Dynamic Page Generation

Basically in the e-government mobile standard framework, DOM is created dynamically when an event takes place. This means that you can receive data only when an event takes place. The method to receive data value in JSON, not in HTML, is defined.

1. Create data in JSON format.

```
var categoryData = {
  animals: {
    name: "Animals",
    description: "All your favorites from aardvarks to zebras.",
    items: [
      {
        name: "Pets",
      },
      {
        name: "Farm Animals",
      },
      {
        name: "Wild Animals",
      }
    ]
  },
  colors: {
    name: "Colors",
    description: "Fresh colors from the magic rainbow.",
    items: [
      {
        name: "Blue",
      },
      {
        name: "Green",
      },
      {
        name: "Orange",
      },
      {
        name: "Purple",
      },
      {
        name: "Red",
      },
```

```
        {
          name: "Yellow",
        },
        {
          name: "Violet",
        }
      ]
    },
    vehicles: {
      name: "Vehicles",
      description: "Everything from cars to planes.",
      items: [
        {
          name: "Cars",
        },
        {
          name: "Planes",
        },
        {
          name: "Construction",
        }
      ]
    }
};
```

2. Declare the tag, in the body, which will receive values in JSON.

Internal links

```
<h2>Select a Category Below:</h2>
<ul data-role="listview" data-inset="true">
  <li><a href="#category-items?category=animals">Animals</a></li>
  <li><a href="#category-items?category=colors">Colors</a></li>
  <li><a href="#category-items?category=vehicles">Vehicles</a></li>
</ul>
```

External links

```
<h2>Select a Category Below:</h2>
<ul data-role="listview" data-inset="true">
  <li><a href="/guide/api/json/animals.do">Animals</a></li>
  <li><a href="/guide/api/json/colors.do">Colors</a></li>
```

```
    <li><a href="/guide/api/json/vehicles.do">Vehicles</a></li>
</ul>
```

3. Write a script that will handle the data received after the pagebeforechange event is bound.

```
// Listen for any attempts to call changePage().
$(document).bind( "pagebeforechange", function( e, data ) {
    // We only want to handle changePage() calls where the caller is
    // asking us to load a page by URL.
    if ( typeof data.toPage === "string" ) {

        // We are being asked to load a page by URL, but we only
        // want to handle URLs that request the data for a specific
        // category.
        var u = $.mobile.path.parseUrl( data.toPage ),
            re = /^#category-item/;

        if ( u.hash.search(re) !== -1 ) {

            // We're being asked to display the items for a specific category.
            // Call our internal method that builds the content for the category
            // on the fly based on our in-memory category data structure.
            showCategory( u, data.options );

            // Make sure to tell changePage() we've handled this call so it doesn't
            // have to do anything.
            e.preventDefault();
        }
    }
});
// Load the data for a specific category, based on
// the URL passed in. Generate markup for the items in the
// category, inject it into an embedded page, and then make
// that page the current active page.
function showCategory( urlObj, options )
{
    var categoryName = urlObj.hash.replace( /.*category=/, "" ),

        // Get the object that represents the category we
        // are interested in. Note, that at this point we could
        // instead fire off an ajax request to fetch the data, but
```

```
    // for the purposes of this sample, it's already in memory.
    category = categoryData[ categoryName ],

    // The pages we use to display our content are already in
    // the DOM. The id of the page we are going to write our
    // content into is specified in the hash before the '?'.
    pageSelector = urlObj.hash.replace( /\?.*$/, "" );

if ( category ) {
    // Get the page we are going to dump our content into.
    var $page = $( pageSelector ),

        // Get the header for the page.
        $header = $page.children( ":jqmData(role=header)" ),

        // Get the content area element for the page.
        $content = $page.children( ":jqmData(role=content)" ),

        // The markup we are going to inject into the content
        // area of the page.
        markup = "<p>" + category.description + "</p><ul data-role='listview' data-inset='true'>",

        // The array of items for this category.
        cItems = category.items,

        // The number of items in the category.
        numItems = cItems.length;

    // Generate a list item for each item in the category
    // and add it to our markup.
    for ( var i = 0; i < numItems; i++ ) {
        markup += "<li>" + cItems[i].name + "</li>";
    }
    markup += "</ul>";

    // Find the h1 element in our header and inject the name of
    // the category into it.
    $header.find( "h1" ).html( category.name );

    // Inject the category items markup into the content element.
    $content.html( markup );
```

```
        // Pages are lazily enhanced. We call page() on the page
        // element to make sure it is always enhanced before we
        // attempt to enhance the listview markup we just injected.
        // Subsequent calls to page() are ignored since a page/widget
        // can only be enhanced once.
        $page.page();

        // Enhance the listview we just injected.
        $content.find( ":jqmData(role=listview)" ).listview();

        // We don't want the data-url of the page we just modified
        // to be the url that shows up in the browser's location field,
        // so set the dataUrl option to the URL for the category
        // we just loaded.
        options.dataUrl = urlObj.href;

        // Now call changePage() and tell it to switch to
        // the page we just modified.
        $.mobile.changePage( $page, options );
    }
}
```

Examples

There are two types of processing the received data; internal link and external link.

Internal link Dynamic Page Example
External link Dynamic Page Example