

Simple Use Case

Script:

```
var myDataSource = new YAHOO.util.DataSource
  ({name:"a",id:"1"}, {name:"b",id:"2"});
myDataSource.responseType =
  YAHOO.util.DataSource.TYPE_JSARRAY;
myDataSource.responseSchema = {fields:["name","id"]};
```

Instantiates a new DataSource object, `myDataSource`, which manages data retrieval for use by other widgets.

Constructors

```
YAHOO.util.LocalDataSource(mixed data[, obj
  configurations]);
```

- (1) **data (array)**: A JavaScript array of strings.
- (2) **Configuration object (object)**: An optional object literal defines property values of a DataSource instance.

```
YAHOO.util.FunctionDataSource(fn function[, obj
  configurations]);
```

- (1) **JS Function (fn)**: A JavaScript function which returns an array of strings.
- (2) **Configuration object (object)**: See above.

```
YAHOO.util.ScriptNodeDataSource(str uri[, obj
  configurations]);
```

- (1) **URI**: URI to the script location that will return data.
- (2) **Schema (array)**: Schema description of server response data.
- (3) **Configuration object (object)**: See above.

```
YAHOO.util.XHRDataSource(str uri[, obj configurations]);
```

- (1) **Script URI (string)**: Server URI (local domains only – use a proxy for remote domains).
- (2) **Schema (array)**: Schema description of server response data.
- (3) **Configuration object (object)**: See above.

```
YAHOO.util.DataSource(mixed data[, obj configurations]);
```

- (1) **Script URI (string)**: Server URI (local domains only – use a proxy for remote domains).
- (2) **Schema (array)**: Schema description of server response data.
- (3) **Configuration object (object)**: See above.

Key Configuration Properties

Property	Description
<code>responseType</code>	Determines which parsing algorithm to use on response data.
<code>responseSchema</code>	Determines what data gets parsed out of response for consumption.

Interesting Moments

Event	oArgs passed to handler
<code>cacheFlushEvent</code>	none
<code>cacheRequestEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.callback {obj}</code> The callback object
<code>cacheResponseEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.response {obj}</code> The response object <code>oArgs.callback {obj}</code> The callback object
<code>dataErrorEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.callback {obj}</code> The callback object <code>oArgs.message {str}</code> Error message
<code>requestEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.callback {obj}</code> The callback object <code>oArgs.tld {int}</code> Unique transaction ID
<code>responseCacheEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.response {obj}</code> The response object <code>oArgs.callback {obj}</code> The callback object
<code>responseEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.response {obj}</code> The response object <code>oArgs.callback {obj}</code> The callback object <code>oArgs.tld {int}</code> Unique transaction ID
<code>responseParseEvent</code>	<code>oArgs.request {obj}</code> The request object <code>oArgs.response {obj}</code> The response object <code>oArgs.callback {obj}</code> The callback object

Subscribe to DataSource Custom Events on your DataSource instance:
`myDS.subscribe("requestEvent", myFn);`

Abstract Methods

Method	Description
<code>doBeforeCallback</code>	This overridable abstract method gives implementers an opportunity to access the data before it has been cached or returned to the callback. Implementers should be sure to return data in a ready-to-return state to avoid errors.
<code>doBeforeParseData</code>	This overridable abstract method gives implementers an opportunity to munge the data before it is schema-parsed. Implementers should be sure to return data in a ready-to-parse state to avoid errors.

Dependencies

DataSource requires the YAHOO Global Object and the Event Utility. Connection Manager (for XHRDataSource), the Get Utility (for ScriptNodeDataSource), and the JSON Utility (for JSON data) are optional.

YAHOO.util.
DataSourceBase
Properties:

dataType (int)
liveData (mixed)
responseSchema (obj)
responseType (int)

YAHOO.util.XHRDataSource
Properties:

connMethodPost (b)
connMgr
(YAHOO.util.Connection)
connTimeout (int)
connXhrMode
("queueRequests" |
"cancelStaleRequests" |
"ignoreStaleResponses"
| "allowAll")

YAHOO.util.ScriptNodeDataSource
Properties:

asyncMode
("ignoreStaleResponses" | "allowAll")
getUtility
(YAHOO.util.Get)
scriptCallbackParam
(str)

Data Flow in DataSource

