NGSP Container – the building bricks of the engine

0.Definition

A NGPS container is a HTML Dom element with a set of functions and

properties that facilitate the engine functionality.

# **UID** – is a unique number identifying the container

*#* ***DOMreference*** – is the property that points the actual HTML DOM object. Use this to access the well

know HTML properties (style, onLoad).

A container can be turned into more a more specialized object (a link, an app, a camera).

These properties indicate what a container is:

# ***isLeaf*** *– is a terminal node*

# ***isApp*** *– has an application loaded onto the container*

# ***isLink*** *– is an actual link between two containers*

# ***isCamera*** *– is a camera object ( contains a child called* ***display*** *where the inner content is found )*

What makes a container?

1. Descriptor – describes how to construct a container
2. Extensions – turning a container into something with more functionality: camera, app

extend(object) extends the functionality of a container.

- extend(Camera) will turn it into a camera

1. Permissions – describes what is allowed to happen to the container
2. Functionality – methods that operate on the container
3. Hi-Performance Callbacks – functions that are called when certain events happen to the container
4. Events – possible events that can be listened for on the container

0.1. Container Hierarchy

Containers are arranged in a tree structure:

container: { UID, parent, children };

parent : a direct reference to it’s parent

children: {UID1: child1, UID2:child2 ,UID3:child3, … }

*The very first container has a NULL parent reference* and the actual HTML parent of that container will be **document.body.**

Note: the Factory framework will take care of the setup (depending on the chosen mode), it will create the first objects needed to make the infrastructure valid.

1. Descriptors – create or redefine containers

To create a container:

var c = new ***Container***(descriptor); //the descriptor specifies initial properties, permissions and styles

c.***load***(parent); //then you need to load the container on a parent ( another NGPS container );

Descriptor syntax: { key:value , … }

Check descriptor annex to see what properties you can set.

c.***restyle***(descriptor); //change the style of an already existing container

1. Extensions – make a container something more

***extend()*** – adds properties and functions to the container

c.**extend**(Camera); //will turn your container into a camera

c.extend(App); //will enable the container to host an all

c.loadApp(appName); //will load the app onto the container

1. Permissions – limit what can happen to a container

c.**permissions** = { key:value ,… } – a map of permissions key – permission , value – action Check permissions annex.

1. Functionality – the functions you can apply to change containers

Check functions annex.

1. Hi-Performance Callbacks

When certain events happen to an event the situation needs to be handled quickly. For example when the user drags their thumb over the container (in this case the container needs to move with the thumb)

The way the system handles these situations can be overridden using these callbacks.

Example callbacks: *onMoved, onRotated, OnZoomed*

Check callback annex to see all of them.

1. Events

The event system allows you to listen for a variety of events on a container. One event can trigger multiple listeners while the callback can only call one function (this makes it faster but less flexible).

c.**addEventListener**(eventName,function); //listen for an event on container c

c.**removeEventListener**(eventName,function); //stop listening for an event on container c

Check the events annex to see all container events

Descriptor Annex

|  |  |  |
| --- | --- | --- |
| Property | Value Type | Description |
| type | string ( div, img, svg, … )  default = div | specifies what type of HTML DOM object the container will be. |
| width | number - pixels, string (percentage ‘100%’) | specifies the width of the container |
| height | same as above | specifies the height of the container |
| x | same as above | specifies x coordinate of container within parent |
| y | same sa above | specifies y coordinate of container within parent |
| class | String | sets the CSS class name for the object |
| style | String | sets the raw style of the HTML DOM object |
| cssText | String | same as above |
| autopos | Boolean | tells the container it will use automatic positioning for alignment ( the position will not be set to absolute ) |
| autosize | Boolean | the overflow will not be hidden and the size of the container will scale to the content |
| top, bottom, left, right | string | set the HTML DOM respective properties overriding any style or class properties |
| background | string | sets the style background property of the container, overrides previous style or class properties |
| opacity | string | sets the style opacity property, overrides the previous style & class |
| border\_size | string, int | size of border in pixels or percentage |
| border\_color | string | style property for border |
| border\_type | string (solid, dashed, etc) | type of border |
| border\_radius | vector, for each corner  [“10px”,”5px”,”10px”,”15px”]  UpL, UpR, LowR, LowL  [“10px”] –sets all corners to the same |  |

