DYNAMIC TAG MANAGER CHEAT SHEET

DYNAMICS

Variables

Variables are used to dynamically insert text into site tags (like an eVar or event). Variables are wrapped in %...% syntax.

Category	Action		Label (Optional)	
blog post	%this.title%		path: %URI%	
ise selected e	lomont	04thic [attribute1%	

%target.[attribute]%

Dynamically insert the title and inner HTML for: Hello!

title %this.title%
Output link

inner HTML %target.innerHTML%
Output *Hello!*

Other example uses:

use child element

href %this.href%
class %this.className%
alt %this.alt%
src %this.src%
inner text %this.@text%
inner text w/trim %this.@cleanText%
custom attribute %this.getAttribute(name)%

Other useful variables:

URL pathname %URI%
URL hostname %hostname%
URL protocol %protocol%

Data Elements

Data Elements are an alternative way of setting a variable. These are most useful when you use value(s) across multiple rules.

using data elements in a rule %Element Name%

set a data element

_satellite.setVar("Element Name",value);

get a data element

_satellite.getVar("Element Name");

Note: Each time you call a data element, DTM tries to reset the value. If you need to store a value more permanently, use cookies (below).

Cookies

Cookies store values over a certain period of time. They are similar to Data Elements, but instead allow you to set a cookie memory in days (integer).

set a Cookie

_satellite.setCookie("Cookie Name",value,days);

read a cookie

_satellite.readCookie("Cookie Name");

COMP SCI

JavaScript

Below are some of the most commonly used methods in JavaScript. Note that many DTM implementations don't even use JS, but just in case...

JavaScript Regular Expressions declare regex variable var RegEx = /pattern/mod

modifier function
/g global matching
/i case insensitive
/s single line mode
/m multi line mode

JavaScript Selectors

get element by id getElementById('id')
get elements by class
get elements by tag getElementsByClassName('i)
getElementsByTagName('a')

JavaScript Logic Tests

if statementswitch statementif(i<0){some action}</td>switch(i){else if(i<1){some action}</td>case n.else{some action}some action;for...next loopdefault:for(i=0;i<10;i++){some</td>some action;action}}

Bonus!

cross-browser trim _satellite.cleanText('text');

jQuery

Below are some common jQuery methods. If available, consider using <u>Sizzle</u> as a selector.

prefixes	targeting	suffixes
\$(this)	.parents()	.text()
\$('#id')	.children()	.html()
\$('.class')	.siblings()	.val()
\$('div')	.find()	.attr()
\$('div.class')		

selectors function \$("div img") select all img within div \$("div>img") only direct img children \$("a[title='equals']") anchor title equals string \$("a[title!='noequal']") title doesn't equal string \$("a:first") select 1st anchor element \$("ul:last li:last") select last *ul* of last *li* \$("ul li:first-child") first li of every ul \$("[id*='contains']") id contains string \$("[class\$='ends']") class ends with string \$("[id^='begins']") id begins with string selects all checked boxes \$(":checked") \$(":selected") selects active inputs

Bonus!

efficient \$(this).text();

_satellite.text('text');

HOGIC

Diagnostics

DTM has useful functions built into the platform that help QA tags. The code below can be used in functions or typed directly into your console.

enable staging library in production localStorage.setItem('sdsat_stagingLibrary',true);

enable debug mode (useful to QA rules)
_satellite.setDebug(true);

hide browsing activity from DTM localStorage.setItem('hide_activity',true);

cross-browser compatible console.log()
_satellite.notify('text to display',[1-5]);

Note: Messages from _satellite.notify() are only viewable in Debug mode. Additionally, the 1-5 integer sets the importance level of the message.

get the date of the last publish _satellite.publishDate;

get the date of the last library build satellite.buildDate;

function

pattern

Regular Expressions

Regular Expressions (RegEx) is a method of matching that can be used within DTM (and JavaScript and others) to select groups of text.

Core Patterns

1	
	match any character
*	0 or more of previous expression
+	1 or more of previous expression
?	0 or 1 of previous expression
٨	start of string
\$	end of string
1	"OR" match on both sides
{}	explicit quantifier notation
[]	explicit character set match
[^]	NOT one of the characters in set
()	logical grouping of expression parts
١	precedes characters above; makes
	special character literal

		Examples
	pattern	function
ı	[abc]	matches a or b or c
ı	[a-z]	match any letter from a-z (lowercase)
ı	[A-Z]	match letter from A-Z (uppercase)
ı	[0-9]	matches any number from 0-9
ı	{3}	matches 3 of preceding expression
ı	{3.9}	matches 3-9 of preceding expression
I	{3,}	matches 3 or more



@jimalytics jimalytics.com