### DYNAMICS

#### Variables

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Action</th>
<th>Label (Optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>blog post</td>
<td>%this.title%</td>
<td>path: %URI%</td>
</tr>
<tr>
<td>use selected element</td>
<td>%this.id%</td>
<td>use child element</td>
</tr>
<tr>
<td>use child element</td>
<td>%this.[attribute]%</td>
<td>Dynamically insert the title and inner HTML for:</td>
</tr>
<tr>
<td>title</td>
<td>%this.title%</td>
<td>Output</td>
</tr>
<tr>
<td>inner HTML</td>
<td>%target.innerHTML%</td>
<td>Hello!</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td>Other example uses:</td>
</tr>
<tr>
<td>href</td>
<td>%this.href%</td>
<td>href</td>
</tr>
<tr>
<td>class</td>
<td>%this.className%</td>
<td>class</td>
</tr>
<tr>
<td>alt</td>
<td>%this.alt%</td>
<td>alt</td>
</tr>
<tr>
<td>src</td>
<td>%this.src%</td>
<td>src</td>
</tr>
<tr>
<td>inner text</td>
<td>%this.@text%</td>
<td>inner text</td>
</tr>
<tr>
<td>inner text w/trim</td>
<td>%this.@cleanText%</td>
<td>trim</td>
</tr>
<tr>
<td>custom attribute</td>
<td>%this.getAttribute(name)%</td>
<td>custom attribute</td>
</tr>
</tbody>
</table>

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);`
- `_satellite.getVar("Element Name");`

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

### COMP SCI

#### JavaScript

Below are some of the most commonly used methods in JavaScript.

**JavaScript Regular Expressions**

- declare regex variable: `var RegEx = /pattern/mod`  
- modifier: `/g` for global matching  
- `/i` for case insensitive  
- `/s` for single line mode  
- `/m` for multi line mode

**JavaScript Selectors**

- `get element by id`: `getElementById(id)`  
- `get elements by class`: `getElementsByClassName("class")`  
- `get elements by tag`: `getElementsByTagName("tag")`

**JavaScript Logic Tests**

- if statement: `if(i<0){some action};`  
- `else if(i<1){some action};`  
- `else if(i<0){some action};`  
- for...next loop: `for(i=0;i<10;i++){some action};`

**jQuery**

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

- prefixes: `$(this)`, `$.children()`, `$.html()`  
- targeting: `.parent()`  
- suffixes: `.text()`, `.show()`, `.hide()`  
- selectors: `$(div)`, `$(div).children()`  
- function: `select all div within div`  

### LOGIC

#### Diagnostics

Launch 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 debug mode (useful to QA rules) _satellite.setDebug(true);`
- `hide browsing activity from Launch localStorage.setItem('hide_activity',true);`
- `cross-browser compatible console.log() _satellite.logger.info("text to display"); //Log _satellite.logger.log("text to display"); // Another log _satellite.logger.warn("text to display"); //Warning _satellite.logger.error("text to display"); //Error`  
- `Note: Messages from _satellite.logger are only viewable in Debug mode.`

Get the date of the last library build

- `_satellite.buildInfo.buildDate;`

#### Regular Expressions

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

**Core Patterns**

- `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 character set`  
- `NOT one of the characters in set`  
- `logical grouping of expression parts`  
- `precedes characters above; makes special character literal`  

**Examples**

- `matches a or b or c`  
- `matches any letter from a-z (lowercase)`  
- `matches letter from A-Z (uppercase)`  
- `matches any number from 0-9`  
- `matches 3 of preceding expression`  
- `matches 3-9 of preceding expression`  
- `matches 3 or more`