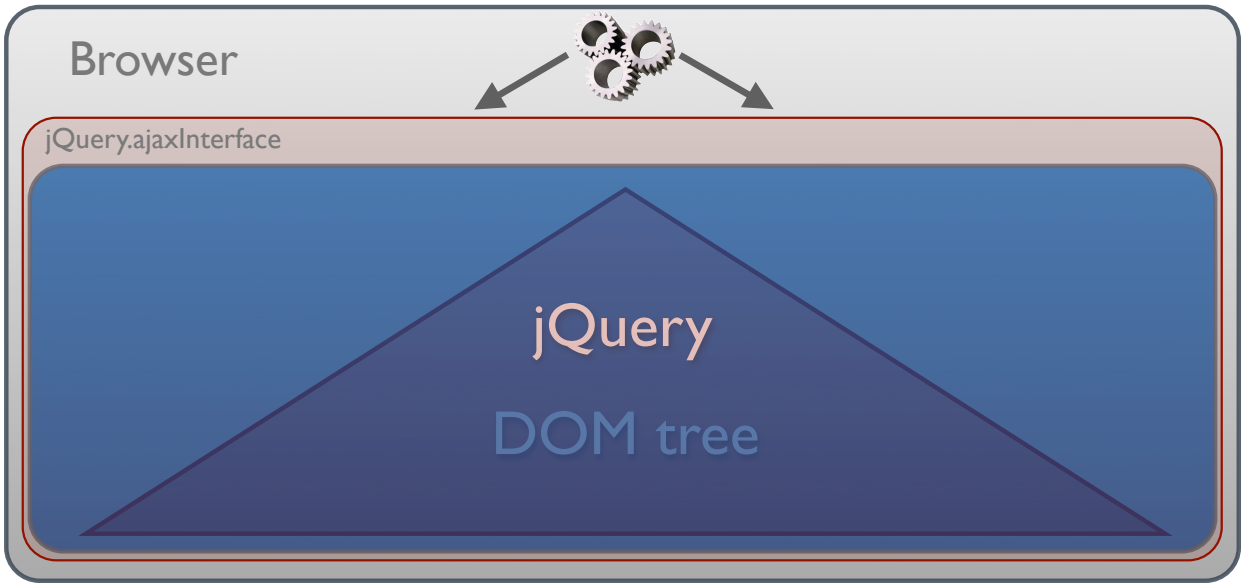
jQuery.ajaxInterface V0.8



Summary:
jQuery.ajaxInterface is a XML wrapper for jQuery.
It's a powerful tool for quickly turning an entire website into an AJAX application and at the same time putting the power of jQuery into the hands of backend developers and template editors.

Idea:
jQuery.ajaxInterface is abstracting the jQuery JavaScript code into an easy readable, editable and first of all generatable XML file.

With the help of a small XML file (e.g. a template) you can replace any frontend content and meta information without writing even one single line of JavaScript code!

Roadmap:
The next version is going to add support for jQuery plugins like jQuery UI.

Example:
This example adds new content to the page and changes the page title:

```
<?xml version="1.0" ?>
<jquery>
  <content>
    <title>new page title</title>
    <modification selector="body">
      <html action="append">
        <![CDATA[
          <h1>new content!</h1>
        ]]>
      </html>
    </modification>
  </content>
</jquery>
```

The only thing you need to do in frontend is to add the library and load the links through the jQuery.ajaxInterface, which is like a tiny layer over the original jQuery Ajax functionality and has exactly the same API—including synchronous and asynchronous requests over POST and GET.

```
<script src="jquery.js"></script>
<script src="jquery.ajaxinterface.js"></script>
```

```
// Turn all links into ajaxInterface requests
$('a').click(function(){
      $.ajaxInterface({url: this.href});
      return false;
});
```

Why?
jQuery.ajaxInterface comes pretty handy, when you have a certain backend functionality and would like to boost the performance of your web application with the help of AJAX.
In this case you can either deliver the content the common way (by generating the whole page with every single request), or create just the altering part describing the exact position it has to be added to with the help of this XML file.
You can even combine both possibilities by adding a parameter (like "ajax=true") to every ajaxInterface request. In this case you could add a simple if-statement to decide whether the whole page needs to be generated or just the altering part.

