Document Instance Reference July. 3, 2024

Document instances are generated from document templates based on a fully generated set of variable values.

Document instances may contain inputs which acta as variable values and can react to changes in input values by updating document instance objects (say, updating a graph as the user enters an expression in x), or moving a framed animation to a different frame, or starting or stopping playback, etc. However, these are all declarative and fixed in the document instance – updates such as these are handled within the renderer, not within the document itself.

Document instance structures are designed to support web-based presentation via HTML, SVG, MathML, and ECMAScript, or can generate PDF versions of content. Styling is aligned with CSS but rather than allow every element to have every property, document instance objects only allow the properties that affect their presentation or behavior.

When used in the content of a question, document content will be emitted within an HTML form, with a submit button to trigger processing, and changes to input values will be reflected in corresponding form input values.

When used as a solution, block of instructions, or other non-question text, inputs may still be included, and entries can still affect other document elements or trigger scripts, but data will not be submitted (a page refresh will return state to its original value).

In general, when a user enters a value in an input, the input value is stored in a form variable for submission, but other elements can react in real-time. The value can be stored (perhaps after some calculation) in attributes or properties of document elements or named script functions can be triggered. For example, a user-entered value can update a graphed function, the visible window represented in a graph, or elements can be made visible or hidden, or enabled/disabled. This mechanism is leveraged by the accessibility controls element to toggle between multiple available representations of content.

Element summary:

DocColumnInst  
DocVSpaceInst  
DocHeading  
DocBlockInst  
DocHRuleInst  
DocPreInst  
DocListInst  
DocListItemInst  
DocDetailsInst  
DocSummaryInst  
DocFigureInst  
DocFigCaptionInst  
DocEmInst  
DocStrongInst  
DocSmallInst  
DocQuoteInst  
DocDfnInst  
DocAbbrInst  
DocVarInst  
DocNumberInst  
DocSpanInst  
DocHSpaceInst  
DocCodeInst  
DocImageInst  
DocVideoInst  
DocAudioInst  
DocTableInst  
DocDrawingInst  
Doc3DModelInst  
DocSymbolPaletteInst  
DocFieldsetInst  
DocLabelInst  
DocMathInst  
Math primitive summary:

MathIdentifierInst  
MathNumberInst  
MathOperatorInst  
MathTextInst  
MathSpaceInst  
MathRowInst  
MathSpanInst  
MathPhantomInst  
MathFractionInst  
MathRadicalInst  
MathRelOffsetInst  
MathFencedInst  
MathEnclosedInst  
MathTabularInst  
MathStackInst  
MathLongDivInst

Input element summary:

InputIntegerInst (Long)  
InputRealInst (Number)  
InputSliderInst (Number)  
InputIntVectorInst (Long[])  
InputRealVectorInst (Number[])  
InputIntervalInst (Interval)  
InputStringInst (String)  
InputTextInst (Span)  
InputSelectInst (Long)  
InputLikertInst (Long)  
InputExpressionInst (Epxr)  
InputRadioButtonInst (Long)  
InputCheckboxInst (Long)  
InputDrawingInst (Number[])  
InputMathStepsInst (File)  
InputFileUploadInst File  
InputHintBtnInst (Long)  
Output element summary:

DocOutputInst  
DocProgressInst  
DocMeterInst  
DocRenderedExpressionInst  
DocNumberLineInst  
DocAnimationInst  
DocScriptInst

Drawing primitive summary:

PrimGroupInst  
PrimLineInst  
PrimCircleInst  
PrimEllipseInst  
PrimRectangleInst  
PrimPolylineInst  
PrimPolygonInst  
PrimPathInst  
PrimArcInst  
PrimTextInst  
PrimTextSpanInst  
PrimTextPathInst  
PrimMath  
PrimRasterInst  
PrimSpanInst  
PrimProtractorInst  
PrimRulerInst  
PrimAxes  
PrimFunctionGraphInst  
PrimPieChart  
PrimBarChart  
PrimLineChart

Other elements:

DocAccessibilityControls

Compound CSS Property Values

There are many sets of CSS properties that are related and always supported as a group, such as font specifications, border or padding parameters, line styles, etc. Rather than make every object implement each property, we provide compound objects to contain related properties.

