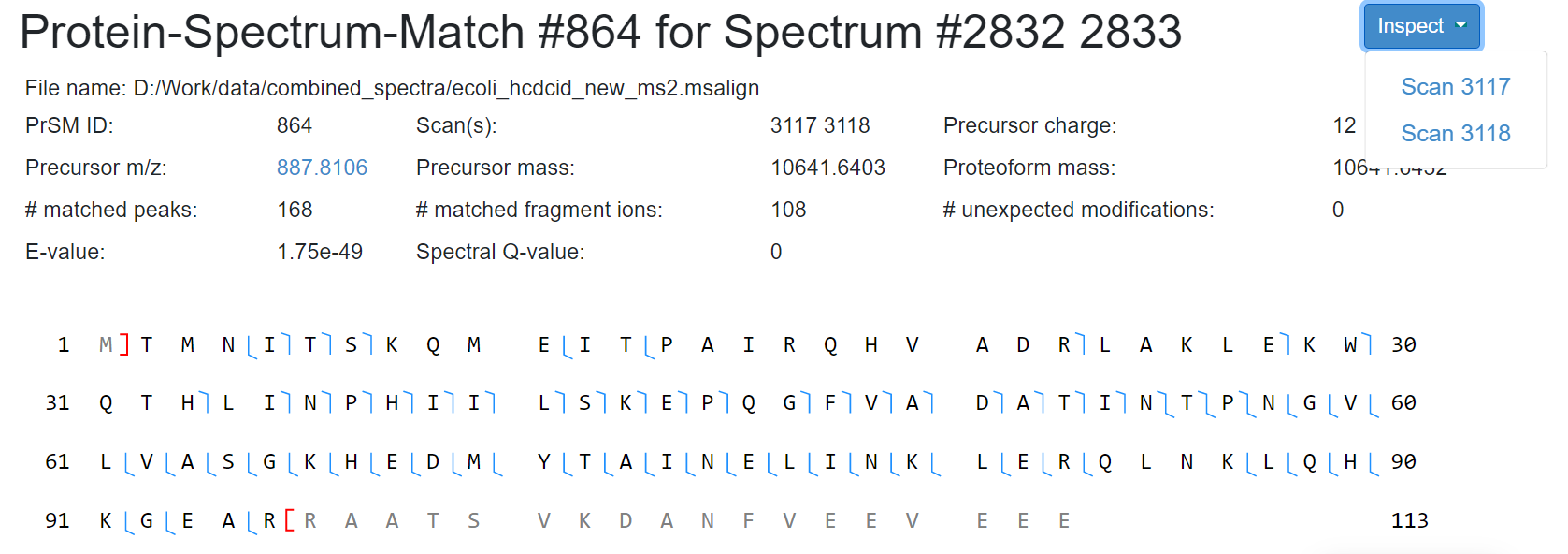
**prsm.html**

When prsm.html page is being launched in this application, all JavaScript codes inside **$(document).ready(function(){})** wait until all elements of the html page is loaded on to browser.

Prsm.html mainly consists of 4 blocks.

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

1. **PRSM Information:**



* First line in the image consists of prsm id number and the spectrum ids 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 button is used to navigate to the inspect page for corresponding spectrum. Please find the detailed flow of Inspect button below.

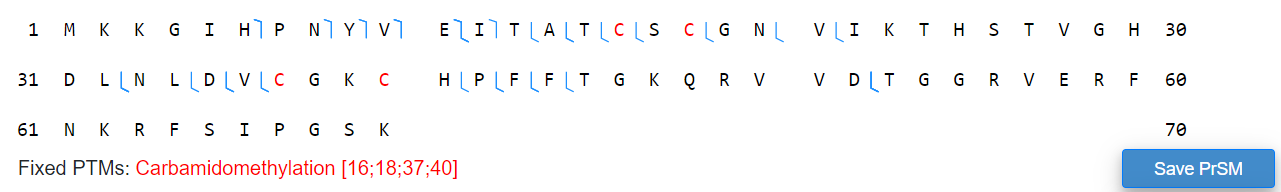
**Code Flow:**

$(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 PTM, variable PTM, unknown PTMs under the SVG

**2.1 Drawing SVG:** The above SVG consists of amino acids separated by same interval and the annotations (in blue color) which represent matched positions. The margin width, space between acids, block length, row length etc.,. are retrieved from PrsmPara class member variables.

