Positron Dynamics Reference

Glossary

view/selector paradigm

In many actions and tags, Positron uses a convention known as the view/selector paradigm to identify target elements on which the operation takes place. This is similar in concept to a CSS selector, and indeed can incorporate a CSS selector. There are two fields in the selector, either of which is optional. The first field specifies the key of a view - if this is specified, then the view's associated element becomes the root of any further selection. The second field specifies a CSS selector - if this is specified, then the selector is used to further select elements off the root element.

Examples --

"menu" - select the associated element of the "menu" view.

"menu/.square" - select anything with the class "square" within the associated element of the "menu" view

".square" - select anything in the document with the class "square"

Actions

addclass

Add a specified class to elements specified using the view/selector paradigm.

The class name, view, and selector are given as action arguments. If there are only two action arguments, they are assumed to be class name and selector.

```
<div
  pos-action="(click) addclass: blue/.square"
  >
  Click to make the squares blue.
</div>
```

addtolist

Add a copy of the action's explicit parameters to the list in application context specified by the key in the first action argument. If the list does not currently exist, construct it. Note that duplicate list entries are not allowed.

```
<div
  pos-action-1="(click) addtolist: people"
  pos-action-1-params=”name: bob; job: janitor;”
  pos-action-2=”(click) refreshview: people”
  >Add Bob to the “people” list</div>
```

addtomap

Add a copy of the action’s explicit parameters, keyed by the value explicit parameter whose key is specified in the first action argument, to the list in application context specified by the key in the second action argument.

```
<div
  pos-action-1=”(click) addtomap: name/people”
  pos-action-1-params=”name: bob; job: janitor;”
  pos-action-2=”(click) refreshview: people”
  >Add Bob to the “people” map under the key “bob”</div>
```

ajaxform

Subclass of *validateform*. Validate the form whose <form> element is associated with this action via the “submit” trigger. If the form validates, incorporate the form elements into an Ajax call, rather than let the form submit.

If the Ajax call succeeds, dispatch an “pos-ajax” event, with any JSON result from the call accompanying in the event detail. If the call fails, dispatch an “pos-error” event, with the error accompanying in event detail.

```
<form
  pos-action-1=”(submit) ajaxformaction”
  pos-action-2=”(pos-ajaxformsuccess) showview: users”
  action=”/users/add”
  method=”get”
  >
  <input name=”login” type=”text” required=”true” />
  <input name=”password” type=”password” required=”true” />
  <input name=”submit” type=”submit” value=”Add User” />
</form>
```

ajaxget
ajaxpost

Make an asynchronous Ajax call to the URL specified in the action argument string, with parameters from the associated parameter attribute, if any. If the Ajax call succeeds, dispatch an “pos-ajax” event, with any JSON result from the call accompanying in the event detail. If the call fails, dispatch an “pos-error” event, with the error accompanying in event detail.

Note that ajaxget and ajaxpost are implemented by AjaxAction.

```
<div
    pos-action=”(click) ajaxpost: /users/login”
    pos-action-params=”username: $username;; password: $password;;”
    >
</div>
```

alert

Call alert() with the whole action argument string, if any, as an argument. Mostly useful for debugging.

call

Call the specified method in the enclosing view, page, application, or as a raw global function. Once a method is found, the scope search stops. If the method is not found in any scope, an error message is written to the console.

A optional second argument indicates the specific view off which the call should be made. If this view doesn’t implement the method, then there is no further search. The view can be in the current page, or in the window - ie *not* in another page.

If the action is configured with parameters, then they are set in the receiving view, page, or application, prior to the method being called.

The name of the method is given in an action argument, and the parameters given as action parameters.

```
<div
  pos-action=”(click) call: foobar”
  pos-action-params=”foocolour: blue;”
  >
  Click to call foobar() with a foocolour parameter of blue.
</div>

<div
  pos-action=”(click) call: foobar/menu”
  pos-action-params=”foocolour: blue;”
  >
  Click to call foobar() with a foocolour parameter of blue, in the “menu”
view.
</div>
```

closewebsocket

Close the web socket referred to by the first action argument. Note that if a web socket is not opened by <pos-websocket> and thereby tracked by name, then “closewebsocket” will not be able to find it in order to close it.

dispatchform

Subclass of *validateform*. Validate the form whose <form> element is associated with this action via the “submit” trigger. If the form validates, incorporate the form element values into a “formdispatch” event, and dispatch it. The form is not actually submitted.

