Dynamic Tag Management Object Reference

This is a basic object reference for _satellite

_satellite

The _satellite object is the essential DTM JavaScript object that is generated by the DTM libraries. All of the following items are part of the _satellite object.

$data: function (elm, prop, val)

LOCATION: core.js

USAGE: `$data(elm, prop, [val])`

PARAMETERS:

- `elm`: the element with a property to get or set
- `prop`: the property name
- `val`: the value of the property. If this parameter is omitted, the method will return the existing value of the property if any exists.

DESCRIPTION: _satellite `$data()` method, [similar to jQuery] (http://api.jquery.com/jQuery.data/), used to get or set properties on DOM elements in a reasonable manner.

`uuid` and `dataCache` are used by `$data()`

BaseTool: function (settings)

LOCATION: core.js

USAGE: `BaseTool(settings)`
Extended usage: * BaseTool.prototype

PARAMETERS:
- `settings`: initial tool configuration

DESCRIPTION: Core queuing mechanism used to handle tool settings and actions.

addEventHandler: function (node, evt, cb)

LOCATION: core.js

USAGE: `addEventHandler(elm, evt, cb)`

PARAMETERS:

- `elm`: the element to receive the event handler
- `evt`: the event type to listen to
- `cb`: callback function

DESCRIPTION: Register an event handler for an element

any: function (arr, cond, context)

LOCATION: core.js

USAGE: `any(arr, cond, context)`

PARAMETERS:
- `arr`: the array being evaluated
- `cond`: the condition
- `context`: valid context to execute the function

DESCRIPTION: An array helper function. Returns true if `cond(item)` returns true for any item in the array.

availableEventEmitters: Array[8]

LOCATION: core.js

USAGE: `availableEventEmitters`

DESCRIPTION: The available event emitters to use.

availableTools: Object

LOCATION: core.js

USAGE: `availableTools`

DESCRIPTION: The available tools to use.

OBJECT DEFINITION: Contains references to all of the available tool types, regardless of whether or not there is an instance defined within the library.

backgroundTasks: function ()

LOCATION: core.js

USAGE: `backgroundTasks()`

PARAMETERS: NA

DESCRIPTION: Background tasks are executed one time, but this function will get invoked periodically.
Trigger condition events:
SL.onEvent({type: 'condition', target: 'document'})
Best practices designate the monitoring of the execution time of the background tasks. If the total time gets to around 50ms for any production site, then either optimize the implementation or start using a task queue.

This monitoring can be accomplished by adding a line like this to the function:

SL.notify('Background tasks executed in ' + (end - start) + 'ms', 3)

basePath: function ()

LOCATION: core.js

USAGE: `basePath()`

PARAMETERS: NA

DESCRIPTION: Returns the base path of all DTM generated assets

bind: function (func, context)

LOCATION: core.js

USAGE: `bind(func, context)`

PARAMETERS:

- `func`: a function
- `context`: an object to be bound as the context of the designated function

DESCRIPTION: Binds a context permanently to a context. The returned function is a new function which, when called, will call the passed in function with `context` bound to it.

browserInfo: Object

LOCATION: core.js

USAGE: `browserInfo`

DESCRIPTION: Defined as a summary object within detectBrowserInfo()

OBJECT DEFINITION: { browser: type, os: type, deviceType: type }

checkAsyncInclude: function ()

LOCATION: core.js

USAGE: `checkAsyncInclude()`

PARAMETERS: NA

DESCRIPTION: Checks to see if DTM is loaded asynchronously. If DTM is loaded asynchronously then in-page HTML and the "pageBottom" event will not work.

cleanText: function (str)

LOCATION: core.js

USAGE: `cleanText(str)`

PARAMETERS:

- `str`: any string value

DESCRIPTION: Returns a string without double-spaces or values in the \000-\177 range.

configurationSettings: Object

LOCATION: core.js

USAGE: `configurationSettings`

DESCRIPTION: Contains all of the current DTM settings in a pre-queue format.

contains: function (arr, obj)

LOCATION: core.js

USAGE: `contains(arr, obj)`

PARAMETERS:

- `arr`: designated array
- `obj`: designated object

DESCRIPTION: Checks to see if an array contains an object.

containsElement: function (container, maybe)

LOCATION: core.js

USAGE: `containsElement(elm1, elm2)`

PARAMETERS:

`elm1`: the possible parent element
`elm2`: the possible child element

DESCRIPTION: Given DOM elements `elm1` and `elm2`, returns whether `elm1` contains `elm2`.
Cf., http://stackoverflow.com/questions/6130737/mouseenter-without-jquery/

cssQuery: function (css, cb)

LOCATION: core.js

USAGE: `cssQuery(css)`

PARAMETERS:

- `css`: the CSS selector
- `cb`: the callback function

DESCRIPTION: Return a list of elements matching the given CSS selector

data: Object

LOCATION: loads on init from /lib/, see init process

USAGE: `data`

DESCRIPTION: Contains contextually valuable data

OBJECT DEFINITION:
```
  URI
  browser
  customVars
    appname
  cartItems
  revenue
  saleData
```

dataCache: Object

LOCATION: core.js

USAGE: `dataCache`

DESCRIPTION: Internal cache used by $data()

OBJECT DEFINITION: This object should not be used as an external reference and is manipulated by $data() as necessary

dataElementSafe: function (key, length)

LOCATION: core.js

USAGE: `dataElementSafe(key, length)`
Extended usage: * dataElementSafe.pageviewCache [UNUSED]

PARAMETERS:
- `key`: data element(s) name(s)
- `length`: number of elements to be processed

DESCRIPTION: Internal method used to get and set data element values and persistence.

dataElements: Object

LOCATION: loads on init from /lib/ see init process

USAGE: Defined during init process

DESCRIPTION: Complete listing of all available data elements

OBJECT DEFINITION: Listing of all the available data element definitions in the library regardless of their availability in an instance.

detectBrowserInfo: function ()

LOCATION: core.js

USAGE: `detectBrowserInfo()`

PARAMETERS: NA

DESCRIPTION: Looks for browser type, OS, and device type. Similar to the usage provided by http://jsbin.com/inubez/3/

Identified browser types: OmniWeb, Opera Mini, Opera Mobile, Opera, Mobile Safari, Chrome, Firefox, IE Mobile, IE, Safari

Identified OS types: iOS, Blackberry, Symbian OS, Maemo, Android, Linux, Unix, Windows, MacOS

Identified device types: iPhone, iPad, iPod, Nokia, Windows Phone, Blackberry, Android

Desktop (everything else)

directCallRules: Array[0]

LOCATION: loads on init from /lib/ see init process

USAGE: Defined during init process

DESCRIPTION: An array of all defined direct call rules

OBJECT DEFINITION: A complete listing of all of the defined direct call rules in the library. Returned as an array of objects.

domReady: function (fn)

LOCATION: core.js

USAGE: `domReady(fn)`

PARAMETERS:

- `fn`: a callback function to be called at `domready`

DESCRIPTION: Registers a callback to be called when the DOM is fully parsed and loaded.

*domReady is borrowed from https://github.com/ded/domready/

domReadyFired: boolean

LOCATION: core.js

USAGE: `domReadyFired`

PARAMETERS: NA

DESCRIPTION: Defined and set within domReady() and returns whether or not document.readyState has switched from ‘loading’ to ‘interactive’

dynamicRules: Array[0]

LOCATION: core.js

USAGE: `dynamicRules`

DESCRIPTION: Internal rule processing array, defined and initialized in init()

each: function (arr, func, context)

LOCATION: core.js

USAGE: `each(arr, func, [context])`

PARAMETERS:
- `arr`: an array
- `func(item, index, arr)`: a function which accepts each item in the array once.

It takes these arguments:
  * `item`: an item
  * `index`: the array index of designated item
  * `arr`: the array
- `context`: the context to be bound to `func` when it is invoked

DESCRIPTION: A method for array iteration without having to write a for-loop.

ecommerce: Object

LOCATION: loads on init from /lib/ see init process

USAGE: Defined during init process

DESCRIPTION: E-commerce API

OBJECT DEFINITION: The ecommerce API that allows the integration of e-commerce tracking with DTM.

More details on the [Google Analytics E-Commerce API's] can be found here (http://code.google.com/apis/analytics/docs/gaJS/gaJSApiEcommerce.html).
Upon any of the methods on the API being called, they will fire an event, which in turn can be handled by a rule in the library.