**Code Flow:**

$(document).ready 🡪 json2Ptms(prsm) (prsm\_to\_html.js)

🡪 json2MassShifts(prsm) (prsm\_to\_html.js)

🡪getAminoAcidSequence(0,prot.annotation.residue.length - 1,prot.annotation.residue) (prsm\_to\_html.js)

🡪json2BreakPoints(prsm, parseInt(prot.annotation.first\_residue\_position)) (prsm\_to\_html.js)

🡪new Proteoform(prot.proteoform\_id, prot.sequence\_name, sequence,

prot.annotation.first\_residue\_position, prot.annotation.last\_residue\_position, prot.proteoform\_mass,

massShifts, fixedPtms, protVarPtms, variablePtms) (proteoform.js)

🡪new Prsm(prsm.prsm\_id, proteoformObj, "", "", breakPoints, prsm.matched\_peak\_number,

prsm.matched\_fragment\_number, prsm.annotated\_protein.unexpected\_shift\_number,

prsm.e\_value, prsm.fdr) (prsm.js)

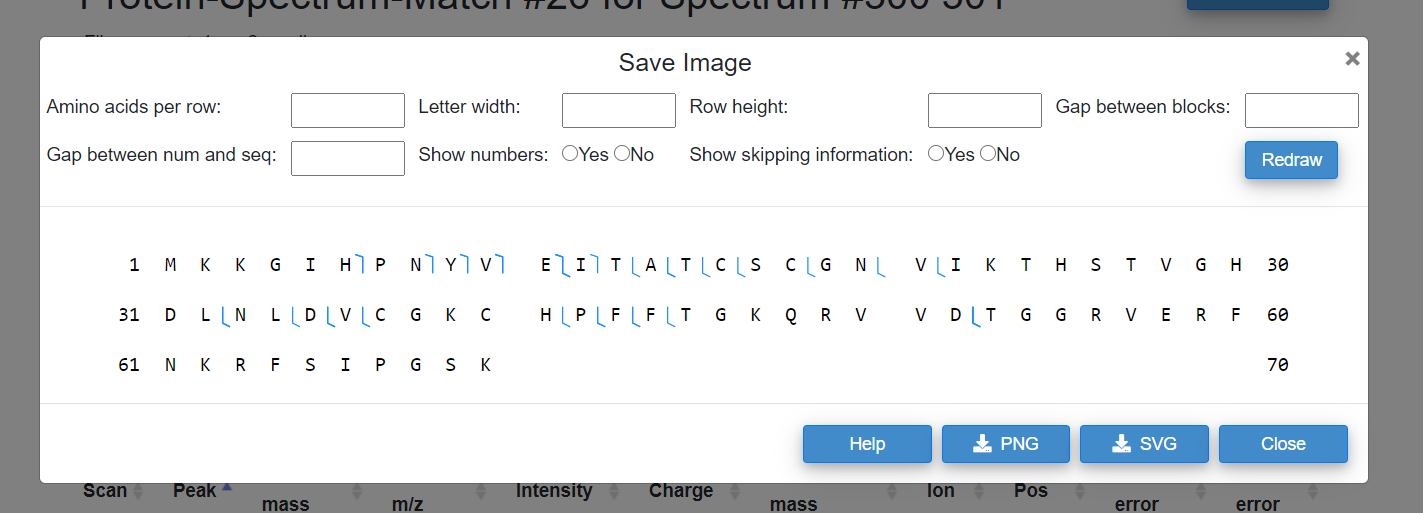
🡪new PrsmGraph(prsmSvgId, prsmObj) (prsm\_graph.js)

🡪prsmGraph.redraw() (prsm\_graph.js)

🡪occurence\_ptm(prsm\_data.prsm) (prsm\_to\_html.js)

🡪getUnknownPtms(prsm\_data.prsm) (prsm\_to\_html.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:**

new SavePrsm(prsmGraph) (save\_prsm.js) 🡪savePrsmObj.main() (save\_prsm.js) 🡪 initPrsmModalEventHandler() (save\_prsm.js) 🡪 d3.select('#prsm\_graph\_redraw\_btn').on("click", (), function(){})

Inside the on click function, it gets data from the input fields and redraws the graph using redraw method in SavePrsm class.

