BaseObject

AI

	_classHierarchy[]	= [BaseObject]
	_notificationListeners{}	
	_super{}	
	_tags{}	
	_tagListeners{}	
	_traceNotifications	= false
r/o	.class	
r/o	.superClass	
r/w	.tag	

- _autoInit([args])
- addListenerForNotification(notification, fn)
- addTagListenerForKey(key, fn)
- defineProperty(name [, options])
- defineObservableProperty(name [, options])
- destroy()
- getClass()
- getSuperClassOfClass (class)
- getTag()
- getTagForKey(key)
- init()
- notify(notification, [args])
- notifyMostRecent(notification [, [args]])
- override(namedFn)
- overrideSuper(class, fnName, fn)
- registerNotification(notification)
- removeLIstenerForNotification(notification, fn)
- removeTagListenerForKey (key, fn)
- setTag(value)
- setTagForKey(key, value)
- subclass(newClass)
- super(class, fnName[, [args]])

⬉ (none)

r/w	read-write	r/o	read-only	.	property
()	method	o	override	a	abstract
p	private			⬉	notification

Defining classes:

```
var _className = “ANewClass”;
function ANewClass = function () {
  var self = new _y.BaseObject();
  self.subclass (_className);
  // define properties
  // define methods
  self._autoInit.apply(self, arguments)
  return self;
}
return ANewClass;
```

Overriding methods:

```
self.override( function init(arg, …) {
  self.super (_className, “init”, arguments );
  // override code
}
```

OR

```
self.overrideSuper ( _className, “init”, self.init );
self.init = function ( arg, …) {
  self.super (_className, “init”, arguments );
  // override code
}
```

Initializing

```
var x = new _y.BaseObject();
x.init();

var x = (new _y.BaseObject()).init();

// for auto-initializable objects only
// marked with AI in model
var x = new _y.BaseObject( arg [, …] );
```

ViewContainer : BaseObject

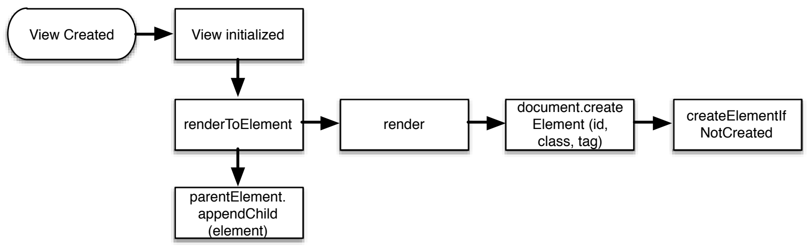
AI

	_element	= null
	_elementClass	= ui-container
	_elementId	= null
	_elementTag	= div
	_parentElement	= null
t	.navigationController (only present when child of NavigationControllers)	
t	.splitViewController (only present when child of SplitViewControllers)	
t	.tabViewController (only present when child of TabViewControllers)	
r/w	.element	
r/w	.elementClass	
r/w	.elementId	
r/w	.elementTag	
r/w	.parentElement	
r/w	.title	
r/o	.superClass	
r/w	.tag	
	createElement()	
	createElementIfNotCreated()	
o	destroy()	
	getElement()	
	getElementClass()	
	getElementId()	
	getElementTag()	
	getParentElement()	
o	init([ID], [tag], [className], [DomElement])	
	initWithOptions (options)	
a	render()	
	renderToElement()	
	setElement(DomElement)	
	setElementClass (className)	
	setElementId (ID)	
	setElementTag (tagName)	
	setParentElement (DomElement)	
↖	viewWasPushed	
↖	viewWasPopped	
↖	viewWillAppear	
↖	viewWillDisappear	
↖	viewDidAppear	
↖	viewDidDisappear	