Documentation

The biggest difference between a normal AJAX request and an jQuery.alaxInterface request is surely its invisible parsing and execution of the XML commands without any developer involvement. The only result you will get in your frontend application code is the "<response />" part of the XML document.

jQuery.ajaxInterface **request API**:

```
// load via ajaxInterface
jQuery.ajaxInterface({
      type: ["get"]/["post"],
      url: [URL],
      async: [false]/[true],
      data: "request=ajax"
});
```

XML document:

```
<?xml version="1.0"?>
+<jquery>
| Root element for the entire jQuery.ajaxInterface document.
|--+<before>
|  | This part is parsed before the actual content is added to the DOM.
|  | Pretty handy to include some CSS rules or run some animations.
|  +</before>
|
|--+<content>
|  | The actual DOM altering content.
|  +</content>
|
|--+<after>
|  | This section is parsed after all the DOM changes where finished.
|  | You can use it for adding JavaScript events.
|  +</after>
|
|--+<response>
|  | This is the response part and will be forwarded into the calling function.
|  | You can use it to pass the backend status to your program.
|  | The inner part is being forwarded as a standalone xml document.
|  +</response>
|
+</jquery>
```

You can place any of the following XML code in one of the first three blocks (before, content and after). The last block is used to pass any backend data to the frontend JavaScript.

Adding and removing CSS files:

```
+<css>
| CSS meta manipulations (adding and removing css files)
|--+ <file action="">path/to/style.file.css</file>
|    action: (String) 'add':       Adds a new CSS file to the DOM.
|                     'remove':    Removes a CSS file from the document.
+</css>
```

Adding JavaScript files:

There is no way to remove the executed JavaScript from a document.

```
+<javascript>
| JavaScript meta manipulations (adding new javascript files)
|--+ <file>path/to/javascript.file.js</file>
+</javascript>
```

