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  
 HTML: <div>

DocVSpaceInst  
 HTML: <div>

DocHeading  
 HTML: <h#>

DocBlockInst  
 HTML: <p>, <div>

DocHRuleInst  
 HTML: <hr>

DocPreInst  
 HTML: <pre>

DocListInst  
 HTML: <ol>, <ul>

DocListItemInst  
 HTML: <li>

DocDetailsInst  
 HTML: <details>

DocSummaryInst  
 HTML: <details>

DocFigureInst  
 HTML: <figure>

DocFigCaptionInst  
 HTML: <figcaption>

DocEmInst  
 HTML: <em>

DocStrongInst  
 HTML: <strong>

DocSmallInst  
 HTML: <small>

DocQuoteInst  
 HTML: <q>

DocDfnInst  
 HTML: <dfn>

DocAbbrInst  
 HTML: <abbr>

DocVarInst  
 HTML: <var>

DocNumberInst  
 HTML: <data>

DocSpanInst  
 HTML: <span>

DocHSpaceInst  
 HTML: <span>

DocCodeInst  
 HTML: <code>

DocImageInst  
 HTML: <image>

DocVideoInst  
 HTML: <video>

DocAudioInst  
 HTML: <audio>

DocTableInst  
 HTML: <table>

DocDrawingInst  
 SVG: <svg>

Doc3DModelInst  
 HTML: <canvas>

DocSymbolPaletteInst  
 HTML: <svg>

DocFieldsetInst  
 HTML: <fieldset>

DocLabelInst  
 HTML: <label>

Math element summary:

DocMathInst  
 HTML: <svg>

MathIdentifierInst  
 SVG: (custom)

MathNumberInst  
 SVG: (custom)

MathOperatorInst  
 SVG: (custom)

MathTextInst  
 SVG: (custom)

MathRowInst  
 SVG: (custom)

MathSpanInst  
 SVG: (custom)

MathPhantomInst  
 SVG: (custom)

MathFractionInst  
 SVG: (custom)

MathRadicalInst  
 SVG: (custom)

MathRelOffsetInst  
 SVG: (various)

MathFencedInst  
 SVG: (custom)

MathEnclosedInst  
 SVG: (custom)

MathTabularInst  
 SVG: (custom)

MathStackInst  
 SVG: (custom)

MathLongDivInst  
 SVG: (custom)

MathSpaceInst  
 SVG: (custom)

Input element summary:

InputIntegerInst (Long)  
 HTML: <input type='number'>

InputRealInst (Number)  
 HTML: <input type='number'>

InputIntVectorInst (Long[])  
 HTML: <input type='text'>

InputRealVectorInst (Number[])  
 HTML: <input type='text'>

InputIntervalInst (Interval)  
 HTML: <input type='text'>

InputStringInst (String)  
 HTML: <input type='text'>

InputTextInst (Span)  
 HTML: <textarea>

InputSelectInst (Long)  
 HTML: <select>

InputSimpleExpressionInst (Epxr)  
 HTML: <input type='text'>

InputExpressionInst (Epxr)  
 HTML: (custom)

InputRadioButtonInst (Long)  
 HTML: <input type='radio'>

InputCheckboxInst (Long)  
 HTML: <input type='checkbox'>

InputGraphXYInst (Number[])  
 HTML: <svg>

InputMathStepsInst (???)  
 HTML: <svg>

InputFileUploadInst (???)  
 HTML: <input type='file'>

InputHintBtnInst (Long)  
 HTML: <input type='button'>

Output element summary:

DocOutputInst  
 HTML: <output>

DocProgressInst  
 HTML: <progress>

DocMeterInst  
 HTML: <meter>

DocRenderedExpressionInst  
 MathML: <svg>

DocNumberLineInst  
 HTML: (custom)

DocAnimationInst  
 HTML: (custom)

DocScriptInst  
 HTML: <script>

Drawing primitive summary:

