What this guide covers

This guide provides an introduction to the basic coding structures used to develop the ‘Mathing’ application. Once read, a developer will have a basic understanding of how workspaces and their related functions, help users create sophisticated forms of mathematics. This guide is by no means a complete reference, it is designed to help give developers a quick start to contributing. Also, each .js file includes comments to describe their main functions and identifiers throughout the application.

Who this manual is for

This manual is for any web developer with a background in JavaScript, HTML and CSS, wishing to contribute to the Mathing application.

Technology

LaTeX, which is a variant of **TeX** is a document markup language with robust math markup commands. It became the industry standard for typesetting of mathematics, and is one of the most common formats for mathematical journals, articles, and books.
MathJax is an open-source JavaScript display engine for LaTeX. Mathematics created in LaTeX will be processed and typeset using JavaScript to produce HTML equations for viewing in any modern browser.
Please see the following URL for an in depth explanation of **MathJax**:
http://docs.mathjax.org/en/latest/start.html
JavaScript: Please see the following URL for an in depth explanation
http://www.w3schools.com/js/
HTML: Please see the following URL for an in depth explanation
http://www.w3schools.com/html/default.asp
CSS: Please see the following URL for an in depth explanation
http://www.w3schools.com/css/default.asp

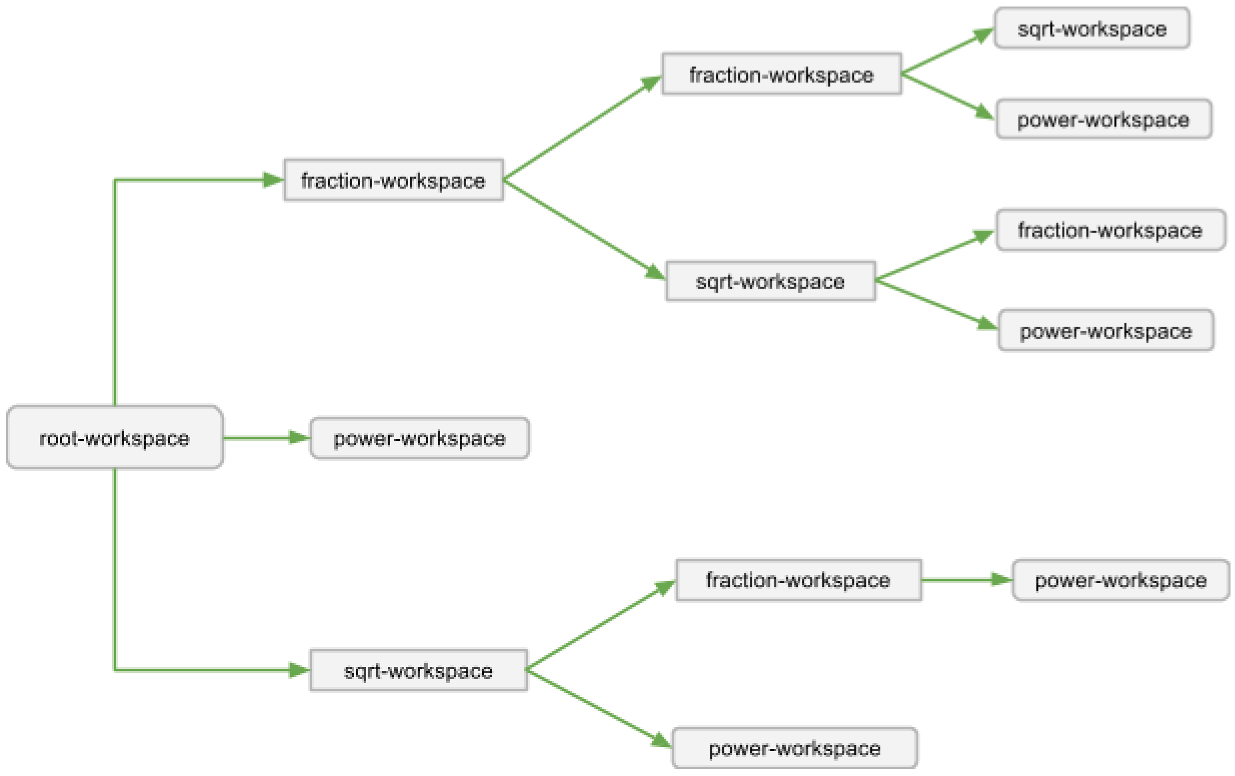
Program Workflow:

Before you begin, please take a moment and open the Mathing application www.mathing.ca and work through some of the exercises. This may help in understanding how complex mathematical forms are structured using individual items such as a fraction, a square root, or a power, built in their own workspace. The term ‘workspace’ is used throughout this document to differentiate between individual items created in their own workspace and then added to a parent workspace. The more time spent working through the exercises will help in understanding this workflow.

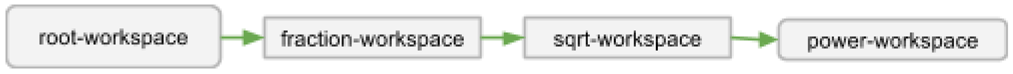
Accessing Child Workspaces:

To enhance your knowledge of how workspaces are functionally related please review the following figures and the example workflow provided.

Figure 1.2 Demonstrates how child workspaces are accessed and their role in the program workflow.



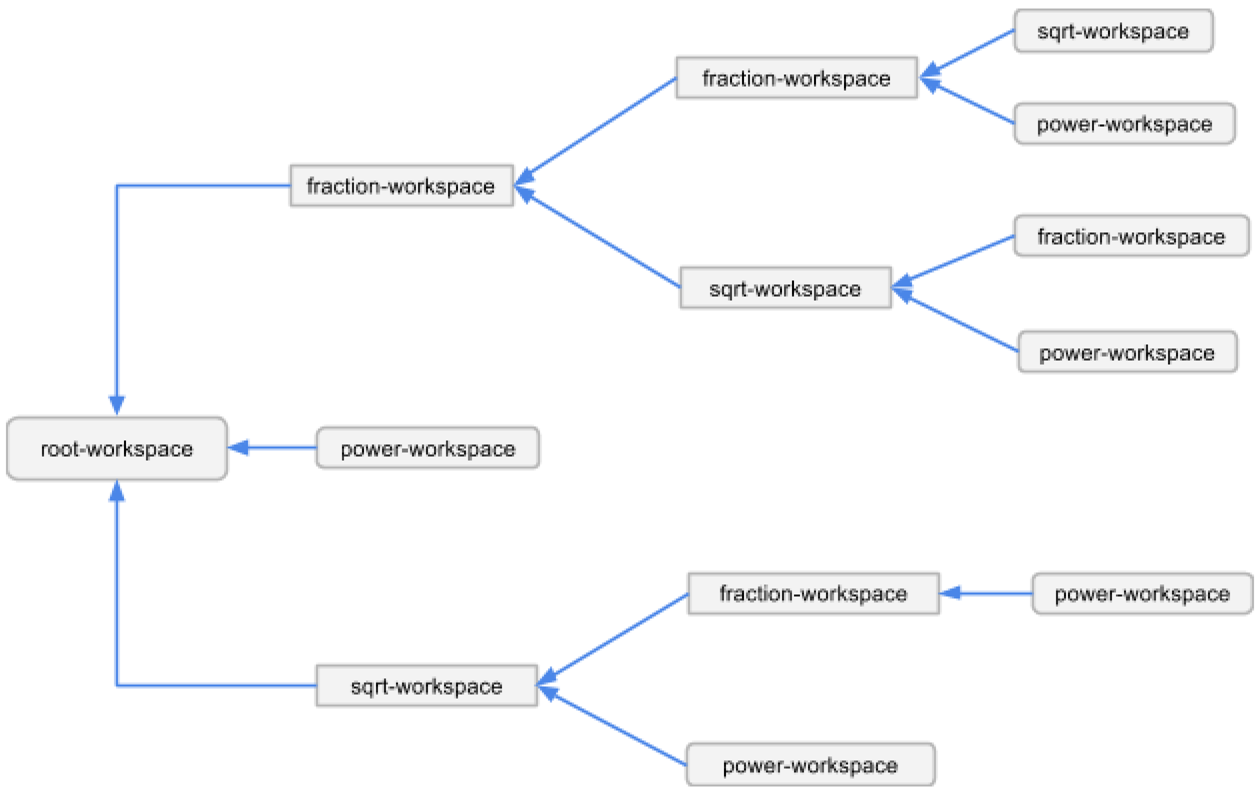
Example workflow:



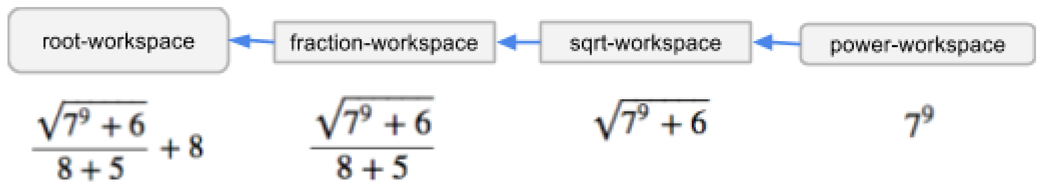
Starting from the root-workspace open it's child fraction-workspace to add a fraction.
Working from the fraction-workspace open it's child sqrt-workspace to add a square root.
Working from the sqrt-workspace open it's child power-workspace to add a power.

Saving work to parent workspace:

Figure 1.3 Demonstrates how work created in each workspace is saved and added to its parent workspace.



Example workflow:



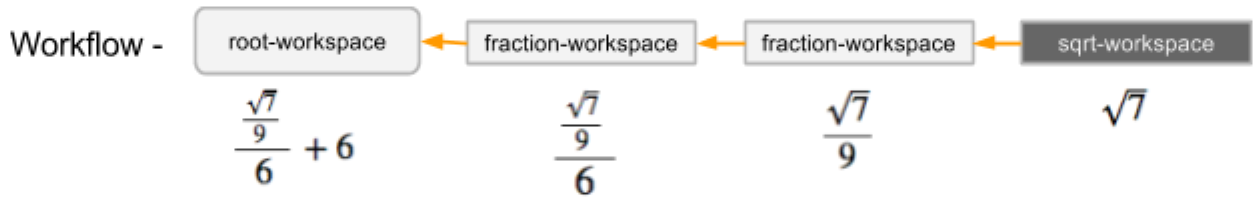
$$\frac{\sqrt{7^9+6}}{8+5}+8 \qquad \frac{\sqrt{7^9+6}}{8+5} \qquad \sqrt{7^9+6} \qquad 7^9$$

To expand on the example workflow above, work created in the child power-workspace is added to its parent sqrt-workspace. The resulting content is added to its parent fraction-workspace. The resulting content is added to its parent root-workspace:

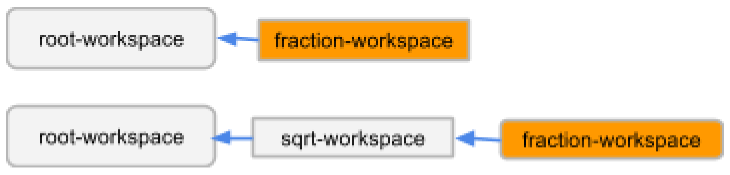
Coding Structures

The remainder of this guide will give a brief explanation of the coding structures used to develop each workspace. Also to be explicit and understand workflow, identifiers have been used for variables and functions that best describe their purpose. E.g. The function fractionSquareRootBuilder() and variable getFractionSquareRootItem indicate that we are working on adding a square root to a fraction, and once validated this item is added to the root workspace. We read these identifiers from right to left.

Lastly, for each workspace a workflow diagram is shown to illustrate its relationship to its parent workspace. E.g.