This allows form submissions to be directed to an Action via a trigger attached to the “pos-formdispatch” event. The eventlet associated with the event type copies the form values into action parameters.

```
<form
  pos-action-1=”(submit) dispatchform”
  pos-action-2=”(pos-formdispatch) showview: users”
  >
  <input name=”login” type=”text” required=”true” />
  <input name=”password” type=”password” required=”true” />
  <input name=”submit” type=”submit” value=”Add User” />
</form>

<div pos-view=”users:”>
  <div>login passed is $params.login;</div>
</div>
```

gotourl

Copy the entire action string, without separating into discrete arguments, into document.location.href, thereby effecting a page change.

```
<div
  pos-action=”(click) gotourl: /home.html”
  >
  Click to go back home.
</div>
```

hideview

Hide a specified view with an optional transition animation.

The view key and animation CSS class name are given as action arguments. Note that the transition-out class must be defined along with its accompanying key frames.

```
<div
  pos-action=”(click) hideview: square/pos-fade-out-quick”
  >
  Click to hide the “square” view by fading it out quickly.
</div>
```

indexedinsert

Insert the record specified in the parameters into the database and store specified in arguments. Dispatch the prefixed event “indexedinsertsuccess” in the event of success, and “error” in the event of error.

Note that this action uses the HTML5 IndexedDB API. Positron ships a shim to adapt the IndexedDB API to WebSQL for browsers that require it.

```
<div
  pos-action-1="(click) indexedinsert: positron-database-example/users"
  pos-action-1-params="id: 1; login: jason; password: fish;"
  pos-action-2="(pos-indexedinsertsuccess) showview: users"
  >
  click to insert user “jason” with password “fish”
  refreshing the “users” view if the insert succeeds
</div>
```

Bugs: there is currently no good way to manage different types of data - items passed as parameters are assumed to be strings. Don’t use this action yet.

log

Log the first action argument to the console.

refreshview

Similar to “showview”, but the target view’s content is always refreshed.

removeclass

Remove a specified class from elements matching the specified CSS selector using the view/selector paradigm. If there are only two action arguments, they are assumed to be class name and selector.

```
<div
  pos-action=”(click) removeclass: blue/.square”
  >
  Click to make the squares not blue.
</div>
```

removefromlist

Remove the record specified by the action’s explicit parameters from the list in application context specified by the key in the first action argument. The record is identified by a match on all provided explicit parameters.

```
<div
```

```
  pos-action-1=”(click) removefromlist: people”
  pos-action-1-params=”name: bob;”
  pos-action-2=”(click) refreshview: people”
  >Remove Bob from the “people” list</div>
```

removefrommap

Remove the record specified by the value of the explicit parameter whose key is specified in the first action argument, from the list in application context specified by the key in the second action argument.

```
<div
  pos-action-1=”(click) removefrommap: name/people”
  pos-action-1-params=”name: bob;”
  pos-action-2=”(click) refreshview: people”
  >Remove the entry “bob” from the “people” map</div>
```

runview

Run the view specified in the first action argument. Note that the view is rendered immediately invisible -- it is assumed to either have no visible elements or to move or sync its visible elements with other page areas.

Any parameters passed in action parameters are set in the receiving view prior to the section being run.

```
<div
  pos-action=”(click) runview: clockproto”
  >
  Click to run the “clockproto” view.
</div>
```

sendwebsocket

Send a message on the web socket whose name is the first action argument. The message is built from parameters specified in action’s explicit parameters, and serialised to JSON format before sending.

```
<div
    pos-action=”(click) sendwebsocket: authserver”
    pos-action-params=”controller: users; action: login; username: $username;
password: $password;”
    >
</div>
```

setattribute

Set specified attributes on the target element using the view/selector paradigm. If there are only two action arguments, they are assumed to be class name and selector.

The attribute names and values are given as action parameters.

setlocalstorage

Set specified values in local storage.

The local storage property names and values are given as action parameters.

setpage

Set the specified page as active, making the current page, if any, inactive, with optional animations for both transitions.