PrimGroupInst  
 SVG: <g>

PrimLineInst  
 SVG: <line>

PrimCircleInst  
 SVG: <circle>

PrimEllipseInst  
 SVG: <ellipse>

PrimRectangleInst  
 SVG: <rect>

PrimPolylineInst  
 SVG: <polyline>

PrimPolygonInst  
 SVG: <polygon>

PrimPathInst  
 SVG: <path>

PrimArcInst  
 SVG: <path>

PrimFunctionGraphInst  
 SVG: <path>

PrimTextInst  
 SVG: <text>

PrimTextSpanInst  
 SVG: <tspan>

PrimTextPathInst  
 SVG: <textPath>

PrimRasterInst  
 SVG: <image>

PrimSpanInst  
 SVG: (custom)

PrimProtractorInst  
 SVG: (custom)

PrimRulerInst  
 SVG: (custom)

PrimAxes  
 SVG: (custom)

Other elements:

DocAccessibilityControls  
 HTML: (custom)

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**)

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.

AbstractInputFieldInst

The base class for all input field instances.

Attributes:

* id (**String**, the ID under which to submit this input's value)

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}

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 (Interval-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)
* 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 (Integer-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:

* prompt (**String**, to display before a choice has been made)

InputExpressionInst : AbstractInputFieldInst (Expression-valued)

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

Attributes:

* prompt (**String**, to display before anything has been entered)
* allow-equation (**Boolean**)
* allow-inequality (**Boolean**)
* allowed-functions (List of **String**)
* allowed-variables (List of **String**)

InputRadioButtonInst : AbstractInputFieldInst (Integer-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:

* value (**Integer**)

InputCheckboxInst : AbstractInputFieldInst (Integer-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:

* value (**Integer**)

InputDrawingInst : AbstractInputFieldInst (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.

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 : AbstractInputFieldInst (File-valued)

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

Attributes:

* ...

InputHintRequestInst : AbstractInputFieldInst (Integer-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:

* ...

InputTextInst : AbstractInputFieldInst (String-valued)

This input is presented in HTML as a <textarea> element that supports entry of a multi-line text value, and which may be resizable when presented.

Attributes:

* minlength (**Integer**, the minimum number of UTF-16 code units required)
* maxlength (**Integer**, the maximum number of UTF-16 code units allowed)
* cols (**Integer** number of columns)
* rows (**Integer** number of rows)
* wrap (**Enumeration** of wrap styles)
* autocapitalize (**Enumeration** of autocapitalize styles)
* spellcheck (**Boolean**)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.
* Users who can operate a keyboard can type a value directly.
* Users with speech-to-text could speak content to enter content into the field.
* The field has the ARIA role "textbox".

InputSelectInst : AbstractInputFieldInst (Integer-valued)

This input is presented in HTML as a <select> element with a set of contained <optgroup> and <option> elements whose values can be parsed as integers.

Attributes:

* (none)

Children:

* One or more {SelectOptionGroupInst | SelectOptionInst}

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.
* Users who can operate a keyboard can use up and down arrows to select an item.
* Users with speech-to-text could potentially speak an item name to have it selected or could say letters to begin spelling the desired item until it was selected.
* The field has the ARIA role "combo".

SelectOptionGroupIns

This element is presented input is presented in HTML as a <optgroup> element with a set of contained <option> elements whose values can be parsed as integers.

Attributes:

* label (**String**)
* disabled (**Boolean**)
* 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)

Accessibility:

* The element has the ARIA role "group.

SelectOptionInst

This input is presented in HTML as an <option> element whose value can be parsed as an integer.

Attributes:

* text (**String** option content, with special characters supported)

Accessibility:

* The field has the ARIA role "option".

InputSimpleExpressionInst : AbstractInputFieldInst (Expression-valued)

This input is presented in HTML as a <text> element that accepts strings that can be parsed as simple real-valued expressions. The primary purpose is to support entry of fractions (using slash characters) or rational multiples of constants like **π** or **e**. Constructions like fractions (represented vertically), roots, exponents, arithmetic operations, function invocations, vectors and matrices, integration, and differentiation, etc. are not supported.