Add an item to the transaction:
`addItem(orderId, sku, name, category, price, quantity)`

Add a new transaction:
`addTrans(orderId, affiliation, total, tax, shipping, city, state, country)`

Send the transaction data that's been set up using `addItem()` and `addTrans()` to Google Analytics to be tracked:
`trackTrans()`

ensureCSSSelector: function ()

LOCATION: core.js

USAGE: `ensureCSSSelector()`

DESCRIPTION: If the browser does not support CSS selector queries, one needs to be included. This is the function that loads selector.js

equalsIgnoreCase: function (str1, str2)

LOCATION: core.js

USAGE: `equalsIgnoreCase(str1, str2)`

PARAMETERS:
- `str1`: a designated string
- `str2`: a designated string

DESCRIPTION: Checks to see if two strings are equivalent if case is ignored

errors: Array[0]

LOCATION: core.js

USAGE: `errors`

DESCRIPTION: Internal error handling. This pushes errors up using _satellite.notify()

eventEmitterBackgroundTasks: function ()

LOCATION: core.js

USAGE: `eventEmitterBackgroundTasks()`

PARAMETERS: NA

DESCRIPTION: Internal function that calls `registerElements` on all event emitters.

eventEmitters: Array[8]

LOCATION: core.js

USAGE: `eventEmitters[]`

DESCRIPTION: Internal array that is initialized within initEventemitters()

every: function (arr, cond, context)

LOCATION: core.js

USAGE: `every(arr, cond, context)`

PARAMETERS:
- `arr`: the designated array
- `cond`: the designated condition
- `context`: the designated context

DESCRIPTION: An array helper function. Returns true if `cond(item)` returns true for every item in the array.

evtHandlers: Object [UNUSED]

LOCATION: core.js

USAGE: NA

DESCRIPTION: Internally reserved object

execute: function (trig, elm, evt, tools)

LOCATION: core.js

USAGE: `execute(trig, elm, evt, tools)`

PARAMETERS:
- `trig`: the designated trigger
- `elm`: the designated element
- `evt`: the designated event
- `tools`: the designated tool

DESCRIPTION: An internal function to execute commands.

extend: function (dst, src)

LOCATION: core.js

USAGE: `extend(dst, src)`

PARAMETERS:

- `dst` - object to copy to
- `src` - object to copy from

DESCRIPTION: Extend an object with the properties of another.

filter: function (arr, cond, context)

LOCATION: core.js

USAGE: `filter(arr, cond, context)`

PARAMETERS:

`arr` - designated array

- `cond` - designated condition
- `context` - designated context

DESCRIPTION: Parameters are the same as `SL.each` except the second argument is `cond` instead of `func` and it is expected to return a truthy value representing whether to include this item in the return array or not. This is a useful method for taking an array and filtering down to a subset of the elements.

filterRules: function ()

LOCATION: core.js

USAGE: `filterRules()`

PARAMETERS: NA

DESCRIPTION: Internal function to filter `SL.rules` down to only the ones relevant for the current page.

find: function (arr, cond, context)

LOCATION: core.js

USAGE: `find(arr, obj)`

PARAMETERS;

- `arr`: designated array
- `obj`: designated object

DESCRIPTION: Return the index of an object within an array.

fireOnceEvents: Array[2]

LOCATION: core.js

USAGE: `fireOnceEvents`

DESCRIPTION: The names of the events that can only fire once.

fireEvent: function (type, target)

LOCATION: core.js

USAGE: `fireEvent(type, target)`

PARAMETERS:

- `type`: designated type
- `target`: designated target

DESCRIPTION: Wrapper function for onEvent().

firePageLoadEvent: function (type)

LOCATION: core.js

USAGE: `firePageLoadEvent(type)`

PARAMETERS:

- `type`: valid values are `pagetop`, `pagebottom`, `domready` and `windowload`.

DESCRIPTION: Fire a page load event

fireRule: function (rule, elm, evt)

LOCATION: core.js

USAGE: `fireRule(rule, elm, evt)`

PARAMETERS:

- `rule`: designated rule name
- `elm`: designated element

- `evt`: designated event

DESCRIPTION: Fire all the trigger actions associated with a rule

flushPendingCalls: function (tool)

LOCATION: core.js

USAGE: `flushPendingCalls(tool)`

PARAMETERS:

- `tool`: a designated tool

DESCRIPTION: Internal function that clears all pending calls related to a specific tool

getDataElement: function (variable, suppressDefault, dataDef)

LOCATION: core.js

USAGE: `getDataElement(variable, suppressDefault, dataDef)`

PARAMETERS:

- `variable`: a designated variable
- `suppressDefault`: designate the suppression of the default value
- `dataDef`: the designated data definition

DESCRIPTION: Internal function to retrieve data elements

getElementProperty: function (elm, prop)