The page key and transition-in and transition-out class names are given as action arguments. Note that if supplied, transition classes must be defined, along with their accompanying key frames.

```
<div
  pos-action=”(click) setpage: detail/pos-fade-in/pos-fade-out”
  >
  Click to fade out the current page and fade in the detail page.
</div>
```

setparams

Set the specified parameters in the receiving view, page, or application.

The receiving type (view, page, or application) and the corresponding view or page key are given as action arguments. The actual parameters to be set are given as action parameters.

```
<div
  pos-action=”(click) setparams: page/detail”
  pos-action-params=”foocolour: blue;”
  >
  Click to set the foocolour parameter in the detail page to blue.
</div>
```

setstyle

Set the style properties specified in the action parameters on the element described by the action arguments using the view/selector paradigm.

settransform

Set the CSS3 transform specified in the action parameters on the element described by the action arguments using the view/selector paradigm. Recommend using the “prefixedproperty” tag to map ratified CSS3 property names to browser-specific ones.

setvalue

Set the value of the element described by the action arguments using the view/selector paradigm to the value of the “value” action parameter.

showview

Show a specified view with an optional transition animation. Note that its content will only be refreshed if this is the first time the view has ever been shown. If the intent is to refresh the view regardless of current visibility, then use “refreshview” instead.

The view key and animation CSS class name are given as action arguments. Note that the transition-out class must be defined along with its accompanying key frames.

Any parameters passed in action parameters are set in the receiving view prior to the view being shown.

```
<div
  pos-action=”(click) showview: square/pos-fade-in-quick”
  pos-action-params=”foocolour: blue;”
  >
  Click to show the “square” view by fading it in quickly.
</div>
```

toggleclass

Toggle a specified class on elements matching the specified CSS selector using the view/selector paradigm. If only two arguments are provided, they are assumed to be class name and selector.

```
<div
  pos-action=”(click) toggleclass: blue/.square”
  >
  Click to alternately make the squares blue and not blue.
</div>
```

toggleview

Invert the visibility of the the specified view, with optional animations for both transitions.

The view key and transition-in and transition-out class names are given as action arguments. Note that if supplied, transition classes must be defined, along with their accompanying key frames.

Any parameters passed in action parameters are set in the receiving view prior to the view being shown.

```
<div
  pos-action="(click) toggleview: square/pos-fade-in/pos-fade-out"
  pos-action-params="foocolour: blue;"
  >
  Click to alternately fade in and out the "square" view.
</div>
```

validateform

Validate the form whose <form> element is associated with this action via the "submit" trigger. If the form validates, then the "submit" event is allowed to propagate, resulting in the submission of the form.

This action is provided to help deal with the inconsistent way that browsers handle form validation.

See also *ajaxform and dispatchform*.

Attributes

pos-action

Declare an action to be associated with this element. The syntax of an action is as follows -

```
pos-action="(triggername: arg1/arg2/[s|p]) actionname: arg1/arg2/..."
pos-action-params="key1: value1; key2: value2;..."
pos-action-param-keys="key1: value1; key2: value2;..."
pos-action-param-aliases="key1: value1; key2: value2;..."
```

The trigger name is optional, defaulting to "click". If supplied, it is looked up in the triggers section of the application configuration - if a match is found, the appropriate triggerlet class is instantiated and invoked. If a match is not found, then the name is assumed to be a regular DOM event name, and an event listener is attached to the action's element, simply firing the accompanying action when the event is caught.

The trigger arguments configure the trigger, so for example the "interval" trigger allows a period for the interval to be set --

```
pos-action="(interval: 1s) showview: clock"
```

If the last trigger argument contains only "s" or "p" characters, then the presence of "p" is assumed to mean "prevent default event behaviour" and the presence of "s" is assumed to mean "stop event propagation". Note that the default trigger checks for these flags automatically, but custom triggers must do their own checking.

The action name is required, with no default. It is looked up in the actions section of the application configuration - if a match is found, then the appropriate actionlet class is instantiated. The action is then linked to the trigger so that the latter can invoke the former.