Attributes:

* allow-slash (**Boolean**)
* allow-pi (**Boolean**)
* allow-e (**Boolean**)
* allow-pos-inf (**Boolean**)
* allow-neg-inf (**Boolean**)
* treat-empty-as (**Double**, value to use when the user does not enter a value)
* treat-minus-as (**Double**, value to use if the user enters just a minus sign)
* treat-plus-as (**Double**, value to use if the user enters just a plus sign)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.
* Users who can operate a keyboard can type a value directly.
* Users with speech-to-text could speak content to enter content into the field.
* The field has the ARIA role "textbox".

InputExpressionInst : AbstractInputFieldInst (Expression-valued)

This input is presented in HTML as a custom collection of elements and an <svg> element that renders the expression as it is entered. It allows the user to dynamically construct an expression, subject to constraints specified in attributes, then submits the expression in text form. The expression will show errors if it is not yet valid, and when the cursor is adjacent to a fence structure, the matched fence pair will be highlighted.

Attributes:

* display (**Enumerated** format, either inline or block or table-cell)
* allow-fractions (**Boolean**)
* allow-roots (**Boolean**)
* allow-exponents (**Boolean**)
* allow-fences (**Boolean**)
* symbols (**List** of special symbols for which to include buttons)
* functions (**List** of function names that are supported)

Accessibility:

* The <svg> element that displays the rendered equation is focusable and accepts keyboard input, allowing expressions to be entered entirely through the keyboard.
* There are also ancillary buttons to control structure, such as adding a fraction bar, wrapping the selection in a radical, applying pre-defined functions, adding exponents, wrapping with parentheses, etc.
* Users who can operate a keyboard can type a value directly.
* Users with speech-to-text could speak content to enter content into the field.
* The SVG element should update its "alt text" constantly to reflect the expression structure so a user could tab away and tab back and hear the constructed expression read aloud.

InputRadioButtonInst : AbstractInputInst (Integer-valued)

This input is presented in HTML as a <input type='radio'> element. All radio buttons with the same name are considered a group, each should have an integer value that will be submitted when that button is selected.

Attributes:

* value (**Integer**)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.

InputCheckboxInst : AbstractInputInst (Integer-valued)

This input is presented in HTML as a <input type='checkbox'> element. If multiple checkboxes have the same name, the submitted value will be the sum of the integer values for each, so the integer values should be chosen to be bitwise-non-overlapping (like 1, 2, 4, 8, 16, 32, etc.).

Attributes:

* value (**Integer**)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.

InputFileUploadInst : AbstractInputInst (File-valued)

This input is presented in HTML as a <input type='file'> element.

Attributes:

* extension-types (List of file extensions allowed)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.

InputMathWorkInst : AbstractInputInst (File-valued)

This input is presented in HTML as a collection of elements. It supports the creation of a sequence of mathematical steps and comments. Each step is a mathematical statement or expression. Users can copy the step above then make modifications, like moving terms across the relation, multiplying both sides by some number or expression, raising both sides to some power, taking functions of both sides, replacing a variable with a value, or performing some simplification. New lines can be constructed from scratch as well. The result is a set of lines of text, each with a prefix that identifies it as a comment or expression. Instructors can provide feedback (in the form of new comments or expressions tagged as feedback) and integrate those into the document for review by the student.

Attributes:

* extension-types (List of file extensions allowed)

Accessibility:

* The field participates in the form's "tab sequence" for keyboard-based access to field focus and should have a label object with a sensible label that can be read as the user tabs to the field.

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)

AbstractInputInst : AbstractDocObjectInst

The base class for all inputs. This base class stores information needed to submit input values, and to trigger actions on value changes.

Attributes:

* name (**String** form variable name under which input value will be submitted)
* disabled (**Boolean**)
* readonly (**Boolean**)
* required (**Boolean**, indicating form cannot be submitted without a value entered)
* tabindex (**Integer** index in the form tab order)
* tooltip (**String** tool-tip text)

AbstractDocObjectInst

The base class for all document tree nodes.

Attributes:

* id (**String** value used by scripts to control node attributes)
* class (**String** value used to indicate a style class)