LOCATION: core.js

USAGE: `getElementProperty(elm, prop)`

PARAMETERS:

- `elm`: designated element
- `prop`: designated property

DESCRIPTION: Returns the designated property of the requested element

EXAMPLE: _satellite.getElementProperty(document.querySelectorAll("a")[0],"href")

getObjectProperty: function (obj, property)

LOCATION: core.js

USAGE: `getObjectProperty(obj, property)`

PARAMETERS:

- `obj`: designated object
- `property`: designated property

DESCRIPTION: Get a potentially nested property from an object

getQueryParam: function (key)

LOCATION: core.js

USAGE: `getQueryParam(key)`

PARAMETERS:

- `key`: the designated URL query parameter

DESCRIPTION: Returns the value of the designated URL query parameter

getVar: function (variable, elm, evt)

LOCATION: core.js

USAGE: `getVar(variable, elm, evt)`

PARAMETERS:

```
- `variable`: designated variable name
- `elm`:  designated element
- `evt`:  designated event

DESCRIPTION: This is where variables are retrieved.
 Mapped values are:
   URI = data.URI
   uri = data.URI
   protocol = document.location.protocol
   hostname = document.location.hostname

 These mapped values can be dereferenced using the %% syntax in the UI
```

getVars: function (variables, elm, evt)

```
LOCATION: core.js

USAGE: `getVars(variable, elm, evt)`

PARAMETERS:

- `variable`: designated variables
- `elm`: designated element
- `evt`: designated event

DESCRIPTION: each() iteration through getVar()
```

handleEvent: function (evt)

```
LOCATION: core.js

USAGE: `handleEvent(evt)`

PARAMETERS:

- `evt`: designated event

DESCRIPTION: Internal function to prevent the processing of an event twice
```

handleOverrides: function ()

LOCATION: core.js

USAGE: `handleOverrides()`

PARAMETERS: NA

DESCRIPTION: Internal function that manages key boundaries

hasMultipleDomains: function ()

LOCATION: core.js

USAGE: `hasMultipleDomains()`

PARAMETERS: NA

DESCRIPTION: Returns whether or not the library has been configured to handle multiple domains

hasSelector: true

LOCATION: core.js

USAGE: `hasSelector`

DESCRIPTION: Internal designator that is defined and initialized within ensureCSSSelector()

indexOf: function (arr, obj)

LOCATION: core.js

USAGE: `indexOf(arr, obj)`

PARAMETERS;

- `arr` - designated array
- `obj` - designated object

DESCRIPTION: Returns the index of an object within an array.

inherit: function (subClass, superClass)

LOCATION: core.js

USAGE: `inherit(subClass, superClass)`

PARAMETERS:

- `subClass`: a JavaScript function representing a constructor - the inheritor
- `superClass`: another constructor - the one to inherit from

DESCRIPTION: This function makes `subClass` inherit `superClass`

init: function (settings)

LOCATION: core.js

USAGE: `init()`

PARAMETERS:

- `settings`: all the settings that comprise a library.

DESCRIPTION: Internal function that initializes DTM. Constructs background tasks and page load events.

initEventEmitters: function ()

LOCATION: core.js

USAGE: `initEventEmitters()`

PARAMETERS: NA

DESCRIPTION: Internal function that initializes all event emitters

initTools: function (toolSpecs)

LOCATION: core.js

USAGE: `initTools(toolSpecs)`

PARAMETERS:

- `toolSpecs`: designated tool specifications

DESCRIPTION: Internal function that initializes all tools.

isArray: function isArray()

LOCATION: core.js

USAGE: `isArray(item)`

PARAMETERS:
- `item`: item to be evaluated

DESCRIPTION: Returns whether the given item is an array

isHttps: function ()

LOCATION: core.js

USAGE: `isHttps()`

PARAMETERS: NA

DESCRIPTION: Returns 'true' if the location.protocol of the current page is 'https', otherwise, returns 'false'.

isLinked: function (elm)

LOCATION: core.js

USAGE: `isLinked(elm)`

PARAMETERS:

- `elm`: the element to test

DESCRIPTION: Returns whether or not the element is an anchor, a descendant of an anchor, or contains an anchor

isLinkerLink: function (elm)

LOCATION: core.js

USAGE: `isLinkerLink(elm)`

PARAMETERS:

- `elm`: the designated element

DESCRIPTION: Returns whether or not the designated element is a part of the internal domain list and is also an outbound link

isObject: function (item)