**BackgroundSpec**

* background-color (**Color**, as a name or as a numeric or functional specification)
* background-image (**URI**)
* background-position-x (**Percentage**)
* background-position-y (**Percentage**)
* background-repeat-x (**Boolean**)
* background-repeat-y (**Boolean**)

**BorderSpec**

* border-top-color (**Color**, as a name or as a numeric or functional specification)
* border-right-color (**Color**, as a name or as a numeric or functional specification)
* border-bottom-color (**Color**, as a name or as a numeric or functional specification)
* border-left-color (**Color**, as a name or as a numeric or functional specification)
* border-top-style (**enumerated** border style value)
* border-right-style (**enumerated** border style value)
* border-bottom-style (**enumerated** border style value)
* border-left-style (**enumerated** border style value)
* border-top-width (**enumerated** border width value or a **Length**)
* border-right-width (**enumerated** border width value or a **Length**)
* border-bottom-width (**enumerated** border width value or a **Length**)
* border-left-width (**enumerated** border width value or a **Length**)
* border-top-left-radius (**Length** or **Percentage**)
* border-top-right-radius (**Length** or **Percentage**)
* border-bottom-right-radius (**Length** or **Percentage**)
* border-bottom-left-radius (**Length** or **Percentage**)
* outer-shadow-color (**Color**, as a name or as a numeric or functional specification)
* outer-shadow-h-offset (**Length**)
* outer-shadow-v-offset (**Length**)
* outer-shadow-blur-radius (**Length**)
* outer-shadow-spread-distance (**Length**)
* inner-shadow-color (**Color**, as a name or as a numeric or functional specification)
* inner-shadow-h-offset (**Length**)
* inner-shadow-v-offset (**Length**)
* inner-shadow-blur-radius (**Length**)
* inner-shadow-spread-distance (**Length**)
* outline-color (**Color**, as a name or as a numeric or functional specification)
* outline-style (**enumerated** border style value)
* outline-width (**enumerated** border width value or a **Length**)

**BoxSpec**

* margin-top (**Length**)
* margin-right (**Length**)
* margin-bottom (**Length**)
* margin-left (**Length**)
* padding-top (**Length**)
* padding -right (**Length**)
* padding -bottom (**Length**)
* padding -left (**Length**)

**TextSpec**

* font-family (**enumerated** font family)
* font-weight (**enumerated** font weight value)
* font-stretch (**enumerated** font stretch value)
* font-style (**enumerated** font style value)
* font-size (**enumerated** absolute or relative size or a **Length** or a **Percentage**)
* font-size-adjust (**Double**)
* line-height (non-negative **Double** or a **Length** or **Percentage**)
* text-align (**enumerated** text align value)
* text-decoration (set of **enumerated** text decoration values)
* white-space (**enumerated** white space value)
* letter-spacing (**Length**)
* word-spacing (**Length**)

Docement Tree Nodes

A document instance is represented as a tree of nodes, where each node may have any number of attributes (considered an unordered map from attribute name to value), and an ordered list of zero or more child nodes.

All nodes extend a common base class that allows every node to have a unique ID and to have one or more CSS style classes.

AbstractDocObjectInst

The base class for all document tree nodes.

Attributes:

* id (**String** value used by scripts to control node attributes)
* class (List of **String** values used to indicate style classes)

Input Elements

We begin with the input elements since user interactions with these elements can potentially affect all other elements in the document.

Inputs store values in HTML input elements within HTML forms for submission to the server. Actions can be triggered when input values change, causing changes in element attributes or execution of scripts. Ultimately, all form input values are strings or file uploads, but certain types of inputs generate strings that are guaranteed to represent valid values of other types. Supported types include:

1. Integers (representable as either Integer or Long)
2. Numbers (representable as Long, Double, or Irrational)
3. Integer Vectors (representable as array of Long)
4. Number Vectors (representable as array of entries, each a Long, Double, or Irrational)
5. Intervals (representable as Interval)
6. Booleans (representable as Boolean)
7. Strings
8. Expressions (see the text form for expressions, documented separately, where a mathematical statement is considered a "Boolean-valued expression", and so these can also be entered)
9. File upload (arbitrary data, which includes generated files from mathematical work capture)

All inputs should participate in the "focus tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read aloud by a screen-reader as the user tabs to the field.