**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:**

new SavePrsm(prsmGraph) (save\_prsm.js) 🡪savePrsmObj.main() (save\_prsm.js) 🡪 initPrsmModalEventHandler() (save\_prsm.js) 🡪 d3.select('#prsm\_popup\_svg\_btn OR #prsm\_popup\_png\_btn).on("click",function(){}) (save\_prsm.js) 🡪 popupnamewindow() (util.js) 🡪 $("#saveimage").click(function(){}) (util.js)

**2.3 Fixed/Variable PTMs:**

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

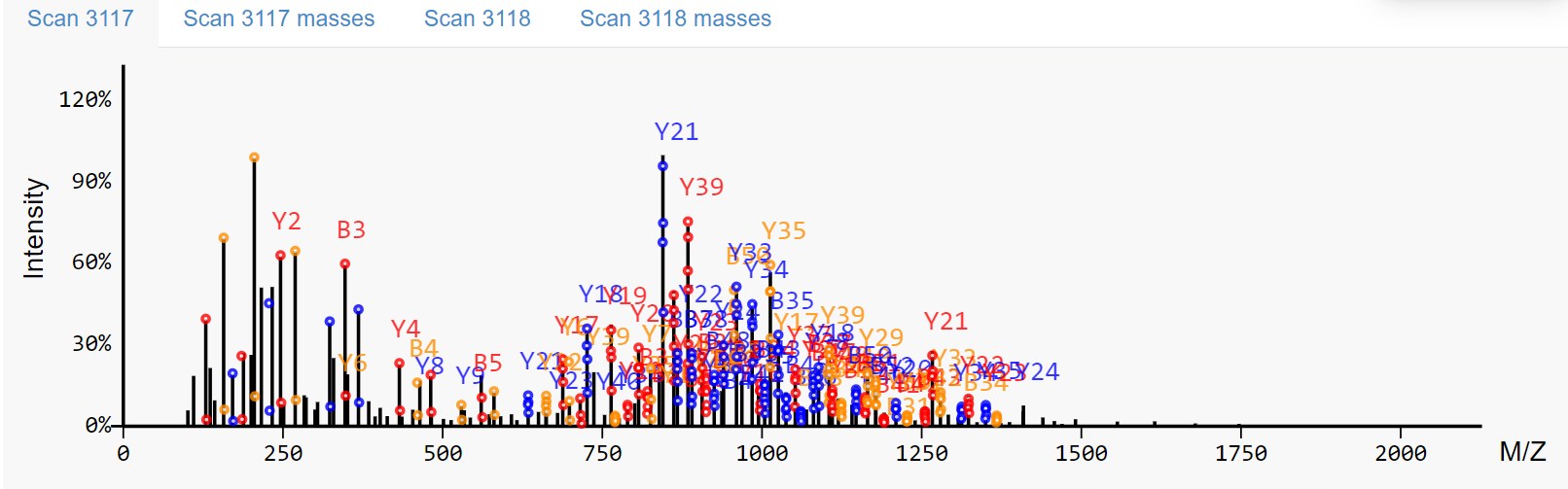
**Code Flow:**

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

**3.Spectrum and MonoMass Graphs:**

This block consists of 2 graphs. One is a spectrum graph for which the data is 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 mass error.

**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 drawing a spectrum graph.

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

$(document).ready 🡪 loadMsTwo (ms2SpecIdList, ms2FileList, prsmGraph.data.proteoform,

"ms2\_svg\_div", "ms2\_graph\_nav") (prsm.html)

🡪createMs2NavElement(j, divId, navId, specList[j].scan) (load\_spectra.js)

🡪createSvg(show, divId, svgId, "ms2\_svg\_graph\_class") (load\_spectra.js)

🡪new Peak(k, peak.mz, peak.intensity) (peak.js) //for each peak

🡪new Envelope(env.mono\_mass, env.charge) (envelope.js) //for each envelope

🡪new Peak(k, peak.mz, peak.intensity) (peak.js) //for each theoretical peak in each envelope

🡪getIons(specId, deconvPeaks, envelopes) (load\_spectra.js)

🡪new SpectrumGraph(svgId,peaks) (spectrum\_graph.js)