NOTE: You may notice that workspaces can have the same name, however, they exist in a different context/workflow as follows:



Root Workspace

Parent to all other workspaces. Only simple objects such as constants, variables, basic arithmetic operators and parentheses are added directly to this workspace. More complex objects such as Fractions, Square Roots and Powers are added to this workspace from within their own context/workspace.

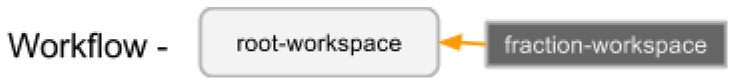
Javascript file: mainPanel.js

Root workspace is displayed when opening the index page of the mathing application. Values entered from this workspace are passed to the function equationBuilder() and appended to global var **concatEquationItems.**

concatEquationItems is passed to the MathJax function mainUpdateMathjax(*concatEquationItems*) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " MathOutput".

The user can ‘Undo’, ‘Reset’ or ‘Cancel’ work done in the current workspace

Fraction Workspace



Javascript file: fractionBuilder.js

Display workspace using HTML element with ID attribute "fractionPanel" via showfractionBuilder(). Values entered from this workspace are passed to fractionBuilder() and appended to global var **getFractionItem.**

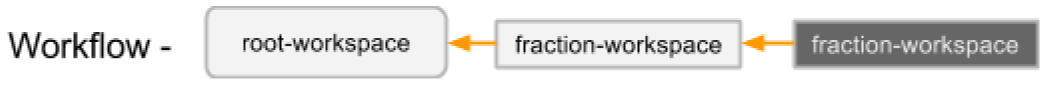
getFractionItem is then passed to the MathJax function F_UpdateMathjax(getFractionItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionOutput".

Also from this workspace getFractionItem is temporarily passed and typeset to the root workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can ‘Reset’ or ‘Cancel’ work done in the current workspace or ‘Insert’ to validate work added to the root workspace.

NOTE: In this context all items created in fraction workspace are added to root workspace once validated

Fraction Workspace



Javascript file: fractionfractionBuilder.js

Display workspace using HTML element with ID attribute "fractionFractionPanel" via showfractionFractionBuilder().

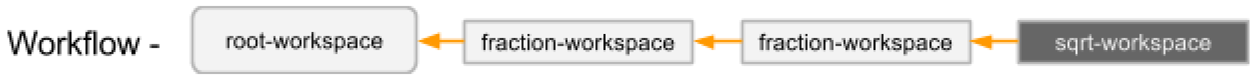
Values entered from this workspace are passed to fractionFractionBuilder() and appended to global var **getFractionFractionItem.**
getFractionFractionItem is then passed to the MathJax function FF_UpdateMathjax(getFractionFractionItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionFractionOutput".

Also from this workspace getFractionFractionItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can ‘Reset’ or ‘Cancel’ work done in the current workspace or ‘Insert’ to validate work added to the fraction workspace[1].

NOTE: In this context all items created in the fraction workspace are added to the parent fraction workspace once validated.

Sqrt Workspace



Javascript file: fractionfractionSquareRootBuilder.js

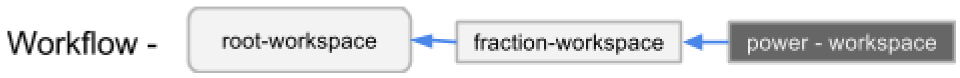
Display workspace using HTML element with ID attribute "fractionFractionSquareRootPanel" via showfractionFractionSquareRootBuilder().
Values entered from this workspace are passed to fractionFractionSquareRootBuilder() and appended to global var **getFractionFractionSquareRootItem**.
getFractionFractionSquareRootItem is then passed to the MathJax function FFS_UpdateMathjax(getFractionFractionSquareRootItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionFractionSquareRootOutput".

Also from this workspace getFractionFractionSquareRootItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can ‘Reset’ or ‘Cancel’ work done in the current workspace or ‘Insert’ to validate work added to Fraction Workspace.

NOTE: In this context all items created in the sqrt workspace are added to the parent fraction workspace once validated.