The action arguments configure the action, so for example the “showview” action requires at least a view key to be set -

```
pos-action=”showview: menu”
```

Parameters set in the pos-action-params attribute are (generally) parameters to be set in the target of the action, such as a view in the case of the “showview” action, or a page in the case of the “setpage” action. This is of course dependent on the nature of the action.

```
pos-action=”showview: menu”
pos-action-params=”showquitoption: true;”
```

Name value pairs in the pos-action-param-keys attribute are the *keys* of parameters to be set in the target of the action at walk time. This is useful when the parameters concerned are not of primitive types and can therefore not have their values set using pos-action-params.

```
pos-action=”showview: menu”
pos-action-param-keys=”userlist: userlist;”
```

Name value pairs in the pos-action-param-aliases attribute are the *keys* of parameters to be set in the target of the action at *action* time. This is very useful to translate between the parameter namespace of a trigger and its associated action.

```
pos-action=”(location) showview: menu”
pos-action-param-aliases=”latitude: position.coords.latitude; longitude:
position.coords.longitude;”
```

Multiple actions can be attached to an element using an ascending numbering scheme, and actions can have any combination of parameters, keys, and aliases -

```
pos-action-1=”...” pos-action-1-params=”...”
pos-action-2=”...” pos-action-2-param-keys=”...”
pos-action-3=”...” pos-action-3-param-aliases=”...”
```

The actions are dispatched in the order they were declared. In ordering terms, an action with no number is considered to be action -1, and action zero is optional.

pos-localise

Replace the contents of this element with the localisation string whose key is specified in this attribute. Prior to replacement, perform regular substitution on the string to satisfy context references.

If the element also has a pos-localise-params attribute, parse the contents as regular parameters, and add the key-value pairs to context prior to substitution. This is useful for mapping regular context space into the localisation space.

pos-page

Declare a page to be associated with this element. The value of this attribute is assumed to be the page "key", by which it is identified in successive page-oriented operations.

The element bearing a "pos-page" attribute is required at the top of a page HTML fragment to identify it as such, and will not be honoured in normal markup.

pos-view

Declare a view to be associated with this element. The value of this attribute is assumed to be the view "key", by which it is identified in successive view-oriented operations. In addition, "load flags" may accompany the view key, specifying whether Positron should load HTML, CSS, and Javascript for the view -

```
<!-- attempt to load everything for this view (the default) →
<!-- these lines are equivalent -->
<div pos-view=”menu”></div>
<div pos-view=”menu:chj”></div>

<!-- just load html for this view, and not css or javascript -->
<div pos-view=”menu:h”></div>

<!-- don’t load anything for this view (assumes HTML is inline) -->
<div pos-view=”menu:”>here is the inline markup</div>
```

When this attribute is encountered, Positron will attempt to instantiate a class to be associated with its element. The class name can be specified in full in the attribute, but if a direct instantiation fails, and the load flags allow, an attempt will be made to load a Javascript file using the method for locating view Javascript found in the configuration. The default path for this is "views/$view;/$view;.js". If this also fails, then an error message is written to the console and a vanilla View class is instantiated as a last resort.

Once a View class is associated with an element, it can be referenced by its key in view-related operations such as showview and hideview, and it will receive various lifecycle and visibility callbacks at appropriate times. Please see the Positron Developers Guide for more details.

pos-view-key

Override the view key specified in the pos-view attribute. Necessary when multiple views of the same type are extant within the same page, and it is necessary to distinguish between them.

Note that this is checked manually by the View attribute, it is not handled by an attributelet, as such.

pos-view-params

Provide initial parameters for a view registered off this element. The format is the standard Positron parameter list: semicolon delimited list of colon delimited name-value pairs.

Note that this is checked manually by the View attribute, it is not handled by an attributelet, as such.

pos-view-param-keys

Provide initial parameters for a view registered off this element. The format is the standard Positron parameter list: semicolon delimited list of colon delimited name-value pairs, but the values in this case denote keys into context. This is useful for passing non-primitive parameters around in markup.

Note that this is checked manually by the View attribute, it is not handled by an attributelet, as such.