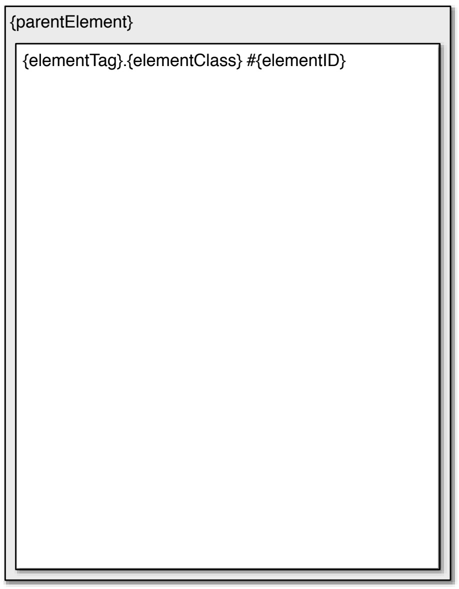
r/w	read-write	r/o	read-only	.	property
()	method	o	override	a	abstract
p	private	t	transient	↖	notification

Initializing

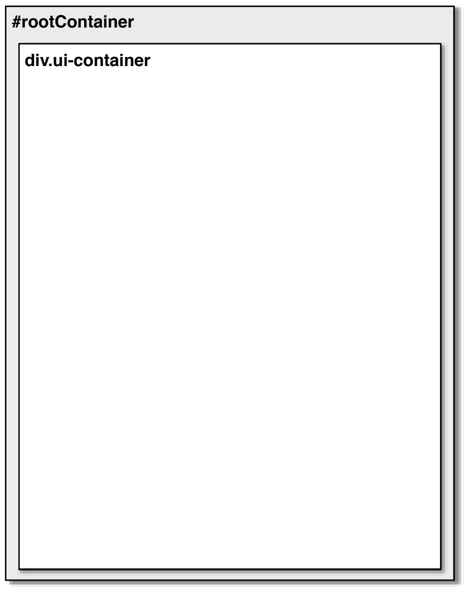
```
var aView = new _y.UI.ViewContainer();
aView.init(); // default class and tag
aView.init( “view1” ); // view now has ID of view1
                       // in the DOM

// view uses <article> instead of <div>
aView.init( undefined, “article” );

// view puts “my-class” on the DOM element
aView.init( undefined, undefined, “my-class” );

// view indicates its parent element
aView.init( undefined, undefined, undefined,
  document.getElementById(“rootContainer”) );

// using initWithOptions is easier:
aView.init( { id: “…”, class: “…”,
              tag: “…”, parent: “…” } );

// or, use auto-initialization:
var aView = new _y.UI.viewContainer( {
  id: “…”, class: “…”, tag: “…”,
  parent: “…” } );
```


View Created
View initialized
renderToElement
render
document.create Element (id, class, tag)
createElementIf NotCreated
parentElement. appendChild (element)

DOM Element Hierarchy



Typical Example



NavigationController : ViewContainer

AI

	_preventClicks	= null
	_subviews[]	
t	.navigationController (only present when child of NavigationControllers)	
t	.splitViewController (only present when child of SplitViewControllers)	
t	.tabViewController (only present when child of TabViewControllers)	
r/w	.rootView	
r/o	.topView	
	_createClickPreventionElement()	
	_createClickPreventionElementIfNotCreated()	
	getRootView()	
	getTopView()	
o	destroy()	
o	init(rootView, [ID], [tag], [className], [DomElement])	
o	initWithOptions (options)	
	popView ([withAnimation], [delay], [animationType])	
	pushView (aView, [withAnimation], [delay], [animationType])	
o	render()	
o	renderToElement()	
	setRootView(aView)	
⬉	viewPushed	
⬉	viewPopped	
⬉	modalViewPushed	(not implemented)

r/w	read-write	r/o	read-only	.	property
()	method	o	override	a	abstract
p	private	t	transient	⬉	notification

Initializing

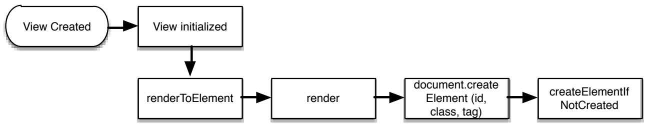
```
var aView = new _y.UI.ViewContainer( {
  className = “ui-container my-first-view” } );
var nc = new _y.UI.NavigationController( {
  parent: document.getElementById(“rootContainer”),
  rootView: aView } );
```

