Old syntax
==========

```
<div class="collapsible">
<div class="collapsible open">
<div class="collapsible closed">
```

```
<a href="foo.html#id" rel=".modal!class=loginModal">content</a>
<a href="foo.html#id1#id2 rel="#listToUpdate:after #buttonBarMore" class="autoLoading-visible">More&hellip;</a>
<li><a href="foo.html#cata-panel2" data-injection="#panel2:content">Cat A</a></li>
```

```
<a href="#" title="Tittle attribute content" data-tooltip="">tooltip default</a>
<a href="#" title="tooltip left top, show on click" data-tooltip="click!position=tl">
<a href="#" title="This is one sticky tooltip" data-tooltip="sticky!position=rm">tooltip right middle</a>
<a href="#" title="title" data-tooltip="position=tm-rt-rm-rb-lt">foo</a>
<a href="foo.html#myTip" data-tooltip="ajax!click" title="Tittle attribute content">
<a href="foo.html#myTip" data-tooltip="ajax" title="Tittle attribute content">
<img src="placeholder.gif" class="floatBefore" data-tooltip="stick" title="foo" alt="foo" />
<img src="placeholder.gif" class="floatAfter" data-tooltip="" title="foo" alt="foo" />
```

```
<button data-toggle="selector=#toCheck!attr=checked!value=checked">toggle checkbox</button>
<button data-toggle="selector=#toToggle!attr=class!value=red blue">change colour</button>
```

```
<a href="index.html#mainContent#appendix" data-injection="#mainContent#appendix" data-setclass="id=body!values=template-*:template-home">Healthy<br/>Workplaces</a>
```

```
<a href="index.html#mainContent#appendix" data-injection="#mainContent#appendix" data-setclass="id=body!values=template-*:template-home">Healthy<br/>Workplaces</a>
```

Carousel
========

carrousel: triggered by class presence, but how to handle the options?

```
<ul class="carousel">
```

```
<ul class="carousel" data-carousel="method: loop; step: 2; speed: 0.3s; controls: slider buttons; direction: vertical">  <-- individual properties method
<ul class="carousel" data-carousel="options: loop 2 0.3s slider buttons vertical"> <-- shorthand method
```

```
data-caroussel="options: loop slider buttons vertical; step: 2; speed: 0.3s"
data-caroussel="loop slider buttons vertical; step: 2; speed: 0.3s"
```

```
<ul class="caroussel" adta-caroussel="effect -- easeIn">
</ul>
```

Carousel options
----

* Loop (Default off)
* Steps (Default 1 step)
* Controls (Buttons, slider, default buttons only and touch as standard)
* Speed (Default 0.2s)
* Direction (Default horizontal)
* More?

Toggle
======

data-toggle="#id h3; speed: 0.3"

data-tooltip="options: slow sticky ajax" <- positional with short-hand second parameter
data-tooltip="slow sticky ajax"
data-tooltip="options: slow sticky ajax"

<img title="Lorem ipsum" class="tooltip" data-tooltip="sticky">

<button disabled="disabled" data-toggle="<selector>;<values>;<attr>"> ?
or:
<button disabled="disabled" data-toggle="<values>;<attr>;<selector>"> ?

<button class="disabled" data-toggle="disabled enabled"> -- same as line 91
<button class="disabled" data-toggle="values: disabled enabled; attr: class; selector: #someID">
<button class="disabled" data-toggle="disabled enabled; attr: class; selector: #someID">
<button class="disabled" data-toggle="disabled enabled; class; selector: #someID">
<button class="disabled" data-toggle="disabled enabled; selector: #someID">
<button class="disabled" data-toggle="disabled enabled; ; #someID">

<button disabled="disabled" data-toggle="values: disabled enabled">
<button disabled="disabled" data-toggle="attr: disabled; values: disabled enabled">

Tooltip (With Ajax)
===================

<a href="balloon-contents.html#myTip" data-tooltip="ajax">
<a href="balloon-contents.html#myTip" data-injection="as -- .toolip">

- fetch ballon-contents
- take #myTip content
- create div
- add content to div
- add class tooltip to div

<a href="balloon-contents.html#myTip" class="tooltip">

<a href="balloon-contents.html#myTip" data-injection="as -- .collapsible">

<a href="foo.html#id1#id2" data-injection="#listToUpdate:after #buttonBarMore" class="autoLoading-visible">More&hellip;</a> <-- Current syntax