Changing the document title:

```
+<title>
| new page title
+</title>
```

Adding, replacing, removing and altering DOM elements:

```
+<modification selector="">
|    selector: (String) You can use any supported
|                                    jQuery selectors for your modifications (CSS or xPath).
|--+<html action="">
|  | This tag is used for any HTML manipulations
|  | (if you want to add, replace or remove DOM elements).
|  | action: (String) 'append':  This command will append the following code
|  |                             to the selected elements
|  |                             (Put your code into CDATA inside of the HTML tag).
|  |                  'replace': This command will replace the content of the selected elements.
|  |                  'remove':  This command is removing the selected elements from the DOM.
|  +</html>
|
|--+<id>
|  | The value of this tag will be set as the ID of the selected element
|  | (make sure that the selector applies for one element only!).
|  +</id>
|
|--+<styles action="">
|  | This tag is performing inline style modifications on the selected elements.
|  | action: (String) 'add':     This command will add the styles.
|  |                  'replace': This code will replace the existing inline
|  |                             styles with the new ones.
|  |                  'remove':  This code is removing any of the inline style informations.
|  +</styles>
|
|--+<class action="">
|  | action: (String) 'add':     This command adds the listed class names to the DOM element.
|  |                  'replace': This command is replacing any of the existing class names with
|  |                             new ones.
|  |                  'remove':  This command will remove the named classes from the elements.
|  |                             Leaving this tag empty will remove all of the existing classes
|  |                             from the element.
|  +</class>
|
|--+<animations event="click">
|  | This tag will perform custom jQuery animations on the selected elements
|  | and can be bound on any event.
|  | event: (String): You don't have to set this attribute if you want
|  |                  to run your animations immediately without waiting for an event to fire.
|  |                  All the events are bound using the jQuery "live" function making sure that they are
|  |                  inherited to any of the new elements matching the given selector.
|  |                  Possible values: blur, focus, focusin, focusout, load, resize, scroll,
|  |                  unload, click, dblclick, mousedown, mouseup, mousemove,
|  |                  mouseover, mouseout, mouseenter, mouseleave, change,
|  |                  select, submit, keydown, keypress, keyup, error
|  |
|  |--+<animation name="" speed="" top="" left="" width="" height="" opacity="">
|  |  | This tag is used to describe the actual animation. You can put any of them
|  |  | into an "animations" tag (equals jQuery method chaining) or by placing them
|  |  | recursive into each other (equals callback calls after each other).
|  |  | name: (String): This is the name of the animation. Possible animations:
|  |  |                   delay: This command delays the following animations for the given time.
|  |  |                         Possible parameters: speed (int: ms)
```

```
|   |   |                 hide: This command is hiding the selected elements.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                 show: This command if showing the selected elements.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                 fadeIn: This command is fading in the selected elements.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                 fadeOut: This command is fading out the selected elements.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                 fadeTo: This command is fading the selected elements to a certain opacity.
|   |   |                       Possible parameters: speed (int: ms), opacity (int: 0-1)
|   |   |                 slideUp: This command is hiding the selected elements with a sliding motion (up).
|   |   |                       Possible parameters: speed (int: ms), opacity (int: 0-1)
|   |   |                 slideDown: This command is hiding the selected elements with a sliding motion (down).
|   |   |                       Possible parameters: speed (int: ms), opacity (int: 0-1)
|   |   |                 moveTo: This command moves an element to a certain position (absolute or relative)
|   |   |                       Possible parameters: speed (int: ms),
|   |   |                                            top  (int: absolute top position, String: "+=px" relative
|   |   |                                                  top position),
|   |   |                                            left (int: absolute left position, String: "+=px" relative
|   |   |                                                  left position)
|   |   |                 resizeTo: This command resizes an element to a certain size (absolute or relative)
|   |   |                           Possible parameters: speed (int: ms),
|   |   |                                            width  (int: absolute width, String: "+=px" relative width
|   |   |                                                    change),
|   |   |                                            width  (int: absolute height, String: "+=px" relative height
|   |   |                                                    change),
|   |   |                 shake: Shake is an animation known from jQuery UI. This command will shake an element
|   |   |                        five times during the given time.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                 animation: If you have to define a more complex animation (e.g. adjusting multiple
|   |   |                            parameters) you can use this generic animation function.
|   |   |                            It will basically forward your JSON values to the jQuery animation function.
|   |   |                       Possible parameters: speed (int: ms)
|   |   |                                            value (String: JSON) Be careful to set the value of this
|   |   |                                                  attribute in single quote,
|   |   |                                            since JSON expects you to user double quote for its values.
|   |   |                 setCSS: This command is setting the CSS of the matched elements and runs the next
|   |   |                         animation.
|   |   |                         This is not an animations and therefor doesn't have a speed parameter.
|   |   |                         Possible parameters: value (String: JSON) Be careful to set the value of this
|   |   |                         attribute in single quote, since JSON expects you to user
|   |   |                         double quote for its values.
|   |--+</animation>
|   |
|  +</animations>
|
+</modification>
```

EXAMPLES:

The **structure** of an ajaxInterface XML document looks like this:

```
<?xml version="1.0"?>
<jquery>

  <before />
  <content />
  <after />
  <response />

</jquery>
```

Explanation:

<before /> - This part is parsed before the actual content is added to the DOM. Pretty handy to include some CSS rules or run some animations.
<content /> - The actual DOM altering content.
<after /> - This section is parsed after all the DOM changes where finished. You can use it for adding event registering JavaScript.
<response /> - This is the response part and will be forwarded into the calling function. You can use it to pass the backend status to your program. The inner part is being forwarded as a standalone xml document.