All fields should support entry through keyboard only without use of mouse or touch gestures and should support "up/down/tab" entry of values in some way, perhaps with the accessibility controls turning these features on or off. Numeric entry inputs should have a "spinbutton" ARIA role and should define a step size for these "up/down" steps. Care should be taken by authors not to make ranges too large to effectively enter values with this system.

AbstractInputInst : AbstractDocObjectInst

The base class for all inputs. This base class stores information needed to submit input values.

Attributes:

* name (**String** form variable name under which input value will be submitted)
* enabled-var (**String** name of an input value used to control enabled state)
* enabled-value (**IntegerVector** with values of input that should enable this control)
* disabled-value (**IntegerVector** with values of input that should disable this control)
* required (**Boolean**, indicating form cannot be submitted without a value entered)
* tabindex (**Integer** index in the form tab order)
* tooltip (**String** tool-tip text)

AbstractInputFieldInst : AbstractInputInst

The base class for inputs that present themselves to the user as a field that they can type into. This base class stores font and display-related attributes that control field presentation.

Attributes:

* text-spec (**TextSpec**)
* background-spec (**BackgroundSpec**)
* border-spec (**BorderSpec**)
* box-spec (**BoxSpec**)
* placeholder (**String** text to show when no value has been entered)

InputIntegerInst : AbstractInputFieldInst (Long-valued)

An input through which users can enter an integer value. Users can enter a leading + or – sign followed by digits.

Depending on the assistive settings, the input could be presented with spin buttons for the whole value, or spin buttons for each digit for larger values.

Attributes:

* min (**Long**)
* max (**Long**)
* step (**Long**)
* treat-empty-as (**Long**, value to use when the user does not enter a value)
* treat-minus-as (**Long**, value to use if the user enters just a minus sign)
* treat-plus-as (**Long**, value to use if the user enters just a plus sign)
* spin-mode {single | each-digit}

InputRealInst : AbstractInputFieldInst (Number-valued)

An input through which users can enter a real value, in the form of a decimal, rational, or irrational number, with a specified number of "meaningful" decimal places after the radix. Scientific notation is supported.

Depending on the assistive settings, the input might have spin buttons or other buttons to enter quantities like π or *e* or a fraction slash.

Attributes:

* min (**Number**)
* max (**Number**)
* decimal-places (**Integer** number of positions after the radix that are meaningful)
* allow-decimal (**Boolean**)
* allow-fraction (**Boolean**)
* allow-pi-irrational (**Boolean**)
* allow-e-irrational (**Boolean**)
* allow-root-irrational (**Boolean**)
* allow-pos-inf (**Boolean**)
* allow-neg-inf (**Boolean**)
* treat-empty-as (**Number**, value to use when the user does not enter a value)
* treat-minus-as (**Number**, value to use if the user enters just a minus sign)
* treat-plus-as (**Number**, value to use if the user enters just a plus sign)
* spin-mode {single | each-digit}

InputSliderInst : AbstractInputInst (Number-valued)

An input that allows the user to select a number by sliding a handle back and forth along a scale. The lower and upper bounds, an initial value, and a step size are specified, and only values that fall between the bounds (inclusive) and are a integer number of steps from the initial value will be generated.

Attributes:

* min (**Number**)
* max (**Number**)
* initial (**Number**)
* step (**Number**)
* orientation {vertical | horizontal}
* scale-style {line | slim | fat}
* tick-style {none | dot | line}
* tick-label-style {none | above-left | below-right | within}
* label-text-spec (**TextSpec**)

InputIntVectorInst : AbstractInputFieldInst (Long Vector-valued)

This input through which the user can enter a list of integers. Behavior matches that of an integer input, with the ability to add or remove entries if the min/max values differ.

Attributes:

* min-entries (**Integer**)
* max-entries (**Integer**)
* entry-min (**Long** or **Long**[])
* entry-max (**Long** or **Long**[])
* entry-step (**Long**)

InputRealVectorInst : AbstractInputFieldInst (Number Vector-valued)

This input through which the user can enter a list of real numbers. Behavior matches that of a real number input, with the ability to add or remove entries if the min/max values differ.