<a href="foo.html" data-injection="#listToUpdate:after #buttonBarMore" class="autoLoading-visible">More&hellip;</a>

Change
======

Adding:
<button data-add=".someClass; space sep list of values to be added">
<button data-add=".someClass; space sep list of values to be added; class"> (same as above)
<button data-add=".someClass; space sep list of values to be added; attr: class"> (same as above)
<button data-change=".someClass; add: space sep list of attr values to be added; attr: class">

<button data-add=".someClass; space sep list of attr values to be added; attr: attrname">
<button data-change=".someClass; add: space sep list of attr values to be added; attr: attrname">

Removing:
<button data-remove=".someClass; space sep list of values to be removed">
<button data-remove=".someClass; space sep list of values to be removed; class"> (same as above)
<button data-remove=".someClass; space sep list of values to be removed; attr: class"> (same as above)
<button data-change=".someClass; remove: space sep list of attr values to be removed; attr: class">

<button data-remove=".someClass; space sep list of attr values to be added; attr: attrname">
<button data-change=".someClass; remove: space sep list of attr values to be added; attr: attrname">

Switching:
<a data-switch="body; *a b*"> (switch the two classes)
<a data-switch="body; *a-* b*"> (switch the two classes - If a-* does not exist, never mind)
<a data-change="body; *switch: a-* b*"> (switch the two classes - If a-* does not exist, never mind)

toggle/rotate elements independently:
<a data-toggle="body; a" (set/unset a)
<a data-toggle="body; a b" (toggle between a and b, start with a if not there)
<a data-toggle="body; a b c" (rotate, start with a if not there)

All together:
<a data-change="body; toggle: e f; *switch: a b*; remove: c; add: d">
<a data-change="body; toggle: e f; *switch: a b*; remove: c; add: d; attr=attrname"> (attr valid for whole directive, separate with && if you need different)

~~<a data-set="body; *a b*; add"> (adds the two classes)~~
~~<a data-set="body; *a b*; method: add"> (Equal to above line)~~

~~<a data-add="body; *a b*"> (adds the two classes - doesn't matter whether there already)~~

~~<a data-set="body; *a b*; remove"> (Removes the two classes)~~
~~<a data-set="body; *a b*; method: remove"> (Equal to above line)~~

~~<a data-remove="body; *a b*"> (Removes the two classes - no failure if non-existant)~~

~~<a data-set="body; *a b*; switch"> (switch the two classes)~~
~~<a data-set="body; *a b*; method: switch"> (switch the two classes)~~
or:

<a data-change="body; *switch: a b*" (switch the two classes)

<a data-set="body; a b && body; c e; switch && body; x y; remove">
<a data-set="body; a b && body; switch: c e && body; remove: x y">

<a data-set="body; switch: *template-* mode-*, template-home and-another*"; attr=class> (same as above)

directive -> option
option -> boolean, key-val
val ->

data-add=".some
data-remove=
data-change

~~<button data-set=".someClass; mode-* mode-A">   (Replacing with wildcards)~~
~~<button data-set=".someClass; replace: mode-* mode-A">   (Equal to line above)~~

~~<button data-set=".someClass; replace: mode-* mode-A; add: space sep list of values to be added">~~

~~<button data-set=".someClass; mode-A">   (Just adds the class mode-A to element with .someClass. Other classes on target element remain intact?)~~
~~<button data-set=".someClass; mode-A && .someOtherClass; mode-A">~~
~~<button data-set=".someClass; mode-A">~~

~~<button data-set="#someID; checked; attr: value">   (Sets an <input type="checkbox" to checked="checked")~~

~~<a data-set="id=body!values=template-*:template-home"> (Current syntax)~~

~~<a data-set="body; values: template-* NEWNEEDED template-home">~~
~~<a data-set="body; template-* mode-* -- to -- template-home and-another">~~
~~<a data-set="body; remove: template-* mode-*; add: template-home and-another">~~
~~<a data-set="body; replace: template-* mode-*; template-home and-another"> (BAD)~~
~~<a data-set="body; switch: template-* mode-*; template-home and-another"> (BAD)~~
~~<a data-set="body; switch-from: template-* mode-*; switch-to: template-home and-another"> (BAD)~~
~~<a data-set="body; switch: *template-* mode-*, template-home and-another*"> (Maybe? - Should work)~~

~~<a data-set="body; switch: mode-* *mode-B && body; switch view-* view-B"*>~~

Data injection
==============

80% use case single id to inject into same id

<a href="foo.html#id" class="inject">

<a href="foo.html" data-injection="#id"> Bad, because it does not jump to #id in text browsers

<a href="foot.html#id1#id2" class="inject">

<a href="foot.html#id" class="inject">
<a href="foot.html#id1#id2" class="inject">

Injection copies the content from the source tag into the target tag.