Pushing a View

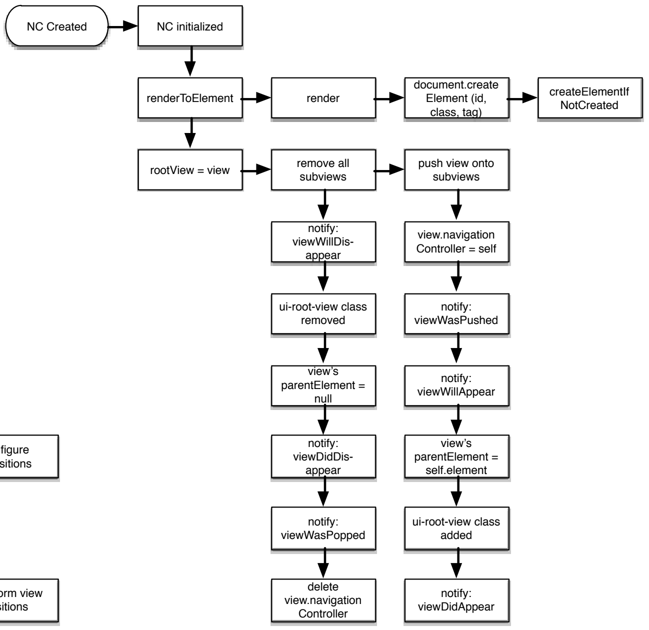
```
var anotherView = new _y.UI.ViewContainer( {
  className = “ui-container my-second-view” } );
nc.pushView ( anotherView );
```

Popping a View

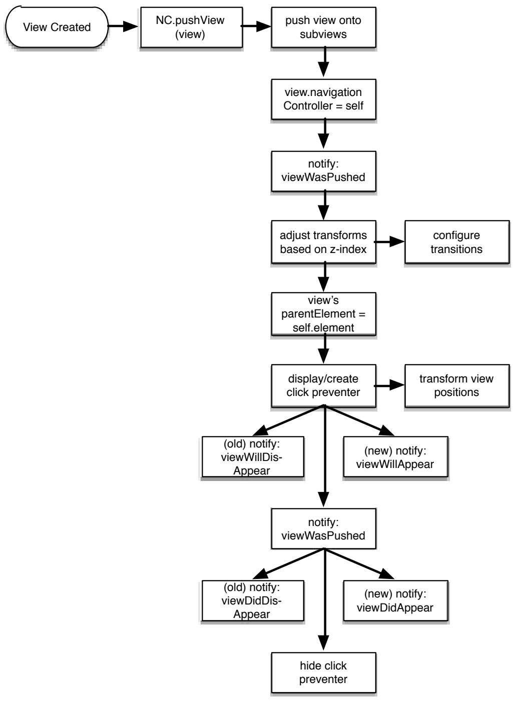
```
nc.popView();
```



View Created
View initialized
renderToElement
render
document.create Element (id, class, tag)
createElementIf NotCreated

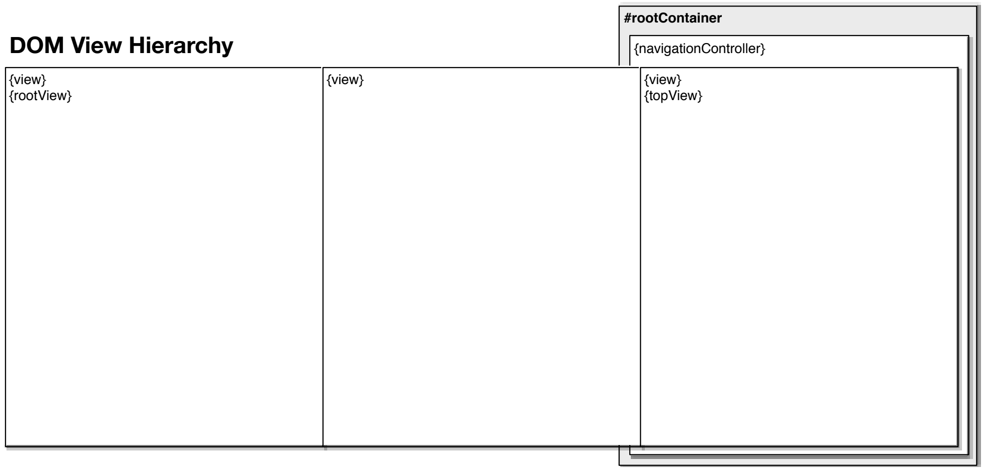


NC Created
NC initialized
renderToElement
render
document.create Element (id, class, tag)
createElementIf NotCreated
rootView = view
remove all subviews
push view onto subviews
notify: viewWillDis-appear
view.navigation Controller = self
ui-root-view class removed
notify: viewWasPushed
view's parentElement = null
notify: viewWillAppear
notify: viewDidDis-appear
view's parentElement = self.element
notify: viewWasPopped
ui-root-view class added
delete view.navigation Controller
notify: viewDidAppear



