fluid.fetchResources

fluid.fetchResources()

Accepts a hash of structures with free keys, where each entry has either href or nodeId set - on completion, callback will be called with the populated structure with fetched resource text in the field `resourceText` for each entry.

```
fluid.fetchResources(resourceSpecs, callback, options);
```

File name: `FluidRequests.js`

Parameters

`resourceSpecs`	(Object) An object describing the resources to be fetched. The structure of this object is described below.
`callback`	(Function) The function to call once all resources have been loaded.
`options`	(Object) New in v1.3: Options controlling the behaviour of the function.

Return Value

none

See Also

- fluid.fetchResources.primeCacheFromResources
- fluid.fetchResources.clearCache
- fluid.parseTemplates

Resource Specification

The first argument to `fluid.fetchResources` is an object describing the resources to be fetched. The keys in `resourceSpecs` represent a unique key to represent each resource specification. The resource specifications must have one of the following fields:

Field	Purpose
href	A full resolvable URL holding a template or template fragment. This will be fetched via AJAX and used to populate the `resourceText` entry. NOTE that if this value is provided, the `nodeId` will be ignored.
nodeId	The id of a node within the current document holding a template, for which `innerHTML` is to be treated as a template. NOTE that if `href` is provided, this value will be ignored.

On conclusion of all of the fetches, the function `callback` will be called with the now filled-in `resourceSpecs` structure as argument. The `resourceSpecs` structure will have the following fields added to it:

Field	Purpose
baseURL	Computed from href, in order to rebase local URLs in the template.
resource Text	The full text of the template fragment, as a string
fetchError	Filled in if the AJAX request to fetch the resource failed. It will be populated with a structure `status:` response status, `textStatus:` textual version of status, and `errorThrown:` holding details of an exception.
queued	A reserved flag by the system to track the progress of AJAX calls.

The filled-in `resourceSpecs` structure is suitable for passing directly as the argument to the low-level function `fluid.parseTemplates`.

Callback

The argument passed to `fluid.fetchResources` as the `callback` should be a function that accepts a single parameter: the filled-in `resourceSpecs` object. The fields of this object are as described in the previous section.

Options

The following options to the function can be used to customize the behaviour of `fluid.fetchResources`:

Name	Description	Values	Default
`amalgamateClasses`			

Example

```
var myResourceSpecs = {
    bodyTemplate: {
        href: "templates/Body.html"
    },
    sidebarTemplate: {
        href: "templates/Sidebar.html"
    }
};
var myCallback = function (returnedResourceSpecs) {
    // very simple: inject the fetched HTML into the DOM
    $(".bodyNode").html(returnedResourceSpecs.bodyTemplate.resourceText);
    $(".sidebarNode").html(returnedResourceSpecs.sidebarTemplate.resourceText);
};
fluid.fetchResources(myResourceSpecs, myCallback);
```

Example

```
var myResourceSpecs = {
    template1: {
        href: "html/template1.html"
    },
    template2: {
        href: "html/template1.html"
    },
    data: {
        href: "data/clientData.json"
    }};
var myCallback = function (returnedResourceSpecs) {
    for (var key in returnedResourceSpecs) {
        // check for errors before proceeding
        if (returnedResourceSpecs[key].fetchError) {
            // log the failed fetch
            fluid.log("Error loading resource " + returnedResourceSpecs[key].href);
            fluid.log("status: " + returnedResourceSpecs[key].fetchError.status +
                    ", textStatus: " + returnedResourceSpecs[key].fetchError.textStatus +
                    ", errorThrown: " + returnedResourceSpecs[key].fetchError.errorThrown);
        } else {
            // process successfully loaded resource
            ...
        }
    }
};
fluid.fetchResources(myResourceSpecs, myCallback);
```