Events

ajaxformsuccess

Insert the “data” property of the event detail into action parameters as “data”. Essentially this represents the object tree resulting from the parsing of the return from the Ajax form submission performed by *ajaxformaction*. Ideally, applications would register their own event types in invocations of *ajaxformaction* and provide eventlets to more intelligently map the results.

change

Assumed to be handling a change event dispatched from a form element. Determine the value of the form element and copy the value to action parameters. In this way, for example, the value of a <select> can be immediately dealt with in a target view, with no Javascript.

```
<form>
  <select name=”type” pos-action=”(change) refreshview: type”>
    <option value=”stout”>Stout Beer</option>
    <option value=”pilsner”>Pilsner Beer</option>
  </select>
</form>

<div pos-view=”type:”>
  <div>beer type is $params.type;</div>
</div>
```

devicemotion

Insert the “acceleration”, “accelerationIncludingGravity”, and “rotationRate” properties of the event record into action parameters.

deviceorientation

Insert the “alpha”, “beta”, “gamma”, and “absolute” properties of the event record into action parameters.

deviceproximity

Insert the “max”, “min”, and “value” properties of the event record into action parameters.

error

Insert the “error” property of the event detail record into action parameters as “error”. Essentially this is a generic eventlet which handles any error event by inserting the error object into the action’s parameters. Again ideally applications would handle this situation more appropriately for their event structure.

formdispatch

Insert all properties from the event detail record into action parameters.

userproximity

Insert the “near” property of the event record into action parameters.

Tags

Note regarding names of introduced context variables

Tags adding data to context all share one thing in common -- an optional “name” attribute which determines the name or initial qualifier for data so added. Best to illustrate with an example -

```
<!-- no “name” attribute supplied →
<!-- the name is tag name minus the configured tag prefix →
<pos-json url=”/session/user”>
  The logged-in user’s name is $json.name;
</pos-json>

<!-- “name” attribute supplied, so the name is its value →
<pos-json url=”/session/user” name=”user”>
```

```
  The logged-in user’s name is $user.name;
</pos-json>
```

In the following descriptions, the value NAME means the tag’s output “name”, either from the explicit name attribute, or derived from the tag name.

Note regarding composite element selection

Some tags, for example “move”, and “sync”, allow their destination elements to be specified from a combination of their “view” and “selector” attributes.

If the “view” attribute is specified, then the element associated with the view identified by the key in the attribute is used as a base element for any further selection. If there is no “view” attribute, then “document” is used as the base element. If the “selector” attribute is specified, then it is used to select the element, off the base element, using CSS selector syntax.

changecase
capitalcase
lowercase
uppercase

Change the case of a string.

These tags are all implemented by the ChangeCase taglet, the mode being determined either by the name of the tag, or an explicit “mode” attribute.

Options for the “mode” attribute are “capital”, “upper”, and “lower”. This tag expects its argument in its “string” attribute, and puts the resulting string into context as NAME.

action

Immediately perform the sequence of actions described in the attributes of this tag (eg “pos-action-1”, etc). Currently under review.

ajax

Superclass for tags wishing to add resources from the network to context, eg <pos-json>. Not directly used.

comment

Comments template markup and does not appear in final markup. It is an attempt to reduce the clutter that results from putting regular HTML comments in template markup, particularly in repeated blocks.

condition

Superclass for tags wishing to conditionally include their subtree. Not directly used.

date

Create a date object from the milliseconds value in the specified “ms” attribute, or the human readable format in the specified “string” attribute, or if neither are present, “now”, and put its discrete fields into context --

NAME.year - four-digit year
NAME.month - month number, 1-based
NAME.month0 - month number, zero-based
NAME.day - day of month, 1-based
NAME.hours - hour of day, 0-23
NAME.hours12 - hour of day, 0-11
NAME.hours24 - hour of day, 0-23
NAME.minutes - minutes, 0-59
NAME.seconds - seconds, 0-59
NAME.milliseconds - milliseconds, 0-999
NAME.ms - milliseconds since midnight Jan 1, 1970
NAME.string - date as resulting from the Javascript Date.toString() method

distance

Determine the distance between the geolocations specified in the “lat1”, “lon1”, “lat2”, and “lon2” attributes, and put an object with that distance in various units into context --

NAME.m - metres
NAME.km - kilometres
NAME.mi - miles
NAME.yards - yards
NAME.feet - feet

get

Fetch the value from context specified by the “key” attribute and put it into context as NAME. Useful for accessing context items whose key has to be derived.

if

Conditionally include the tag’s children depending on the value of attributes.

