪 getDataFromPRSMtoSpectralView()

🡪 getMassAndIntensityData()

🡪 getSequence()

🡪 getFixedPTMMassList()

🡪 getUnknownMassList()

Stores all the data using **window.localStorage**