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%

use selected element	%this.[attribute]%
use child element	%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:

href	%this.href%
class	%this.className%
alt	%this.alt%
src	%this.src%
inner text	%this.@text%
custom attribute	%this.getAttribute(*name*)%

Note: %this.@text% pulls all text from within the element and trims the blank space around it. Use this instead of %this.innerText% - which is not compatible across browsers!

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	getElementsByClassName('')
get elements by tag	getElementsByTagName('a')

JavaScript Logic Tests

if statement
```
if(i<0){some action}
else if(i<1){some action}
else{some action}
```

for...next loop
```
for(i=0;i<10;i++){some action}
```

switch statement
```
switch(i){
 case n:
   some action;
   break;
 default:
   some action;
}
```

Bonus!

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

jQuery

Below are some common jQuery methods. If available, consider using Sizzle 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
$(":checked")	selects all checked boxes
$(":selected")	selects active inputs

Bonus!

efficient $(this).text();	_satellite.text('text');

LOGIC

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 publish
```
_satellite.buildDate;
```

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

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

Examples

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

by jimalytics.com

@Jimalytics