If the “empty” attribute is specified, then the children are included if the its value is zero length.

If the “notempty” attribute is specified, then the children are included if the its value is not zero length.

If the “true” attribute is specified, then the tag’s children are included if the value evaluates to true or false. Expression syntax is a subset of Javascript - note that eval() is *not* used on this string, for security and clarity reasons.

Allowed expressions are -

x == y
x > y
x >= y
x < y
x <= y

Note that terms in expressions which are deemed to be numeric are compared numerically, otherwise they are compared as strings.

Any number of expressions may be chained, using “and”, “&&”, “or”, or “||”. Note that the last chaining logical operator binds, so the evaluator may not make the decision the developer expects if “and” and “or” are combined.

There is currently no provided support for subexpressions.

It is recommended that developers not code complex expressions in markup, rather authoring custom tags to aid readability.

indexedquery

Query the store in the database specified by the “database” and “store” attributes, inserting a list of results into context as NAME. If “index” and “search” attributes are provided, search on the specified index for the specified value.

This tag uses the HTML5 IndexedDB API. Positron ships with a shim to adapt the IndexedDB API to WebSQL for browsers which require it.

Bugs: searching is not currently supported

isolator

Impersonate another tag until the tree walker arrives. Browsers will not allow certain tag combinations to occur and will rewrite the DOM in response. This can break dynamics processing.

An example would be this dynamic popup menu -

```
<select>
  <pos-list key=”recipients” name=”recipient”>
    <option value=”$recipient.id;”>$recipient.name;</option>
  </pos-list>
</select>
```

Most browsers will not allow tags between <select> and <option>.

Another situation where isolation is required is when the browser encounters a tag with a URL as a source attribute before any dynamics references in the attribute have been satisfied -

```
<img src=”$user.profile_pic_url;”>
```

- which would result in an error message in the console.

The solution is to associate a derived tag name with the IsolatorTag class. The default configuration does this to provide isolation for <audio>, <img>, <option>, <select>, <table>, <td>, <tr>, and <video>. Use <pos-audio>, <pos-img>, etc, instead to delay evaluation until the tree is complete.

join

Retrieve a Javascript array from context with the name provided in the “key” attribute, join its elements together using the delimiter in the “delimiter” attribute, if provided, or space, if not. Add the resulting string to context as NAME.

json

Load JSON-formatted content from the URL provided in the “url” attribute, parse it to produce a Javascript object tree, and put the tree into context as NAME.

If the attribute “jsonp” is set to “true”, then JSONP is used to access the URL.

This tag is asynchronous.

list

Retrieve a Javascript array from context with the name provided in the “key” attribute, and traverse it, adding a copy of its children for each element in the list.

If an “offset” attribute is provided, then traversal of the list starts at that offset.

If a “limit” attribute is provided, then at most, that number of elements are traversed.

If “searchkey” and “searchvalue” attributes are provided, and the array to be traversed contains objects, then only those objects whose property names and values match those in the search key and value will be included.

For each matching element in the list, it is added to context for the traversal of its copy of the tag’s children, under NAME.

In addition, several times of metadata are also added to context -

NAME.meta.globalindex - absolute index within the list

NAME.meta.globalcount - length of the list

NAME.meta.index - index within the sub-list, honouring offset, limit, and search
NAME.meta.count - length of the sub-list, honouring offset, limit, and search
NAME.meta.isfirst - true if the element is the first in the sub-list, otherwise false
NAME.meta.islast - true if the element is the last in the sub-list, otherwise false

Example -

```
<pos-json url=”/users/all” name=”users”>
  <pos-list key=”users” name=”user”>
    <div>User $user.meta.index;’s name is $user.name;</div>
  </pos-list>
</pos-json>
```

localstorage

Get the values of the keys provided in “key” (and “key-0”, “key-1”, etc) attributes and place into context as NAME.key.

location

Get the current GPS location and add it to context as NAME. Note that unlike the location trigger, this is one-shot only.

log

Write a message to the console, the value of which is either the value of the “value” attribute, the value within context of the key contained within the “valuekey” attribute, or, if neither attribute exists, the current context.

map

Retrieve a Javascript object from context with the name provided in the “key” attribute, and traverse it.