Attributes:

* min-entries (**Integer**)
* max-entries (**Integer**)
* entry-min (**Number** or **Number**[])
* entry-max (**Number** or **Number**[])
* entry-step (**Number**)
* decimal-places (**Integer** number of positions after the radix that are meaningful)
* allow-decimal (**Boolean**)
* allow-fraction (**Boolean**)
* allow-pi-irrational (**Boolean**)
* allow-e-irrational (**Boolean**)
* allow-root-irrational (**Boolean**)
* allow-pos-inf (**Boolean**)
* allow-neg-inf (**Boolean**)

InputIntervalInst : AbstractInputFieldInst (Long-valued)

An input through which the user can enter an interval, including a choice of closure at each bound. Bounds are entered as real numbers.

Attributes:

* lower-bound-min (**Number**)
* lower-bound-max (**Number**)
* upper-bound-min (**Number**)
* upper-bound-max (**Number**)
* bound-step (**Number**)
* decimal-places (**Integer** number of positions after the radix that are meaningful)
* allow-decimal (**Boolean**)
* allow-fraction (**Boolean**)
* allow-pi-irrational (**Boolean**)
* allow-e-irrational (**Boolean**)
* allow-root-irrational (**Boolean**)
* allow-pos-inf (**Boolean**)
* allow-neg-inf (**Boolean**)

InputStringInst : AbstractInputFieldInst (String-valued)

An input through which the user can enter a single-line String.

Attributes:

* min-length (**Integer**, the minimum number of UTF-16 code units required)
* max-length (**Integer**, the maximum number of UTF-16 code units allowed)
* pattern (**RegEx**, a regular expression the input must match to be valid)
* columns (**Integer**, number of characters wide the input should appear)
* autocapitalize (**Enumeration** of autocapitalize styles)
* spellcheck (**Boolean**)

InputTextInst : AbstractInputFieldInst (String-valued)

An input through which the user can enter a multi-line String.

Attributes:

* columns (**Integer**, number of characters wide the input should appear)
* rows (**Integer**, number of lines tall the input should appear)
* wrap (**Enumeration** of wrap styles)
* min-length (**Integer**, the minimum number of UTF-16 code units required)
* max-length (**Integer**, the maximum number of UTF-16 code units allowed)
* autocapitalize (**Enumeration** of autocapitalize styles)
* spellcheck (**Boolean**)

InputSelectInst : AbstractInputFieldInst (Long-valued)

An input that allows the user to select from a list of choices using a drop-down. Items from which to select are child elements.

Attributes:

* style {dropdown | spin | drag-drop}

Children:

* One or more {SelectOptionGroupInst | SelectOptionInst}

SelectOptionGroupIns

A group of options within a selection input.

Attributes:

* label (**String**)

SelectOptionInst

A single option within a selection input

Attributes:

* value (**Integer** value to submit when this option is chosen)
* text (**String** option content, with special characters supported)

InputLikertInst : AbstractInputFieInst (Long-valued)

An input in the style of a Likert scale, with a set of numbered options. This is a short-hand for a common use-case of a select input, but where choice content is simple strings.

Attributes:

* values (List of **Integer**)
* choices (List of **String**, same number of items as in values)

InputExpressionInst : AbstractInputFieldInst (Expression-valued)

An input that allows the user to enter a mathematical expression.

Attributes:

* allow-equation (**Boolean**)
* allow-inequality (**Boolean**)
* allowed-functions (List of **String**)
* allowed-variables (List of **String**)

InputRadioButtonInst : AbstractInputFieldInst (Long-valued)

An input that presents a radio button. Multiple radio button inputs can have the same ID, and each must have a unique value within that ID. Only one radio button with a given ID can be selected at a time.

Attributes:

* style {filled-circle | checked-circle | x-circle | filled-box | checked-box | x-box | depressed | highlighted}
* value (**Integer**)

InputCheckboxInst : AbstractInputFieldInst (Long-valued)

An input that presents a checkbox. Multiple checkbox inputs can have the same ID, in which case the value submitted is the sum of the values of all selected checkboxes. A common technique, then, is to set values to powers of 2 (like 1, 2, 4, 8, 16, etc.) to ensure the set of selected boxes can be determined uniquely.

Attributes:

* style {filled-circle | checked-circle | x-circle | filled-box | checked-box | x-box | depressed | highlighted}
* value (**Integer**)

InputDrawingInst : AbstractInputInst (Real vector-valued)

An input that presents a drawing that can include an interactive primitive. The type of primitive dictates the interpretation of the submitted list of numbers.

Note that "drawing" is a blanket term for basic drawings as well as charts and graphs.