Power Workspace



Javascript file: fractionExponentBuilder.js

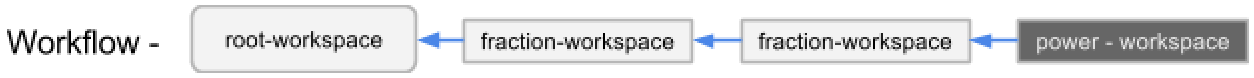
Display workspace using HTML element with ID attribute "fractionExponentPanel" via showFractionExponentBuilder().
Values entered from this workspace are passed to fractionExponentBuilder() and appended to global var **getFractionPowerItem.**
getFractionPowerItem is then passed to the MathJax function FE_UpdateMathjax(getFractionPowerItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionExponentOutput".

Also from this workspace getFractionPowerItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to Fraction Workspace[2].

NOTE: In this context all items created in the power workspace are added to the parent fraction workspace once validated.

Power Workspace



Javascript file: fractionFractionExponentBuilder.js

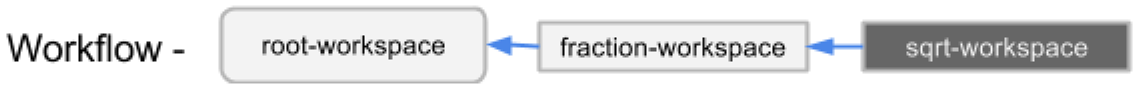
Display workspace using HTML element with ID attribute "fractionFractionExponentPanel" via showfractionFractionExponentBuilder().
Values entered from this workspace are passed to fractionFractionExponentBuilder() and appended to global var **getFractionFractionPowerItem.**
getFractionFractionPowerItem is then passed to the MathJax function FFE_UpdateMathjax(getFractionFractionPowerItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionFractionExponentOutput".

Also from this workspace getFractionFractionPowerItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to Fraction Workspace.

NOTE: In this context all items created in the power workspace are added to the parent fraction workspace once validated.

Sqrt Workspace



Javascript file: fractionSquareRootBuilder.js

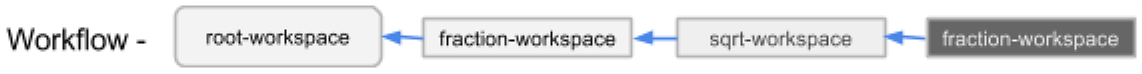
Display workspace using HTML element with ID attribute "fractionSquareRootPanel" via showFractionSquareRootBuilder().
Values entered from this workspace are passed to fractionSquareRootBuilder() and appended to global var getFractionSquareRootItem.
getFractionSquareRootItem is then passed to the MathJax function FS_UpdateMathjax(getFractionSquareRootItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionSquareRootOutput".

Also from this workspace getFractionSquareRootItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to Fraction Workspace[1].

NOTE: In this context all items created in the sqrt workspace are added to the parent fraction workspace once validated.

Fraction Workspace



Javascript file: fractionSquareRootFractionBuilder.js

Display workspace using HTML element with ID attribute "fractionSquareRootFractionPanel" via showFractionSquareRootFractionBuilder().

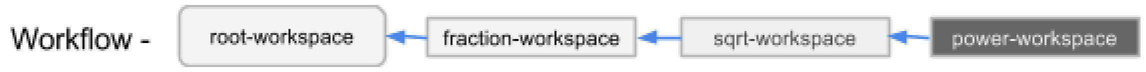
Values entered from this workspace are passed to fractionSquareRootFractionBuilder() and appended to global var getFractionSquareRootFractionItem.
getFractionFractionSquareRootItem is then passed to the MathJax function FSF_UpdateMathjax(getFractionSquareRootFractionItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionSquareRootFractionOutput".

Also from this workspace getFractionFractionSquareRootItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to sqrt Workspace.

NOTE: In this context all items created in the fraction workspace are added to the parent sqrt workspace once validated.

