

## Simple Use Case: Get an External Script

With the **Get Utility** on the page, you can bring in an external script file in two steps:

1. Define the logic you want to execute when the script successfully loads;
2. Get the script.

```
var successHandler = function(oData) {
    //code to execute when all requested scripts have been
    //loaded; this code can make use of the contents of those
    //scripts, whether it's functional code or JSON data.
}

var aURLs = [
    "/url1.js", "/url2.js", "/url3.js" //and so on
];

YAHOO.util.Get.script(aURLs, {
    onSuccess: successHandler
});
```

## Usage: YAHOO.util.Get.script()

```
YAHOO.util.Get.script(str or arr url[s][, obj options])
```

**Arguments:**

- (1) **URL[s]:** A string or array of strings containing the URL[s] to be inserted in the document via script nodes.
- (2) **options:** An object containing any configuration options you'd like to specify for this transaction. See the **Configuration Options** table for the full list of options.

**Returns:**

- (1) **Transaction Object:** Object containing single field, *str* tld, a unique string identifying this transaction.

**Note:** Scripts downloaded will be executed immediately; only use this method to procure JavaScript whose source is trustworthy beyond doubt.

## Usage: YAHOO.util.Get.css()

```
YAHOO.util.Get.css(str or arr url[s][, obj options])
```

**Arguments:**

- (1) **URL[s]:** A string or array of strings containing the URL[s] to be inserted in the document via link nodes.
- (2) **options:** An object containing any configuration options you'd like to specify for this transaction. See the **Configuration Options** table for the full list of options.

**Note:** Returns the same Transaction Object as `script()`; see above.

## Configuration Options\*

Field	Type	Description
onSuccess	fn	Callback method invoked when the requested file(s) have loaded successfully.
onFailure	fn	Callback method invoked when an error is detected or <code>abort</code> is called.
onTimeout	Fn	Callback method invoked when a resource doesn't load within the timeframe specified by the <code>timeout</code> config.
timeout	Int	The number of millisecond to wait for a resource to load.
win	obj	The window into which the loaded resource(s) will be inserted.
scope	obj	The execution scope in which the callbacks will run.
data	any	Data to pass as an argument to all callbacks.
autopurge	bool	If <code>true</code> , script nodes will automatically be removed every 20 transactions (configurable via <code>YAHOO.util.Get.PURGE_THRESH</code> property. Default: <code>true</code> for script nodes, <code>false</code> for CSS nodes.
varName	arr (of strings)	Safari 2.x does not reliably report the load-complete status of script nodes; use this property to provide Get with a globally accessible property that will be available when the script has loaded. This array is parallel to the <code>urls</code> array passed in as the first argument to <code>script()</code> .
insertBefore	str el	Element reference or id of a node that should be used as the insertion point for new nodes. This is useful for making sure CSS rules are parsed in the correct order (place your style overrides in a single style block and <code>insertBefore</code> this node).
charset	str	The charset attribute for new node(s). Default: <code>utf-8</code>

## Callback Arguments

Fields available in the object passed to your `onSuccess` or `onFailure` callback.

Field	Type	Description
tld	str	The unique identifier for this transaction; this string is available as the <code>tId</code> member of the object returned to you upon calling the <code>script</code> or <code>css</code> method.
data	any	The <code>data</code> field you passed to your configuration object when the <code>script</code> or <code>css</code> method was called. Default: <code>null</code> .
win	obj	The window into which the loaded resource(s) were inserted.
nodes	array	An array containing references to node(s) created in processing the transaction. These will be script nodes for JavaScript and link nodes for CSS.
purge	Fn	Calling the returned <code>purge()</code> method will immediately remove the created nodes.

## Solutions

Set up a transaction making use of configuration options; then make use of the data passed to the callback handler:

```
var successHandler = function(o) {
    //o contains all of the fields described in the callback args table
    o.purge(); //removes the script node immediately after executing;
    YAHOO.log(o.data); //the data passed in configobject
}

var objTransaction = YAHOO.util.Get.script("http://json.org/json.js",
    { onSuccess: successHandler,
      scope: this, //successHandler will run in the scope of "this"
      data: {field1: value1, field2: value2} //you can pass data in
    //any format here
    });
```

## YAHOO.util.Get Methods

**css**(string | arr *URLs*[, obj *config options*]) see [usage at right](#)  
**script**(string | arr *URLs*[, obj *config options*]) see [usage at right](#)  
**abort**(string | obj *tId*) takes either the transaction id or transaction object generated by `script` or `css`; aborts transaction if possible; fires `onFailure` handler

## YAHOO.util.Get Global Configuration Properties

**YAHOO.util.Get.POLL\_FREQ** int  
 when polling is necessary to check on the status of a loading file (eg, where the load event is unreliable), this controls the polling interval in milliseconds  
**YAHOO.util.Get.PURGE\_THRESH** int  
 controls the number of added script or link nodes that will accumulate prior to being automatically purged

## Dependencies

The Get Utility requires only the YAHOO Global Object.

\* Use configuration options by passing an optional object containing config options to `YAHOO.util.Get.script` or `YAHOO.util.Get.css` as the second argument:  
`YAHOO.util.Get.script("http://json.org/json.js", {onSuccess: function(o) {YAHOO.log("success!");}});`