LOCATION: core.js

USAGE: `isObject(item)`

PARAMETERS:

- `item`: designated item

DESCRIPTION: Returns whether or not the item is an object

isOutboundLink: function (elm)

LOCATION: core.js

USAGE: `isOutboundLink(elm)`

PARAMETERS:

- `elm`: the designated element to test

DESCRIPTION: Returns whether or not the element is an outbound link

isRegex: function (item)

LOCATION: core.js

USAGE: `isRegex(item)`

PARAMETERS:

- `item`: designated item

RETURN TYPE: String

DESCRIPTION: Returns whether or not the designated item is a RegExp object

isRuleActive: function (rule, date)

LOCATION: core.js

USAGE: `isRuleActive(rule,date)`

PARAMETERS:

- `rule`: designated rule identifier
- `date`: designated date

DESCRIPTION: Returns whether or not the designated rule is currently scheduled and active

isString: function (item)

LOCATION: core.js

USAGE: `isString(item)`

PARAMETERS:

- `item`: designated item

DESCRIPTION: Returns whether the designated item is a string

isSubdomainOf: function (sub, root)

LOCATION: core.js

USAGE: `isSubdomainOf(sub, root)`

PARAMETERS:

- `sub`: designated sub-domain
- `root`: designated root domain

DESCRIPTION: Validates that a sub-domain is actually a part of the root domain

loadEventBefore: function (one, other)

LOCATION: core.js

USAGE: `loadEventBefore(one, other)`

PARAMETERS:

- `one`: first designated event
- `other`: second designated event

DESCRIPTION: Internal queuing function to set event order

loadScript: function (url, callback)

LOCATION: core.js

USAGE: `loadScript(url, [callback])`

PARAMETERS:

- `url`: the URL of the script
- `callback`(optional): the function to be called after the script has loaded.

DESCRIPTION: Load an external script.

loadStoredSettings: function ()

LOCATION: core.js

USAGE: `loadStoredSettings()`

PARAMETERS: NA

DESCRIPTION: This function reads localStorage values to get the current debug and hide activity settings

logError: function (err)

LOCATION: core.js

USAGE: `logError()`

PARAMETERS: NA

DESCRIPTION: Raises a DTM console error

Example: _satellite.logError(new Error('Failed to load script'))

map: function (arr, func, context)

LOCATION: core.js

USAGE: `map(arr, func)`

PARAMETERS:

Parameters are the same as `SL.each`, except that `func` is expected to return a value in the corresponding index of the returned array.

DESCRIPTION: A method for mapping an array to another array using a one-to-one mapping for each element

matchesCss: function (selector, elm)

LOCATION: core.js

USAGE: `matchesCss(css, elm)`

PARAMETERS:

- `css`: the CSS selector
- `elm`: the element

DESCRIPTION: Returns whether or not a DOM element matches a given css selector

notify: function (msg, pty)

LOCATION: core.js

USAGE: `notify(msg, pty)`

PARAMETERS:

- `msg`: message to print
- `pty`: priority

DESCRIPTION: Notify the user of things happening in DTM using a protected version of `console.log`.

Priority list:
1 - 3: (no change to output): prefixed with "SATELLITE: "
4: Leverages console.warn and prefixed with "SATELLITE: "
5: Leverages console.error and prefixed with "SATELLITE: "

If a priority is not provided then this error will be returned: "SATELLITE: Notify called with incorrect priority."

EXAMPLE: _satellite.notify('debug',1)

onEvent: function (evt)

LOCATION: core.js

USAGE: `onEvent(evt)`

PARAMETERS:

- `evt`: the event triggered

DESCRIPTION: Internal function that handles an event, whether it is a DOM event or a synthetic event.

pageBottom: function ()

LOCATION: core.js

USAGE: `pageBottom()`

PARAMETERS: NA

DESCRIPTION: This function is to be called by the web page, just before the closing `</body>` tag, using a script tag. Only one pageBottom() call should appear on a page.