Examples HEADER:

```
<?xml version="1.0"?>
<jquery>
  <content>
   ALL THE EXAMPLES GO INHERE!
  </content>
</jquery>
```

- Replacing the page title:

jQuery:

```
$("title").text("new page title");
```

jQuery.ajaxInterface:

```
<title>parsing successful</title>
```

- Adding a CSS file to the DOM tree:

jQuery:

```
$("head").append('<link rel="stylesheet" href="path/to/cssfile.css" />');
```

jQuery.ajaxInterface:

```
<css>
  <file action="add">path/to/cssfile.css</file>
</css>
```

- Removing a CSS file from the DOM tree:

jQuery:

```
$("head link[href='path/to/cssfile.css']").remove();
```

jQuery.ajaxInterface:

```
<css>
  <file action="remove">path/to/cssfile.css</file>
</css>
```

- Adding a JavaScript file to the DOM tree:

jQuery:

```
$("body").append('<script type="text/javascript" src="path/to/javaScriptFile.js"></script>');
```

jQuery.ajaxInterface:

```
<javascript>
  <file>path/to/cssfile.css</file>
</javascript>
```

- Removing a JavaScript file from the DOM tree:

There is no way to remove the executed JavaScript from a document.

Examples DOM:

- Adding a new element to the DOM tree:
jQuery:
```
$("body").append("<h1>new content!</h1>");
```

jQuery.ajaxInterface:
```
<modification selector="body">
  <html action="append">
    <![CDATA[
      <h1>new content!</h1>
    ]]>
  </html>
</modification>
```

- Removing an element from the DOM tree:
jQuery:
```
$("h1").remove();
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <html action="remove" />
</modification>
```

- Replacing an element in the DOM tree:
jQuery:
```
$("h1").html("another content");
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <html action="append">
    <![CDATA[
      <h1>another content</h1>
    ]]>
  </html>
</modification>
```

- Adding or replacing an ID of a DOM element:
jQuery:
```
$("h1").attr("id","ID1");
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <id>ID1</id>
</modification>
```

- Adding a CSS class to a DOM element:
jQuery:
```
$("h1").addClass("newClass");
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <class action="add">
    newClass
  </class>
</modification>
```

- Removing a CSS class from a DOM element:
jQuery:
```
$("h1").removeClass("newClass");
```

jQuery.ajaxInterface:

```
<modification selector="h1">
  <class action="remove">
    newClass
  </class>
</modification>
```

- Replacing a CSS class on a DOM element:

jQuery:

```
$("h1").removeClass().addClass("newClass");
```

jQuery.ajaxInterface:

```
<modification selector="h1">
  <class action="replace">
    newClass
  </class>
</modification>
```

- Adding CSS styles to a DOM element:

jQuery:

```
$("h1").css({border: "1px solid red", width: "100px", color: "red"});
```

jQuery.ajaxInterface:

```
<modification selector="h1">
  <styles action="add">
    border: 1px solid blue;
    width: 100px;
    color: red;
  </styles>
</modification>
```

- Replacing CSS styles on a DOM element:

jQuery:

```
$("h1").removeAttr("style").css({border: "1px solid red", width: "100px", color: "red"});
```

jQuery.ajaxInterface:

```
<modification selector="h1">
  <styles action="replace">
    border: 1px solid blue;
    width: 100px;
    color: red;
  </styles>
</modification>
```

- Removing CSS styles from a DOM element:

jQuery:

```
$("h1").removeAttr("style");
```

jQuery.ajaxInterface:

```
<modification selector="h1">
  <styles action="remove" />
</modification>
```

Examples ANIMATIONS:
jQuery.ajaxInterface is also bringing an abstraction for jQuery events and animations.
There are two possible types of animations: the first ones are running immediately on the element, the second ones have to be fired by an event.
A possible XML document for running two animations after each other could look something like this:

```
<?xml version="1.0" ?>
<jquery>
  <content>
    <modification selector="h1">
      <animations>
          <animation name="delay" speed="750">
          <animation name="moveTo" top="+=200" left="+=200" speed="1500">
            <animation name="fadeOut" speed="1500" />
          </animation>
          </animation>
      </animations>
    </modification>
  </content>
</jquery>
```

Every animation needs a referencing modification selector to know which part of the DOM tree to animate, therefore the animations have to be placed inside of this element.
All of the animations have to be surrounded by the `<animations>` tag.
There are two ways to place a multiple number of animations. Placing them one after the other would equal the normal method chaining, however placing them inside of the previous one would execute it in a callback manner.