Permissions Annex

|  |  |  |
| --- | --- | --- |
| Permission | Default | Description |
| save | true | controls if this container can be saved or not |
| connect | false | controls if this container can be connected with another |
| children | true | controls if container can have children appended to it |
| edit | false | controls if container can be edited |

Functions Annex

|  |  |
| --- | --- |
| load(<parent>) | creates the HTML object representing the container, without calling load the container is not functional and not visible |
| extend( extensions ) | extends a container’s functionality – is used to turn a container into Camera / App and to make container interactive. It adds the functions and members of the extensions object to the current container. |
| strip ( extensions ) |  |
| addChild( properties ) | adds a child container to the current one |
| removeChild( UID ) | removes a child container by UID |
| discard( ) | deletes the container ( cleans everything up – links, events, children ). The call is bubbled through to all children. |
| changeParent( parent ) | changes the parent of the container |
| addPrimitive( descriptor ) | adds a primitive HTML object inside the current container. Used to load images, videos, text, websites inside the current container.  Example of primitive objects ( any HTML tag ):  <img></img> or <iframe></iframe> |
| removePrimitive( ) | removes a primitive HTML object |
| show( ) |  |
| hide( ) |  |
| getPos( cx, cy, <refX>, <refY> ) | Gets the position of the container. refX and refY are where the point represented by the position will be on the container.  example:  .getPos(0,0) – upper left corner position  .getPos(0.5,0.5) – center of the container  .getPos(1,1) – lower right corner |
| getWidth() |  |
| getHeight() |  |
| getPureWidth( ) | - does not work |
| getPureHeight( ) | - does not work |
| local2global(<refX>,<refY>,<stopAt>) | gets the position of the current container on the root object or at the container specified by stopAt. |
| global2local(x,y); | converts root object coordinates to local container coordinates |
| setWidth( width ) |  |
| setHeight( height ) |  |
| setAngle( angle, <ox>, <oy> ) | Set the angle of the container. The center of rotation is specified by refX and refY in percentages  0,0 – upper left  0.5,0.5 – center  1,1 – bottom right |
| putAt( x, y, <refX>, <refY> ) |  |
| move( dx, dy, <noevent> ) |  |
| scale( amount, <ox>, <oy>) |  |
| enlarge( amount ) |  |
| rotate( delta\_angle, <ox>, <oy> ) |  |
| getAncestors( node ) |  |
| greatestCommonParent( target ) | Finds the greatest (deepest) parent that the current container and the target one have in common |
| link( target, descriptor ) | Creates a outgoing link between the current container and the target. |
| unlink( target ) | Deletes an outgoing link between the current container and the target |
| unlinkAll() | Deletes all links that are connected to the current container |
| changeLinkTarget( oldTarget, newTarget) | change the target of an outgoing link |
| maintainLink( target ) | this function updates the link position, size and angle in case any of the two connected containers changes in any way ( width, height, position ) |
| maintainLinks() | maintains all the links of a container |
| addEventListener(event, handler, <context>) |  |
| removeEventListener( event, handler, <context>) |  |
| loadApp(appName) | loads an application onto the current container |

Callback Annex

|  |  |  |
| --- | --- | --- |
| Name | Parameters received | Override effect to system |
| onMouseDown | container, mouse\_event\_object | none |
| onMouseUp | container, mouse\_event\_object | none |
| onMoved | delta x, delta y, container | The system no longer moves the container, the callback function must deal with that. For example if you want to make the container unmovable by mouse/touch interaction override this callback with an empty function  c.onMoved = function(){}; |
| onTrigger | container, mouse\_event\_object | none |
| onRotated | delta angle | Same as for onMoved |
| onZoomed | delta scale | Same as for onMoved |

* delta = difference between current and previous value
* Callbacks are much faster than event listeners but have the disadvantage of not being able to call multiple functions when the event occurs, therefore it is preferable to use GEM events when building apps

Events Annex

|  |  |
| --- | --- |
| loadContainer |  |
| addChild |  |
| removeChild |  |
| discardContainer |  |
| hideContainer |  |
| showContainer |  |
| changeParent |  |
| changeWidth |  |
| changeHeight |  |
| changeAngle |  |
| changePosition |  |
| link |  |
| unlink |  |
| linkChange |  |
| appLoaded |  |