EXAMPLE: <script>_satellite.pageBottom()</script>

pageBottomFired: true

LOCATION: core.js

USAGE: `pageBottomFired`

PARAMETERS: NA

DESCRIPTION: Defined and initialized in pageBottom() and references whether or not the pageBottom() function has fired

pageLoadPhases: Array[4]

LOCATION: core.js

USAGE: `pageLoadPhases`

DESCRIPTION: An array that contains the four components of the web page load.
pagetop
pagebottom
domready
windowload

pageLoadRules: Array[1]

LOCATION: loads on init from /lib/ see init process

USAGE: Defined during init process

DESCRIPTION: Internal page load queue array

parseQueryParams: function (str)

LOCATION: core.js

USAGE: `parseQueryParams(str)`

PARAMETERS:

- `str`: the designated URL query parameter

DESCRIPTION: Parses the designated URL query parameter

RETURN TYPE: Object

EXAMPLE: _satellite.parseQueryParams(window.location.search)

preprocessArguments: function (args, elm, evt, forceLowerCase)

LOCATION: core.js

USAGE: `preprocessArguments(args, elm, evt, forceLowerCase)`

PARAMETERS:

- `args`: designated arguments
- `elm`: designated element
- `evt`: designated event
- `forceLowerCase`: force lowercase evaluation

DESCRIPTION: Internal pre-process arguments (variable substitutions and lower-casing) before feeding them to the tools

preventDefault: function (e)

LOCATION: core.js

USAGE: `preventDefault(evt)`

PARAMETERS:

- `evt`: the event triggered

DESCRIPTION: Prevent the default browser behavior for this event

propertiesMatch: function (property, elm)

LOCATION: core.js

USAGE: `propertiesMatch(property, elm)`

PARAMETERS:

- `property`: designated property
- `elm`: designated element

DESCRIPTION: Determines whether or not an element contains a specific property. Additional usage: * SL.propertiesMatch.fallbacks

pushAsyncScript: function (cb)

LOCATION: core.js

USAGE: `pushAsyncScript(callback)`

PARAMETERS:

- `callback`: callback function

DESCRIPTION: Protected function called by an async custom user script

pushBlockingScript: function (cb)

LOCATION: core.js

USAGE: `pushBlockingScript(callback)`

PARAMETERS:

- `callback`: callback function

DESCRIPTION: Protected function called by a blocking custom user script

readCookie: function (name)

LOCATION: core.js

USAGE: `readCookie(name)`

PARAMETERS:

- `name`: the name of a valid domain cookie

DESCRIPTION: Reads the value out of a stored cookie

readStoredSetting: function (name)

LOCATION: core.js

USAGE: `readStoredSetting(name)`

PARAMETERS:

- `name`: a valid localStorage name

DESCRIPTION: Reads the localStorage setting of the given name

EXAMPLE: _satellite.readStoredSetting('stagingLibrary')

realGetDataElement: function (dataDef)

LOCATION: core.js

USAGE: `realGetDataElement(dataDef)`

PARAMETERS:

- `dataDef`: valid data element definition

DESCRIPTION: Internal data element function

registerEvents: function (elm, events)

LOCATION: core.js

USAGE: `registerEvents(elm, events)`

PARAMETERS:

- `elm`: the element on which to listen for events
- `events`: an array of event types (strings)

DESCRIPTION: Register events for an element using `track` as the callback

registerEventsForTags: function (tags, events)

LOCATION: core.js

USAGE: `registerEventsForTags(tags, events)`

PARAMETERS:

- `tags`: an array of tags to match (strings)
- `events`: an array of event types (strings)

DESCRIPTION: Register events for all elements that have the specified tags

registerNewElementsForDynamicRules: function ()

LOCATION: core.js

USAGE: `registerNewElementsForDynamicRules()`

PARAMETERS: NA

DESCRIPTION: Find rules that poll for dynamically injected elements on the page and register events for them

removeCookie: function (name)

LOCATION: core.js

USAGE: `removeCookie(name)`

PARAMETERS:

- `name`: the designated cookie

DESCRIPTION: Removes a stored cookie

replace: function (str, elm, evt)

LOCATION: core.js

USAGE: `replace(str, [elm], [target])`

PARAMETERS:

- `str`: designated string to apply substitutions
- `elm`(optional): object or element to use for substitutions. Use the form `%this.property%`
- `target`(optional): target element to use for substitution. Use the form `%target.property%`

DESCRIPTION: Perform variable substitutions to a string where substitutions are specified in the form `"%foo%"`. Variables are looked up either in `SL.data.customVars`, or if the `elm` parameter is passed it, and the variable specification is in the form `"%this.tagName%"`, then it is substituted with the properties on `elm`, i.e., `elm.tagName`

ruleInScope: function (rule, LOCATION)

LOCATION: core.js

USAGE: `ruleInScope(rule, LOCATION)`

PARAMETERS:

- `rule`: the designated rule
- `LOCATION`: the designated location, or scope

DESCRIPTION: Determines whether or not a rule is in scope

ruleMatches: function (rule, evt, elm, eventEntriesFound)

LOCATION: core.js

USAGE: `ruleMatches(rule, evt, elm, eventEntriesFound)`

PARAMETERS:

- `rule`: the rules to match
- `evt`: the triggered event
- `elm`: the element to which the event was attached
- `eventEntriesFound`: number of currently matched rules

DESCRIPTION: Evaluates the matching of rules to events and elements

rules: Array[16]

LOCATION: core.js (also in /lib/ init process)

USAGE: `rules`

DESCRIPTION: Returns an array of the currently active rules

scriptOnLoad: function (url, script, callback)

LOCATION: core.js

USAGE: `scriptOnLoad(url, script, callback)`

PARAMETERS:

- `url`: the URL of the designated script
- `script`: the designated script
- `callback`: the callback function

DESCRIPTION: The worker function for loading a script. In most cases, the wrapper function `loadScript()` should be used instead of `scriptOnLoad()`

searchVariables: function (vars, elm, evt)

LOCATION: core.js

USAGE: `searchVariables(vars, elm, evt)`

PARAMETERS:

- `vars`: the designated variables
- `elm`: the designated element
- `evt`: the designated event

DESCRIPTION: From an object literal of a variable, generate a query string.

setCookie: function (name, value, days)

LOCATION: core.js

USAGE: `setCookie(name, value, days)`

PARAMETERS:

- `name`: the designated cookie name
- `value`: the value to be set
- `days`: the number of days before the cookie is to expire

DESCRIPTION: Function to set cookie values

setDebug: function (debug)

LOCATION: core.js

USAGE: `setDebug(debug)`

PARAMETERS:

- `debug`: (true) enables and (false) disables JavaScript console debugging

DESCRIPTION: Enables or disables JavaScript console debugging. Only works if window.localStorage exists. Sets 'sdsat_debug' in window.localStorage

setFormListeners: function ()

LOCATION: core.js

USAGE: `setFormListeners()`

PARAMETERS: NA

DESCRIPTION: Internal function to listen for events on form elements

setListeners: function ()

LOCATION: core.js

USAGE: `setListeners()`

PARAMETERS: NA

DESCRIPTION: Internal function to set events for `document`

setLocation: function (url)

LOCATION: core.js

USAGE: `setLocation(url)`

PARAMETERS:

- `url`: the URL to set to

DESCRIPTION: Set the URL (window.location) of the current page

setVar: function ()

LOCATION: core.js

USAGE: `setVar(name, value)` or `setVar(mapping)`

PARAMETERS:

Set a variable. Can be either called like this:

_satellite.setVar('name', 'value')

Or by passing in a mapping(object literall) which allows setting multiple variables at the same time.

_satellite.setVar({name: 'value', foo: 'bar'})

DESCRIPTION: Used to set a variable. This can be used to manipulate page level data elements, or any valid user level custom variable. New custom variables are stored in _satellite.data.customVars

setVideoListeners: function ()

LOCATION: core.js

USAGE: `setVideoListeners()`

PARAMETERS: NA

DESCRIPTION: Internal function to listen for events on video elements.

settings: Object

LOCATION: loads on init from /lib/ see init process

USAGE: `settings`

DESCRIPTION: Contains all of the current _satellite settings. The initial settings are the same as _satellite.configurationSettings.settings, but are referenced by page level events, whereas configurationSettings is not.

specialProperties: Object

LOCATION: core.js

USAGE: `specialProperties`

DESCRIPTION: Contains both the ‘text’ and ‘cleanText’ versions of a string.

USAGE: specialProperties.text
    specialProperties.cleanText

stagingLibraryOverride: function ()

LOCATION: core.js

USAGE: `stagingLibraryOverride()`

PARAMETERS: NA

DESCRIPTION: This allows Rover to configure the browser to use the staging library instead

stopPropagation: function (e)

LOCATION: core.js

USAGE: `stopPropagation(evt)`

PARAMETERS:

`evt`: the event triggered

DESCRIPTION: Cross-browser `stopPropagation`

text: function (obj)

LOCATION: core.js

USAGE: `text(obj)`

PARAMETERS:

- `obj`: said object

DESCRIPTION: Returns either textContent or innerText of an object if they are defined.

textMatch: function (str, pattern)

LOCATION: core.js

USAGE: `textMatch(str, str_or_regex)`

PARAMETERS:

`str`: the input string to be matched
`str_or_regex`: the pattern to match against. If this is a string, it requires an exact match. If it is a regex, then it will do regex match.