If a “limit” attribute is provided, then at most, that number of elements are traversed.

For each matching property in the list, its key and value are added to context under NAME.key and NAME.value.

In addition, several times of metadata are also added to context -

NAME.meta.index - index within the map, honouring offset, limit, and search
NAME.meta.count - length of the map, honouring offset, limit, and search
NAME.meta.isfirst - true if the element is the first in the map, otherwise false
NAME.meta.islast - true if the element is the last in the map, otherwise false

Example -

```
<pos-json url="/session/user" name="user">
  <div>Properties of the logged-in user object are:</div>
  <pos-map key="user" name="userproperty">
    <div>$userproperty.key; = $userproperty.value;</div>
  </pos-map>
</pos-json>
```

Bugs: does not currently walk its subtrees properly

move

Move all this tag's children to another element. The destination element is specified by a combination of the "view" and "selector" attributes, as described above.

numberformat

Format the numeric value specified in the "number" attribute according to the parameters in the "type" and "digits" attributes. Options for the "type" attribute are --

"fixed" - format to the number of decimal places given in the "digits" attribute

"precision" - format to the number of total digits given in the "digits" attribute

"frontpad" - add zeroes to the front of the number until the number of digits equals the "digits" attribute.

"floor" - round the number down to the nearest integer.

"ceil" - round the number up to the nearest integer.

"round" - round the number to the nearest integer.

The formatted number is then put into context as NAME.

Bugs: The formatting options are inadequate.

prefixedproperty

Map the browser neutral CSS property supplied in the "property" attribute to browser-specific space, adding it to context as NAME.

replace

This tag does simple string replacement on the string specified in the "string" attribute, replacing occurrences of the string specified in the "replace" attribute with the string specified in the "with" attribute, and adding the resulting string to context as NAME.

If the “regexp” attribute is “true”, then the string in “replace” is taken to be a regular expression.

If the “all” attribute is “true”, then all occurrences are replaced.

selectoption

Select the <option> tag within this tag whose values matches the value of this tag’s “value” attribute. Used to save a lot of if/then when setting the value of a <select> to an existing value.

set

Set a value in context. The value is specified either in “value”, “valuekey”, or “expression”. For strings and primitive values, “value” is sufficient, but if the value is an object or array, etc, then “valuekey” is used to give the key of the value.

If “expression” is specified, then its value is evaluated as a simple arithmetic expression. Allowed operands are --

+ add last term to sum
- subtract last term from sum
* multiply sum by last term
/ divide sum by last term
% remainder of sum divided by last term

Note that terms and operands must be space delimited, and precedence and subexpressions are not supported.

The resulting value is added to context as NAME.

Set also accepts a “context” attribute, which can be used to “promote” context variables to a context higher up in the DOM, so that the value can be accessed outside of the default tag context. Values accepted for “context” are “application”, “page”, and “view”.

Bugs: terms should not have to be space delimited.

split

Divide the string provided in the “string” attribute into list elements delimited by the value of the “delimiter” attribute, if provided, or space if not provided. Add the resulting array of elements to context as NAME.

If the “regexp” attribute is “true”, then the “delimiter” attribute is interpreted as a regular expression.

sqlquery

Submit the SQL query specified by the "query" attribute with the arguments in the comma-separated "parameters" attribute to the web database identified by the "database" attribute, putting result records into context as NAME.

Bugs: This tag uses the WebSQL API, which is officially obsolete. For best browser compatibility, use *indexedquery* instead.

sync

Synchronise all this tag's children with those of another element. The destination element is specified by a combination of the "view" and "selector" attributes, as described above.

Children of this tag should have "id" attributes uniquely identifying them in the DOM, otherwise they will be ignored. If there is a child in the destination element with the same ID as a child in this tag, then its contents will be replaced and its attributes copied over. If there is no child in the destination element with the same ID, then it is inserted. If there is a child in the destination element whose ID is not represented in the children of this tag, then the child is removed.

This tag is generally used to update rapidly changing displays without the flicker that results from the rebuilding of the markup that accompanies a regular view refresh.

timeago

Calculate the time elapsed between "now" and the time specified in either the "ms" (milliseconds) or "s" (seconds) attribute. Convert this value to the largest non-zero time unit and put into context as NAME.units and NAME.value. Currently available time units are years, months, weeks, days, hours, minutes, and seconds.