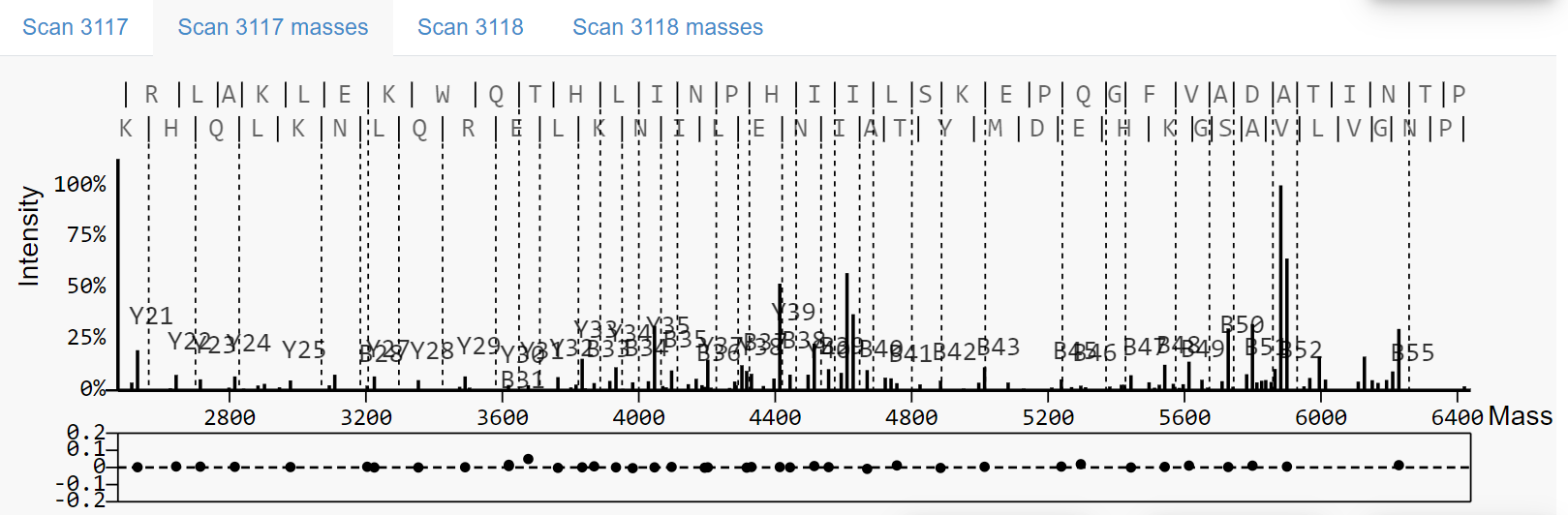
🡪SpectrumGraph.addRawSpectrumAnno(envelopes,ions) (spectrum\_graph.js)

🡪SpectrumData.assignLevelPeaks(peaks) (spectrum\_data.js)

🡪SpectrumData.assignLevelEnvs(envelopes) (spectrum\_data.js)

🡪SpectrumGraph.redraw() (spectrum\_graph.js)

**3.2 Mono Mass Graph:**



This graph is drawn after the spectrum graph. The code flow starts right after the code flow for the spectrum graph.

**Code Flow:**

🡪createSvg(show, divId, monoSvgId, "ms2\_svg\_graph\_class") (load\_spectra.js)

🡪getMonoMasses(deconvPeaks) (load\_spectra.js)

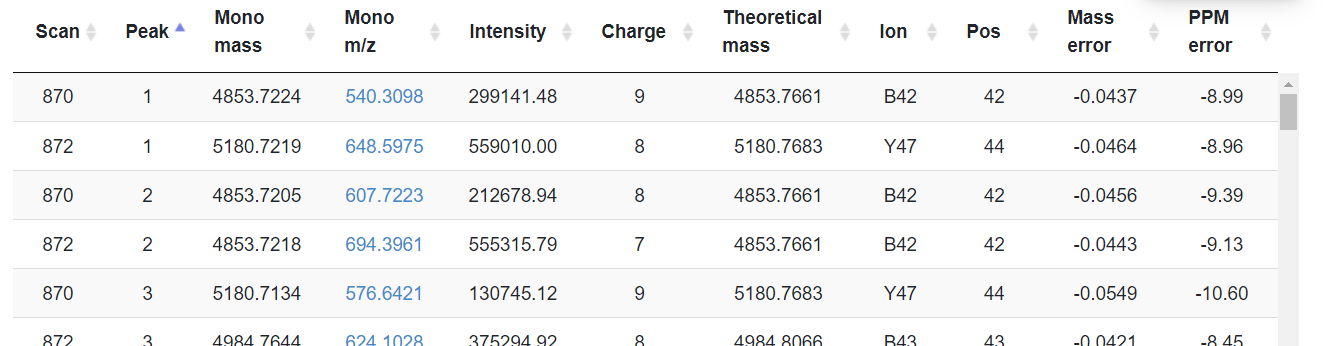
🡪assignLevelPeaks(monoMasses) (spectrum\_data.js)  
🡪new SpectrumGraph(monoSvgId,monoMasses, proteoform.getSeq().length) (spectrum\_graph.js)