Option 1: the user can select one or more discrete points.

Option 2: the user can define a line segment or line by selecting two points.

Option 3: the user can create a piecewise linear graph by selecting function values at some number of points

Option 4: the user can drag a primitive around to position it within the graph, and optionally rotate or scale the primitive.

Option 5: the user can select a primitive by clicking, with the currently selected primitive highlighted.

Option 6 the user can select an angle by clicking somewhere – a ray is then drawn from the origin to the point, defining an angle.

Attributes:

* (all attributes allowed on a graph)

InputMathStepsInst : AbstractInputFieldInst (File-valued)

An input that allows the user to enter a sequence of mathematical manipulations. Each line is submitted as an expression or mathematical statement, or as explanatory text. The user has the option to copy the line above, then to perform some algebraic operation (like "add 4 to both sides", or "divide both sides by 6") to move to the next step.

Attributes:

* ...

InputFileUploadInst : AbstractInputInst (File-valued)

An input that allows the user to upload an arbitrary file.

Attributes:

* extension-types (List of file extensions allowed)

InputHintRequestInst : AbstractInputInst (Long-valued)

An input that allows the user to request a hint. Hints are numbered, and the act of requesting a hint shows that hint's content and (if there are additional hints) enables a button to request the next hint.

Attributes:

* hint-number (**Long**)

Math Primitives

Math primitives represent any mathematical construction, including inline references to variables or inline numeric quantities. They can be rendered in a different font or color for emphasis and to make it easier for students to pick out relevant variables and quantities from surrounding text.

Math can appear inline (baseline-aligned to surrounding text), as a block-mode display, or as a primitive in a drawing. Math objects exist within a container object, much like drawing primitives exist in a containing drawing object.

AbstractMathInst : AbstractDocObjectInst

The base class for all mathematical primitives. This base class defines default formatting such as font and color settings for use within math constructions.

Attributes:

* background-color (**Color**, as a name or as a numeric or functional specification)
* color (**Color**, as a name or as a numeric or functional specification)

AbstractMathTokenInst : AbstractMathInst

The base class for mathematical primitives that behave as tokens (things displayed with some quantity of space around them in a costruction).

Attributes:

* variant {normal | bold | italic | bold-italic | double-struck | bold-fraktur | script | bold-script | fraktur |  
   sans-serif | bold-sans-serif | sans-serif-italic | sans-serif-bold-italic | monospace | initial |  
   tailed | looped | stretched}
* size {small | normal | big | <length>}

MathIdentifierInst : AbstractMathTokenInst

An object that represents a mathematical identifier like a variable name. We define a unique class for identifiers to allow control over the spacing between identifiers and other types of math primitive.

MathNumberInst : AbstractMathTokenInst

An object that represents a number, including symbols like π, *e*, or ∞. We define a unique class for numbers to allow control over the spacing between numbers and other types of math primitive.

MathOperatorInst : AbstractMathTokenInst

An object that represents an operator. We define a unique class for numbers to allow control over the spacing between numbers and other types of math primitive. Note that MathML includes fences in this class, but we define a separate fenced construction instead.

Attributes:

* form {prefix | infix | postfix}
* separator (**Boolean**)
* symmetric (**Boolean**)
* large (**Boolean**)
* movable-limits (**Boolean**)
* lspace (**Length**)
* rspace (**Length**)
* stretchy (**Boolean**)
* maxsize (**Length**)
* minsize (**Length**)

MathTextInst : AbstractMathTokenInst

An object that represents text appearing within math, like "rise / run" or "opposite / hypotenuse".

MathSpaceInst : AbstractMathTokenInst

The base class for space added within mathematical constructions.

Attributes:

* width (**Length**)
* ascent (**Length**)
* descent (**Length**)

MathRowInst : AbstractMathInst

An object that represents a row of laid out math. For example, the numerator or denominator of a fraction construction, the interior of a radical, or a stand-alone expression.

Children:

* One or more {AbstractMathInst}

MathSpanInst : AbstractMathInst

A sequence of mathematical objects with styles applied. The MathML equivalent is the mstyle element.

Children:

* One or more {AbstractMathInst}

MathPhantomInst  
MathFractionInst  
MathRadicalInst  
MathRelOffsetInst  
MathFencedInst  
MathEnclosedInst  
MathTabularInst  
MathStackInst  
MathLongDivInst