Single injection
<a href="snippets.html#source" class="inject"> (source and target with same id)
<a href="snippets.html#source" data-inject="#target"> (source and target with different id)
<a href="snippets.html#source" data-inject="#source #target"> (more verbose)

Multiple injection
<a href="snippets.html" data-inject="#source1 #target1 && #source2 #target2">
<a href="snippets.html#source1" data-inject="#target1 && #source2 #target2">
<a href="snippets.html#source2" data-inject="#source1 #target1 && #target2">
<a href="snippets.html#source" data-inject="#target1 && #target2"> (one source into two targets)
same as: <a href="snippets.html#source" data-inject="#target1,#target2">

<a href="snippets.html#default_source data-inject="#target1 && #source2 #target2 && #target3">
#default_source would be used for #target1 and #target3

Binding other patterns
<a href="snippets.html#foo" data-inject="#myWizard.modal"> (target has id #target and class modal)
<a href="snippets.html#foo" data-inject="a#myLink[data-injection='#myTarget']">
<a href="snippets.html#foo" data-inject="a#myLink[data-injection='#myTarget'] && a#anotherLink[data-injection='#anotherTarget']">
<a href="snippets.html#foo" data-inject="a#myLink[data-injection='#myTarget'], a#anotherLink[data-injection='#anotherTarget']"> (one target, that can be passed to jquery and returns two elements to put the content in)

<a href="snippets.html#foo" data-inject="a#myLink[data-injection='#myTarget && #myotherTarget'] && a#anotherLink[data-injection='#anotherTarget']"> (WONTWORK)

<a href="snippets.html#foo" data-inject="#myWizard; class: modal"> (target has id #target, class modal is added to the target tag)

Options
<a href="snippets.html#foo" data-inject="<target_that_either_exist_in_full_or_is_created>">
<a href="snippets.html#foo" data-inject="#myWizard.modal; replace">
~~<a href="snippets.html#foo" data-inject="#ul.modal:after">~~ <-- not supported
~~<a href="snippets.html#foo" data-inject="#ul.modal:before">~~ <-- not supported
<a href="snippets.html#foo" data-inject="ul#myUl; method: append">
<a href="snippets.html#foo" data-inject="ul#myUl; method: prepend">
<a href="snippets.html#foo" data-inject="#myWizard.modal; method: replace">
<a href="snippets.html#foo" data-inject="#myWizard.modal; class: foo; method: append">

Methods
- prepend: insert as first child of matched element (programmer: prepend, designer: before, jquery: prepend)
- append: insert as last child of matched element (programmer: append, designer: after, jquery: append)
- replace: replace matched element
- content: replace content of matched element (default)
- before-tag?: insert before matched element (programmer: before, jquery: before)
- after-tag?: insert after matched element (programmer: after, jquery: after)

General discussion
==================

Patterns syntax new?

collapsible: triggered by class presence
<div class="collapsible">

collapsible is a toggle "open closed" that is auto-triggered if neither open nor closed are set.

toggle pattern: triggered by data-toggle attribute
Simple case: no selector means element itself, no attr defaults to class

```
 <button class="red" data-toggle="values=red blue">Click me</button>
```

Complex case: other element, other attribute

```
<button data-toggle="selector=#other, #moreother ; attr=disabled ; values=disabled">Click me</button>

<button data-toggle="value=disabled && selector=#other ; attr=disabled ; values=disabled">

 <button class="red" data-toggle="values=red blue &  ">Click me</button>


 <a data-confirm="message='An example of using toggle is: 'selector=#other, #moreother ; attr=disabled ; values=disabled''">...</a>

 data-xyz="<directives>"
 directives = <directive>[ && <directives>]
 directive = <option> [; <directive>]



title="bla &quot;sdfsdf&quot;"

<button data-toggle="<toggle_directive_1>;<toggle_some_other_element_too>"
```