Power Workspace



Javascript file: fractionSquareRootExponentBuilder.js

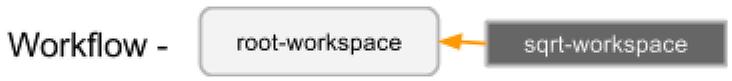
Display workspace using HTML element with ID attribute "fractionSquareRootExponentPanel" via showFractionSquareRootExponentBuilder().
Values entered from this workspace are passed to fractionSquareRootExponentBuilder() and appended to global var getFractionSquareRootExponentItem.
getFractionSquareRootExponentItem is then passed to the MathJax function FSE_UpdateMathjax(getFractionSquareRootExponentItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " fractionSquareRootExponentOutput".

Also from this workspace getFractionSquareRootExponentItem is appended to getFractionItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getFractionItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to sqrt Workspace.

NOTE: In this context all items created in the power workspace are added to the parent sqrt workspace once validated.

Sqrt Workspace



Javascript file: squareRootBuilder.js

Display workspace using HTML element with ID attribute "squareRootPanel" via showSquareRootBuilder().
Values entered from this workspace are passed to squareRootBuilder() and appended to global var **getSquareRootItem.**

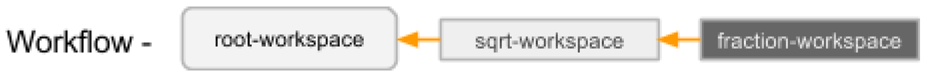
getSquareRootItem is then passed to the MathJax function S_UpdateMathjax(getSquareRootItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " squareRootOutput".

Also from this workspace getSquareRootItem is temporarily passed and typeset to the root workspace via mainUpdateMathjax(concatEquationItems + getSquareRootItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to root Workspace.

NOTE: In this context all items created in sqrt workspace are added to the parent root workspace once validated

Fraction Workspace



Javascript file: squareRootFractionBuilder.js

Display workspace using HTML element with ID attribute "squareRootFractionPanel" via showSquareRootFractionBuilder().
Values entered from this workspace are passed to squareRootFractionBuilder() and appended to global var **getSquareRootFractionItem.**

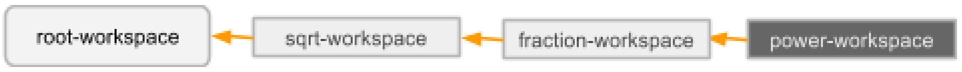
getSquareRootFractionItem is then passed to the MathJax function SF_UpdateMathjax(getSquareRootFractionItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " squareRootFractionOutput".

Also from this workspace getSquareRootFractionItem is appended to getSquareRootItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getSquareRootItem) for viewing prior to validation.

The user can ‘Reset’ or ‘Cancel’ work done in the current workspace or ‘Insert’ to validate work added to sqrt Workspace

NOTE: In this context all items created in the fraction workspace are added to the parent sqrt workspace once validated.

Power Workspace

Workflow - 

Javascript file: squareRootFractionExponentBuilder.js

Display workspace using HTML element with ID attribute "squareRootFractionExponentPanel" via showSquareRootFractionExponentBuilder().
Values entered from this workspace are passed to squareRootFractionExponentBuilder() and appended to global var **getSquareRootFractionExponentItem.**

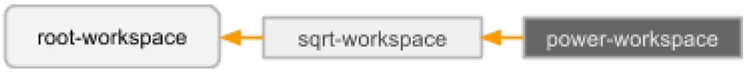
getSquareRootFractionExponentItem is then passed to the MathJax function SFE_UpdateMathjax(getSquareRootFractionExponentItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " squareRootFractionExponentOutput".

Also from this workspace getSquareRootFractionExponentItem is appended to getSquareRootItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getSquareRootItem) for viewing prior to validation.

The user can ‘Reset’ or ‘Cancel’ work done in the current workspace or ‘Insert’ to validate work added to fraction Workspace

NOTE: In this context all items created in the power workspace are added to the parent fraction workspace once validated.

Power Workspace

Workflow - 

Javascript file: squareRootExponentBuilder.js

Display workspace using HTML element with ID attribute "squareRootExponentPanel" via showSquareRootExponentBuilder().