- Animating an element with method chaining:
jQuery:
```
$("h1").animate({top: "+=200", left: "+=200"}, 1500).animate({width: 300, height: 300}, 1500);
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <animations>
      <animation name="moveTo" top="+=200" left="+=200" speed="1500" />
      <animation name="resizeTo" width="300" height="300" speed="1500" />
  </animations>
</modification>
```

- Animating an element with callbacks:
jQuery:
```
$("h1").animate({top: "+=200", left: "+=200"}, 1500, function(){
  $(this).animate({width: 300, height: 300}, 1500);
});
```

jQuery.ajaxInterface:
```
<modification selector="h1">
  <animations>
      <animation name="moveTo" top="+=200" left="+=200" speed="1500">
        <animation name="resizeTo" width="300" height="300" speed="1500" />
      </animation>
  </animations>
</modification>
```

Now we want to run the animation after an event was fired.

- Animating an element with click event:
jQuery:
```
$("h1").click(function(){
  $(this).animate({top: "+=200", left: "+=200"}, 1500, function(){
    $(this).animate({width: 300, height: 300}, 1500);
  });
});
```

jQuery.ajaxInterface:
The only thing we need to add is the event parameter.
```
<modification selector="h1">
  <animations name="click">
      <animation name="moveTo" top="+=200" left="+=200" speed="1500">
        <animation name="resizeTo" width="300" height="300" speed="1500" />
      </animation>
  </animations>
</modification>
```

- Animating an element with multiple events:
We just have to add an other animations block with an other event.
```
<modification selector="h1">
  <animations name="mouseover">
      <animation name="moveTo" top="+=200" left="+=200" speed="1500">
        <animation name="resizeTo" width="300" height="300" speed="1500" />
      </animation>
  </animations>
  <animations name="click">
      <animation name="fadeOut" speed="1500" />
  </animations>
</modification>
```

Possible events:
blur, focus, focusin, focusout, load, resize, scroll,
unload, click, dblclick, mousedown, mouseup, mousemove,
mouseover, mouseout, mouseenter, mouseleave, change,
select, submit, keydown, keypress, keyup, error

Possible animations:
hide, show, fadeIn, fadeOut, fadeTo, moveTo, resizeTo,
shake

Shake is an animation known from jQuery UI. It shakes your element five times during the given time.

Putting it all together an jQuery.ajaxInterface XML document would look like this:

```
<?xml version="1.0"?>
<jquery>
  <before>
    <css>
      <file action="add">css/style.css</file>
    </css>
  </before>

  <content>
    <modification selector="body">
      <html action="append">
        <![CDATA[
        <h1>new content!</h1>
        ]]>
      </html>
      <html action="append">
        <![CDATA[
        <h2>even newer content</h2>
        <div id="divWithID">huhu</div>
        ]]>
      </html>
    </modification>
       <modification selector="h1">
      <class action="add">
        huhu
      </class>

      <id>
        id1
      </id>

      <styles action="add">
        border: 1px solid blue;
        width: 100px;
        color: red;
      </styles>

      <animations event="click">
       <animation name="shake" speed="1500">
           <animation name="fadeOut" speed="1500" />
      </animations>
    </modification>

    <modification selector="#divWithID">
      <styles action="add">
        border: 2px dotted red;
        width: 100px;
        height: 50px;
        color: blue;
      </styles>

      <animations>
        <animation name="shake" speed="500">
          <animation name="moveTo" top="+=200" left="+=200" speed="1500" />
          <animation name="resizeTo" width="300" height="300" speed="1500" />
          <animation name="fadeOut" speed="1500" />
        </animation>
      </animations>
    </modification>
  </content>

  <after>
    <title>parsing successful</title>

    <javascript>
      <file>test.js</file>
    </javascript>
  </after>

  <response>
    true
  </response>

</jquery>
```