NGClient specific servoy plugins

When developing a solution for NGClient, Servoy provides a plugin for interacting with the browser.

That plugin is called ngclientutils (plugins.ngclientutils.*)

It has the following functions:

Function	Documentation
getUserAgent	This will return the user agent string of the clients browser
setOnUnloadConfirmation (Boolean showConfirmation)	Set whether browser default warning message will be shown when the browser tab is closed or the users navigates away, this can be used to let users know they have data modifications that are not yet saved.
setViewportMetaDefaultForMobileAwareSites()	Call this when a solution can handle mobile device layouts (responsive design, can handle nicely width < height). This call is equivalent to calling setViewportMetaForMobileAwareSites(plugins.htmlHeaders.VIEWPORT_MOBILE_DEFAULT). It should be what most solutions that are able layout correctly on smaller mobile screens need; it will still allow the user to zoom-in and zoom-out.
setViewportMetaForMobileAwareSites(viewportDefType)	Call this when a solution can handle mobile device layouts (responsive design, can handle nicely width < height). It will tell the device via the "viewport" meta header that it doesn't need to automatically zoom-out and use a big viewport to allow the page to display correctly as it would on a desktop. 'viewportDefType' can be one of: plugins.ngclientutils.VIEWPORT_MOBILE_DEFAULT - will show content correctly, allow zoom-in and zoom-out; the generated meta tag will be <meta name="viewport" content="width=device-width, initial-scale=1.0" /> plugins.ngclientutils.VIEWPORT_MOBILE_DENY_ZOOM - will show content correctly, denies zoom-in and zoom-out; the generated meta tag will be <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0" /> plugins.ngclientutils.VIEWPORT_MOBILE_DENY_ZOOM_OUT - will show content correctly, allows zoom-in but denies zoom-out; the generated meta tag will be <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0" /> plugins.ngclientutils.VIEWPORT_MOBILE_DENY_ZOOM_IN - will show content correctly, denies zoom-in but allows zoom-out; the generated meta tag will be <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" /> This method actually uses replaceHeaderTag. For example plugins.ngclientutils.VIEWPORT_MOBILE_DEFAULT would call replaceHeaderTag("meta", "name", "viewport", { * tagName: "meta", * attrs: [{ name: "name", value: "viewport" }, * { name: "content", value: "width=device-width, initial-scale=1.0" }] * });
replaceHeaderTag (tagName, attrNameToFind, attrValueToFind, newTag)	Utility method for manipulating 'contributedTags' array. It searches for an existing 'tag' that has the given 'tagName' and attribute ('attrNameToFind' & 'attrValueToFind'). If found it will replace it with 'newTag'. If not found it will just append 'newTag' to 'contributedTags'. NOTE: this call will only replace/remove tags that were added via this plugin/service, not others that were previously present in the DOM
addFormStyleClass (formname,styleclass)	Utility method for manipulating form style classes. It will add a style class to a certain form, similar as a design style class would work.
getFormStyleClass (formname)	Utility method for manipulating form style classes. It will get styleclasses assigned to a certain form, multiple styleclasses are separated by space. NOTE: this call will only get style classes that were added via this plugin/service, not others that were previously set at design time or via solution model.
removeFormStyleClass (formname,styleclass)	Utility method for manipulating form style classes. It will remove a styleclasse assigned to a certain form. NOTE: this call will only remove style classes that were added via this plugin/service, not others that were previously set at design time or via solution model.

scrollIntoView(anchorSelector, scrollIntoViewOptions)	Move the scrollbar to the position of the given anchorSelector. The target anchorSelector can be a Servoy Form, Layout Container or element in a responsive form or any element in a form. You can use styleClass as selector. For example: you can add 'scroll-element' to an element of the form. Examples of usage: - plugins.ngclientutils.scrollIntoView(".toScroll-To"); - plugins.ngclientutils.scrollIntoView(".toScroll-To", { behavior: "smooth", block: "start", inline: "nearest" }); @param anchorSelector {string} the selector to which the scrollbar should be moved to. @param scrollIntoViewOptions option argument used for scrolling animation (example: { behavior: "smooth", block: "start", inline: "nearest" }).
addClassToDOMElement (cssSelector, className)	Utility method for manipulating any DOM element's style classes. It will add the given class to the DOM element identified via the jQuery selector param. NOTE: This operation is not persistent; it executes client-side only; so for example when the browser is reloaded (F5 /Ctrl+F5) by the user classes added by this method are lost. If you need this to be persistent - you can do that directly via server side scripting elements.myelement.addStyleClass(...) if the DOM element is a Servoy component. If the DOM element is not a component then you probably lack something in terms of UI and you could build what you need as a new custom component or use another approach/set of components when building the UI. @param cssSelector {string} the css selector string that is used to find the DOM element. @param className {string} the class to be added to the element.
removeClassFromDOMElement(cssSelector, className)	Utility method for manipulating any DOM element's style classes. It will remove the given class from the DOM element identified via the jQuery selector param. NOTE: This operation is not persistent; it executes client-side only; so for example when the browser is reloaded (F5 /Ctrl+F5) by the user classes removed by this method are lost; If you need this to be persistent - you can do that directly via server side scripting elements.myelement.removeStyleClass (...) if the DOM element is a Servoy component. If the DOM element it is not a component then you probably lack something in terms of UI and you could build what you need as a new custom component or use another approach/set of components when building the UI. @param cssSelector {string} the css selector string that is used to find the DOM element. @param className {string} the class to be added to the element.