Values entered from this workspace are passed to squareRootExponentBuilder() and appended to global var **getSquareRootExponentItem.**

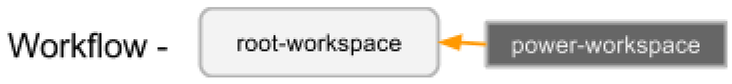
getSquareRootExponentItem is then passed to the MathJax function SE_UpdateMathjax(getSquareRootExponentItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " squareRootExponentOutput".

Also from this workspace getSquareRootExponentItem is appended to getSquareRootItem and temporarily passed and typeset to the root-workspace via mainUpdateMathjax(concatEquationItems + getSquareRootItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to sqrt Workspace

NOTE: In this context all items created in the power workspace are added to the parent sqrt workspace once validated.

Power Workspace



Javascript file: exponentBuilder.js

Display workspace using HTML element with ID attribute "exponentPanel" via showExponentBuilder().
Values entered from this workspace are passed to exponentBuilder() and appended to global var **getExponentItem.**

getExponentItem is then passed to the MathJax function E_UpdateMathjax(getExponentItem) as TEX. This function will typeset mathematical notation to the HTML element with id attribute " exponentOutput".

Also from this workspace getExponentItem is temporarily passed and typeset to the root workspace via mainUpdateMathjax(concatEquationItems + getExponentItem) for viewing prior to validation.

The user can 'Reset' or 'Cancel' work done in the current workspace or 'Insert' to validate work added to root Workspace.

NOTE: In this context all items created in the power workspace are added to the parent root workspace once validated

Creating Questions on the fly

User activates specific question types using the selection menu located in the HTML element with ID attribute "questionFeed". Two question types are available to date: Linear Equations and Quadratic Expressions.

'New' btn calls LErandomLinearEquation() see below to build Linear equations. The result is then passed to the global var inputLinEqn. The contents of this variable is then passed as an argument to getInputLine(inputLinEqn) which outputs the new question to the HTML element with ID attribute "p1"

'New' btn can also call type3FactorableQuadratic() see below to build quadratic expressions activated via the selection menu. The result is then passed to the global var inputQuadratic. The contents of this variable is then passed as an argument to getInputLine(inputQuadratic) which outputs the new question to the HTML element with ID attribute "p1"

Tara provide a brief explanation of each:
LErandomLinearEquation()
type3FactorableQuadratic()

Solution to question

'Show' btn calls both LEdisplayLinearEqnSolution() to show solutions for Linear equation question types and factorRationalQuadraticExpressionSolution() to show solutions for quadratic expression question types. Both output their respective solution to the HTML element with ID attribute "p2"

Question History

Javascript file: questionSelector.js

New questions are added to history using the "OBJECT CONSTRUCTOR" new questionObject(questionType,questionValue). Every time this constructor is activated the question is stored in an array: questionHistory[]. The array enables the user to peruse questions for the current browser session using the "Previous" and "Next" btns

Keypad Input - Add items to workspaces via keypad.

Javascript file: numberPadInput.js

A static label/number is assigned to each workspace to determine where, in what context, items entered will be queued and rendered. When you open a workspace a label is assigned to the global var keyBoardInputPanelNumber
E.g.
keyBoardInputPanelNumber = “0” directs input into root workspace context
keyBoardInputPanelNumber = “1” directs input into Fraction workspace context

In the above e.g. When the user enters items from the root context, keyBoardInputPanelNumber is set to ‘0’. When the user accesses the Fraction workspace, keyBoardInputPanelNumber is set to ‘1’ and items are enter to that context. Once work is concluded in the Fraction workspace, keyBoardInputPanelNumber is set to 0 so items are now entered again to the root workspace

See numberPadInput.js for a complete description for how workspaces are labeled in reference to Keypad input.

Editing workspace entries (Future Development)

To date each workspace is able to reset items only. It will be necessary to develop an “UNDO/REDO” mechanism, enabling users to edit entries more readily stepping back through or re-adding one item at a time, updating the current and root queues/workspaces. This will give users the necessary tools to control and edit input.