View Created
NC.pushView (view)
push view onto subviews
view.navigation Controller = self
notify: viewWasPushed
adjust transforms based on z-index
configure transitions
view's parentElement = self.element
display/create click preventer
transform view positions
(old) notify: viewWillDis-Appear
(new) notify: viewWillAppear
notify: viewWasPushed
(old) notify: viewDidDis-Appear
(new) notify: viewDidAppear
hide click preventer

DOM View Hierarchy

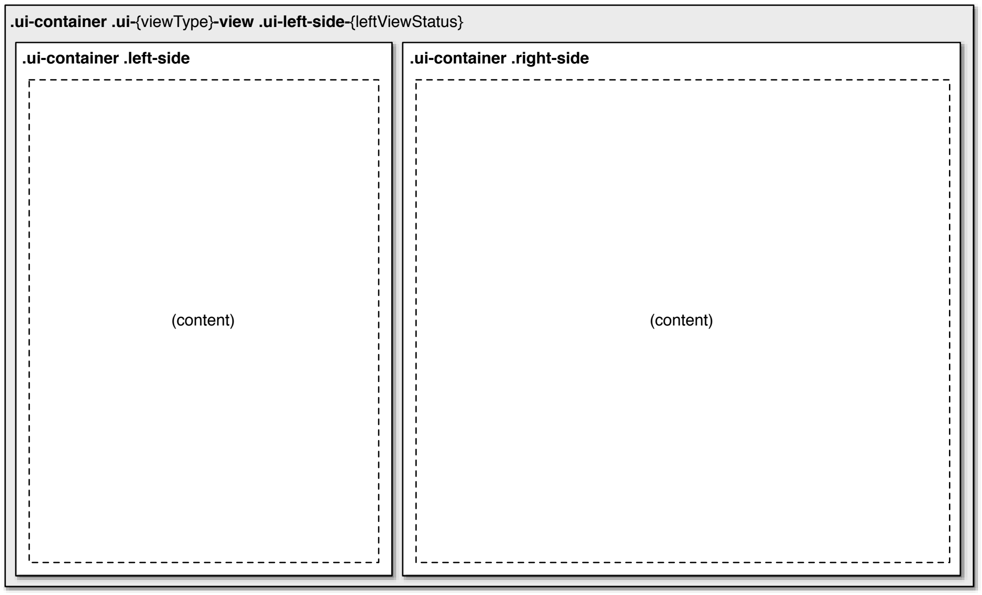


SplitViewController : ViewContainer

	_subviews[]
r/w	.leftView
r/w	.leftViewStatus = `invisible` (`invisible`, `visible`)
r/w	.rightView
r/o	.subviews[left, right]
r/w	.viewType = `split` (`split`, `off-canvas`, `split-overlay`)
t	.navigationController (only present when child of NavigationControllers)
t	.splitViewController (only present when child of SplitViewControllers)
t	.tabViewController (only present when child of TabViewControllers)
	_assignViewToSide(side, view)
	_createElements
	_createElementsIfNecessary
o	destroy()
	getLeftView()
	getLeftViewStatus()
	getRightView()
	getSubviews()
	getViewType()
o	init(theLeftView, theRightView, [ID], [tag], [class], [parent])
o	initWithOptions(options)
o	render()
o	renderToElement()
	setLeftView(view)
	setLeftViewStatus (status)
	setRightView(view)
	setViewType(viewType)
	toggleLeftView()
⬉	viewsChanged

r/w	read-write	r/o	read-only	.	property
()	method	o	override	a	abstract
p	private	t	transient	⬉	notification

SplitViewController DOM Layout



.ui-container .ui-{viewType}-view .ui-left-side-{leftViewStatus}
.ui-container .left-side
(content)
.ui-container .right-side
(content)