🡪addMonoMassSpectrumAnno(monoIons,proteoform, nIonType, cIonType) (spectrum\_graph.js)

🡪setMonoMassGraph(true) (spectrum\_graph.js)

🡪SpectrumGraph.redraw() (spectrum\_graph.js)

**4.MonoMass Table:**



This block consists of a tabular form with information from the prsm data file. The 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)

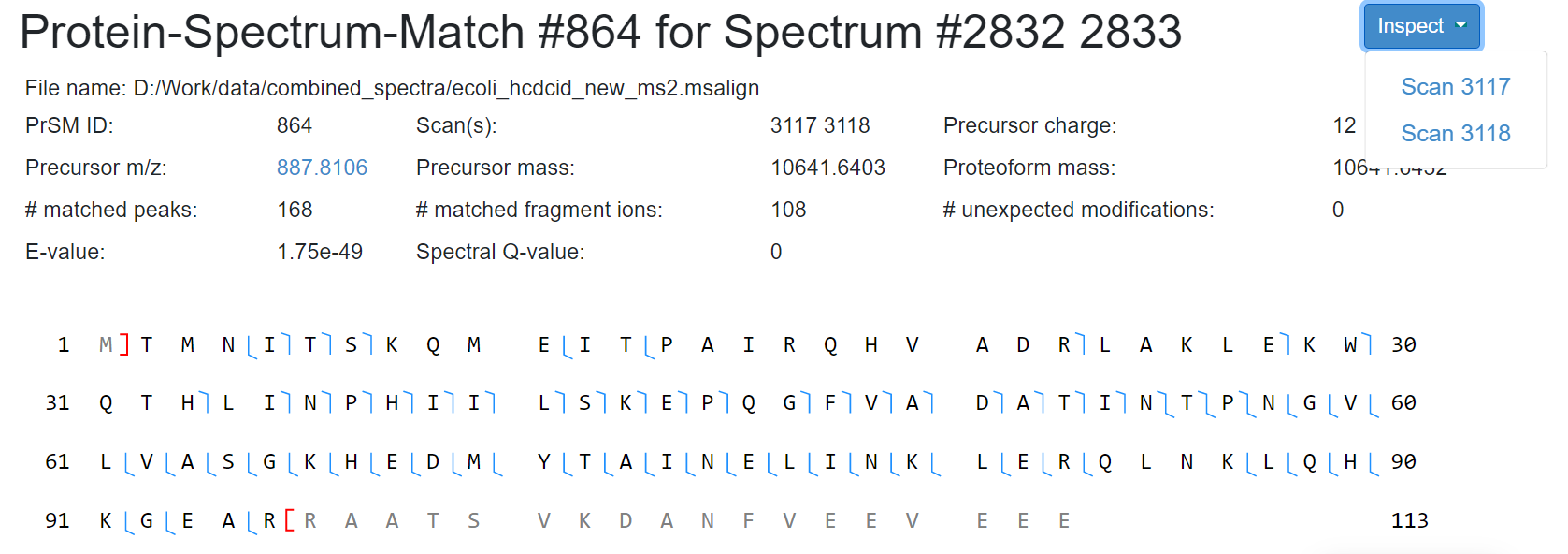
When the table is being created, the mono m/z column is a hyperlink column with a class name **peakRows.** Based on the class name **peakRows**, on click actions of the mono m/z is determined.

The on click action code is defined in

addButtonActions () (button\_actions.js) 🡪$(".row\_mono\_mz").click(function() {} 🡪 showMs2Graph () (button\_actions.js)

Inside the on click function takes the value of the **mono m/z** and calls showMs2Graph(), which will redraw the spectrum graph. 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(scanIds,specIds) (prsm.html)

🡪onClickToInspect() (prsm.html) 🡪 onclickTopView(e) (inspect.js)

🡪 getDataFromPRSMtoSpectralView(ms2\_data) (inspect.js)

🡪 getMassAndIntensityData() (inspect.js)

🡪 getSequence() (inspect.js)

🡪 getFixedPtm () (inspect.js)

🡪 getMassShift () (inspect.js)

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

**End of Document**