Bugs: should continue to break down the elapsed time into smaller units, rather than give up when the largest non-zero time unit has been found.

Triggers

animationframe

Fire when requestAnimationFrame() calls back, with a frequency in frames per second specified in the trigger argument. If this is not specified, the default of 33fps is assumed.

circle

Fire associated action periodically with a set of x, y coordinates for sequential points on a circle. The first trigger argument gives the interval in degrees, with the default 5, while the second argument gives the interval in milliseconds, with the default 10.

click

tap

Fire associated action when a single click occurs on the associated element. A single click is defined as a mouse or touch down and up without any movement.

Note that this is actually implemented as a custom trigger, due to the suboptimal performance of the default click algorithm on some touch devices.

Note that click and tap are functionally identical and point to the same trigger class. The implementation automatically detects whether the platform is desktop or mobile and registers for events appropriately.

Bugs: should allow configurable distance slop.

defer

Fire associated action when the current refresh completes. Given that some triggers have to be deferred in order to prevent their actions firing at inopportune times, it is often helpful to defer actions similarly. For example, the web socket message trigger is deferred, so it is advised to also defer anything sending on that web socket. Deferred operations are performed in the order declared.

delay

Fire associated action when the delay elapses. The period is determined by the value of the first argument, which can be any “time” string (eg 2000ms, 4.5s, etc), or 1 second if there are no arguments or the arguments do not parse.

doubleclick
doubletap

Fire associated action when a double click occurs on the associated element. A double click is defined as two distinct mouse clicks or taps whose “down” or “start” events happen within the required time.

The maximum time required to qualify is 500ms by default, and configurable via the first trigger parameter. The maximum “slop”, or distance between the clicks, is 20 pixels by default, and configurable via the second trigger parameter.

The following examples show configuration usage -

```
<div pos-action=”(doubleclick: 1s/100) showview: doubleclicked”>
  Double click within 1 second, 100 pixel slop.
</div>

<div pos-action=”(doubleclick: 750ms/50) showview: doubleclicked”>
  Double click within 750 milliseconds, 50 pixel slop.
</div>
```

interval

Fire associated action periodically. The period is determined by the value of the first argument, which can be any “time” string (eg 2000ms, 4.5s, etc), or 1 second if there are no arguments or the arguments do not parse.

keydown

Fire associated action when a keydown event is caught. If a keycode or key symbol name is passed as a trigger parameter, restrict to a keydown event for that key. Otherwise, fire on all keydown events.

keypress

Fire associated action when a keypress event is caught. If a keycode or key symbol name is passed as a trigger parameter, restrict to a keypress event for that key. Otherwise, fire on all keypress events.

location

Fire associated action when the HTML5 location API determines that the user’s location has changed. When the action is fired, the position is added to the action’s parameters.

longclick
longtap

Fire associated action when a mouse or touch down and hold occurs on the associated element.

The minimum time required to qualify as a long click or tap is 2 seconds by default, and configurable via the first trigger parameter. The value can be specified as a valid time span. The following examples configures long clicks for 5 seconds -

```
<div pos-action=”(longclick: 5s) showview: longclicked”>
  Long click
</div>

<div pos-action=”(longtap: 5000ms) showview: longclicked”>
  Long click
</div>
```

Note that longclick and longtap are functionally identical and point to the same trigger class. The implementation automatically detects whether the platform is desktop or mobile and registers for events appropriately.

mousedown

Fire associated action when the mouse goes down or a touch starts.

mousemove

Fire associated action when the mouse or touch moves.

mouseup

Fire associated action when the mouse goes up or the touch ends.

now

Fire associated action immediately. This functionality is duplicated by <pos-action> and this trigger is therefore under review and may be deprecated.

prefixedevent

Fire associated action when the appropriate prefixed event is caught. The event name specified in the first argument is mapped to a browser-specific (prefixed) event and a listener installed for it. If the event name specified does not map to a prefixed event name, then the listener is installed on the unmapped event name.

websocketmessage

Fire associated action when a message appears on the web socket named in the trigger arguments. The message itself is parsed from JSON and inserted into the action’s parameters as “message”.