Actions Tech. Note 1

Draft 1.0

Getting & Setting HTML Attributes.

When Freeway generates a page during output it generates a tree-structure of the tags on the page prior to physically generating the HTML file. This tree structure is available to Freeway actions and can be accessed and manipulated using JavaScript.

Introduction

The tree structure that Freeway generates is based on the tags that are generated. So looking at the code that is generated for a page with nothing on it you can see that the page has the following structure:

```
<html>

  <head>
    <title>Untitled</title>
    <meta name="GENERATOR" content="Freeway 3.0">
  </head>

  <body topmargin=8 leftmargin=8 bgcolor="#ffffff">
  </body>

</html>
```

There is an `<html>` tag. The `<html>` tag has no properties. The `<html>`tag has two tags within it. These tags are the `<head>` and the `<body>` tag. The `<head>` tag has no attributes, but contains a `<title>` tag and a `<meta>` tag. The `<body>` tag has no contents but has attributes of `topmargin`, `leftmargin` and `bgcolor`.

For this trivial example Freeway generates a tag-tree that reflects this structure, with the slight difference that there is an extra enclosing tag that is never output that contains the `<html>` tag as it's contents. This tag is present as a convenience, and also to make it possible to have content that is before and after the `<html>` tag.

The tag tree generated by Freeway is partial. What this means is that some parts of the HTML generated by Freeway , that appears as tags are not available as tags in the tag-tree. Currently in Freeway 3.0 the tags generated by HTML text boxes are not available as discrete tags in the tag-stream, and the tags that are responsible for the table structure that is generated by Freeway to position page elements is not available either. It is likely that text will be available as tags in a future release of Freeway but there are currently no plans to make the layout table available.

You can find a given tag on the page by using the `fwFind` method. `fwFind` is a method of the tag object. The tag tree (stream) is available from as the `fwTags` property of the document. Typically you would use the following to reference it within an Action.

```
fwDocument.fwTags
```

so to find a given tag you would use this:

```
var headTag = fwDocument.fwTags.fwFind("head");
```

This will then yield the first `<head>` tag in the tag stream. If there is no `<head>` tag it will return `null`.

You can in fact call `fwFind` from any tag, and the `fwFind` method will search the contents of the tag that you call it from. So:

```
var imgTag = fwDocument.fwTags.fwFind("img");
```

Will set `imgTag` to the first `<img>` tag in the whole document. But

```
var tableTag = fwDocument.fwTags.fwFind("table");
var imgTag = tableTag.fwFind("img");
```

Will set `imgTag` to the first `<img>` tag that is in the first `<table>` tag in the document.

The `fwFind` method is very powerful and can take a number of arguments. The JavaScript reference guide has full details. There is a `fwFindAll` which is very similar but returns an array of all the tags that satisfy the criteria specified.

Getting and Setting Properties

The properties of tags are available as JavaScript attributes. This means that if you wanted to get the `bgcolor` of the body you would access it as

```
bodyTag.bgcolor
```

or

```
bodyTag["bgcolor"]
```

In the normal JavaScript way. So the following code snippet would post the `bgcolour` in an alert.

```
var bodyTag = fwDocument.fwTags.fwFind("body");
alert(bodyTag.bgcolor);
```

If you want to set the attribute you can do so using normal assignment. So you can change the page colour to black `#000000` like so:

```
var bodyTag = fwDocument.fwTags.fwFind("body");
```

```
bodyTag.bgcolor='"#000000"';
```

Note: from the above example you can see that you are responsible for ensuring that the property is correctly quoted.

Should you wish to delete the attribute entirely you can do so by setting it to null:

```
var bodyTag = fwDocument.fwTags.fwFind("body");
bodyTag.bgcolor= null;
```

Some properties are used to contain JavaScript statements. A good example is the `onload` property of the page. This property is used to contain a number of statements that will be executed when the page is loaded. When dealing with this sort of property you should be sensitive to other actions on the page. It may be that other actions on the page will have set the `onload` attribute and if you action simply uses a straight assignment to set it you will overwrite what they have set. This would be a bad thing. What you should do is append to any existing value. So of the body tag was

```
<body onload="foo()">
```

And you wished to add the call `bar()`. You would want the tag to be:

```
<body onload="foo();bar()">
```

You could do this in JavaScript by getting the property (if any), stripping off the quote, adding a semi-colon, adding your call and then replacing the quote. However as this is a common requirement Freeway provides a method `fwAddJToTag` that will do this for you. The following code will handle this situation for you:

```
var bodyTag = fwDocument.fwTags.fwFind("body");
bodyTag.fwAddJToTag("onload","bar()");
```

The situation is slightly complicated by some attributes dealing with a return value. For example the `onclick` attributs of a `<href>` will stop the link from being followed if the `onclick` JavaScript returns false. `fwAddJToTag` will deal with the simple cases where a statement contains return true, or retirn false by ensuring they appear at the end of the sequence of statements.