DESCRIPTION: Perform a string match based on another string or a regex.

EXAMPLES:

String match: _satellite.textMatch(window.tm_omn.prop21,'12345')
Regex match: _satellite.textMatch(window.tm_omn.prop21,/^12345$/i)

throttle: function (fn, delay)

LOCATION: core.js

USAGE: `throttle(fn, delay)`

PARAMETERS:

- `fn`: a function
- `delay`: delay in milliseconds

DESCRIPTION: *Throttles* a function `fn` to be called no more than once during the interval specified by `delay`

More information on throttling a function can be found here http://remysharp.com/2010/07/21/throttling-function-calls/

toArray: function (thing)

LOCATION: core.js

USAGE: `toArray(item)`

PARAMETERS:

- `item`: the designated item to convert into an array

DESCRIPTION: Provides a fallback method if prototype.slice() does not work.

tools: Object

LOCATION: loads on init from /lib/ see init process

USAGE: `tools`

DESCRIPTION: The current list of tools available on the page

track: function (ruleName)

LOCATION: core.js

USAGE: `track(id)`

PARAMETERS:

- `ruleName`: the designated rule name

DESCRIPTION: Directly launch an event by id. Extra spaces are trimmed that may exist from the beginning and end of the string. This is the preferred method of firing a direct call rule.

EXAMPLE: _satellite.track(`rule_id`)

trim: function (str)

LOCATION: core.js

USAGE: `trim(str)`

PARAMETERS:

- `str`: the designated string

DESCRIPTION: Removes spaces from the beginning and end of a string and then returns the result

uuid: 4

LOCATION: core.js

USAGE: `uuid`

DESCRIPTION: Internal reference value used by $data().

whenEvent: function (evtName, callback)

LOCATION: core.js

USAGE: `whenEvent(evtName, callback)`

PARAMETERS:

- `evtName`: the designated event
- `callback`: the callback function

DESCRIPTION: Specifies a callback function to be fired when an event is triggered

DTM Topical Scripting Reference

APIs:

ecommerce

Arrays:

any
contains
each
every
filter
find
*handleOverrides
indexOf
isArray
map
toArray

Classes:

inherit

Cookies and localStorage:

readCookie
readStoredSetting
removeCookie
setCookie

Data Elements:

dataElements
*dataElementSafe
*getDataElement
getVar
getVars
*realGetDataElement

Elements and CSS:

containsElement

cssQuery
*ensureCSSSelector
*execute
getElementProperty
*hasSelector
isLinked
isLinkerLink
isOutboundLink
matchesCss
propertiesMatch
*registerNewElementsForDynamicRules
replace

Events:

addEventHandler
availableEventEmitters
*eventEmitterBackgroundTasks
*eventEmitters
*evtHandlers
*execute
*handleEvent
fireOnceEvents
fireEvent
firePageLoadEvent
*initEventEmitters
*loadEventBefore
*onEvent
preventDefault
registerEvents
registerEventsForTags
*setFormListeners
*setListeners
*setVideoListeners
stopPropagation
whenEvent

Notifications:

logError
notify

Objects:

bind
extend
getObjectProperty
isObject
isRegex
text

Rules:

*dynamicRules
*filterRules
fireRule
directCallRules
isRuleActive
*pageLoadRules
ruleInScope
ruleMatches
rules
searchVariables
track

Scripts:

loadScript
pushAsyncScript
pushBlockingScript
scriptOnLoad

Settings:

*$data()
*BaseTool
*backgroundTasks
basePath
browserInfo
checkAsyncInclude
configurationSettings
data
*dataCache
detectBrowserInfo
*errors
getQueryParam

getVar
getVars
hasMultipleDomains
*init
isHttps
isSubdomainOf
*loadStoredSettings
parseQueryParams
*preprocessArguments
setDebug
setLocation
setVar
settings
stagingLibraryOverride
throttle
tools
*uuid

Strings:

cleanText
equalsIgnoreCase
isString
replace
specialProperties
textMatch
trim

Timing:

domReady
domReadyFired
pageBottom
pageBottomFired
pageLoadPhases

Tools:

availableTools
*flushPendingCalls
*initTools

Best Practices affected items:

backgroundTasks
checkAsyncInclude

* Internally restricted objects, settings, and functions