Inspiration for syntax

```
something {
	border-style: solid;
	border-width: 1px;
	border-color: red;
}

something {
      border: 1px solid red;
   }
```

```
<button data-toggle="selector=#toRotate!attr=class!value=red green blue">change colour</button>

<a href="foo.html#foo" data-injection="#container -- as -- .modal.loginModal"

<a href="foo.html" data-injection="#foo:first-child.source -- into -- #container -- as --
.modal.loginModal"

<a href="foo.html" data-injection="#foo:first-child.source -- into -- #container -- as -- .tooltip[data-
something='…']"

a href="foo.html" data-injection="#something :: #something; #sthelse :: #sthelse"

<a href="....." class="tooltip" data-tooltip="click" title="Hello my lovely">…</a>
<a href="foo.html#something" data-injection="as -- .tooltip"
<a href="foo.html#something" data-injection="#foo -- as -- .modal"

<div class="modal">

<a hef="#tip" rel="tooltip" data-tooltip="click">...</a>
```

tooltip-click

Parameter requirements:
- boolean options - key only, no value (for example: sticky or click for tooltips)
- valued options: both key and value (for example: attr: disabled for toggle)
- separate options with semi-colon (for example: ajax; click)
- create multiple instances of the same pattern on a single element by separating options with &&
- no support for quoting/escaping in parameters. If human-visible text is needed use a separate attribute or element

an example:

```
data-toggle="selector=#id; attr=class; value=red"
data-toggle="#id; red"
data-toggle=";red"
data-toggle="#id; selector=#other"
--> data-toggle="selector=#id; selector=#other"
```

- always needs selector --> selector is the first

-> positional parameters first (for toggle: selector, value)
-> followed by key/value parameters and boolean parameters

```
data-tooltip="forcePosition ; tm"

data-tooltip="forcePosition; pos: tm"
data-tooltip="forcePosition; tm"
data-tooltip="tm; rt; rm" <- problem: no order of parameters kept
  { tm: true, rt: true, rm: true}
```

data-tooltip="forcePosition ; tm :: rt :: rm"

data-tooltip="pos: tm rt rm forcePosition" <-- CSS like syntax. Parser has knowledge of meaning of property values. Like in CSS: box-shadow: 0 1px 0 white;

data-tooltip="forcePosition=tm rt rm"
data-tooltip="tm rt rm"
data-tooltip="tm rt rm forcePosition"

data-tooltip="tm rt rm"

data-toggle="selector=<selector>;"
data-tooltip="position=<list_of_flags>"

selector="arbitrary selector" directly passed to jquery, therefore the value must not be parsed by the generic parser but instead handed over to the pattern.

values are not to be parsed?
generic parser only to split down to single options and leave it to the pattern to interpret these?

stage 1:
- split &&
- split ;
- handover list to pattern

-> what positional parameters
-> names of positional parameters
-> value type of each parameters (list, single string, boolean)
-> list values: allowed options and allowed booleans

```
parser = new Parser()
parser.add_argument("selector", {required: true, type: "string"})
parser.add_argument("position", {type: "list",
                                "allowed-values": ["lt", "lm", "lb", ...],
                                flags: ["forcePosition"]})
options = parser.parse(this.dataset.tooltip)
```

Possible extension to allow short-hand format combining multiple options in one:

```
parser.add_argument("behaviour": {type: "combined",
                                 parameters: ["method", "step", "speed", "controls", "vertical"]})
```

-> this only works of all options that take a list have fixed allowed values that do not overlap
   and if options that take a list are *after* all other values
   and only one option (the last) can be a list?

data-tooltip="tm forcePosition rt rm" <-- input from messy designer ;)
options = {options: ["tm", "rt", "rm"], forcePosition: true} <-- strict output for javascript

Things that work well:
- classes to trigger behaviour
- data-xyz attributes per pattern

Questions:
- conflict with other class names: do not worry about conflicts for now, assume
  use mostly in new projects, not existing projects that may use same classes.

[12:54:34 PM CEST] Florian Friesdorf: <a href="foo.html#id" class="inject">

<a href="foo.html" data-injection="#id"> Bad, because it does not jump to #id in text browsers

<a href="foot.html#id1#id2" class="inject">

<a href="foot.html#id" class="inject">
<a href="foot.html#id1#id2" class="inject">