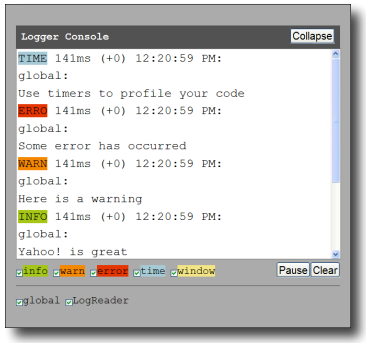
YUI Library: Logger

2007-12-4 **v2.4**

Simple Use Case (LogReader)

```
<div id="myLogger"></div>
<script>
var myLogReader = new
   YAHOO.widget.LogReader
   ("myLogger");
</script>
```



Instantiates a new LogReader object, myLogReader, which is bound to a div whose id attribute is 'myLogger'. The result will be a visual LogReader display.

To create a LogReader that floats outside the page context, omit the reference to a context div. Your LogReader will then be appended to the page and positioned absolutely. If the YUI Drag & Drop Library is included on the page, it will be draggable.

Constructor (LogReader)

```
YAHOO.widget.LogReader([str html id | obj element
   reference, obj configuration object]);
```

Arguments:

(1) **HTML element (string or object):** An optional reference to an HTML id string or element object binds the LogReader to an existing page element.

(2) **Configuration object (object):** An optional object literal defines LogReader settings. All properties of a LogReader instance can be set via the constructor by using this object.

Logging via `console.log()`

A growing number of browsers and extensions support the JavaScript method `console.log()`. The excellent FireBug extension to FireFox supports this method, as does the JavaScript console in Apple's Safari browser. Enable this feature using Logger's `enableBrowserConsole()` method.

Dependencies

Logger requires the YAHOO object, Dom, and Event; Drag & Drop is optional. Use in combination with –debug versions of YUI files for built-in logging from components.

Simple Use Case (Logger)

```
YAHOO.log("My log message", "error", "mysource");
```

Logs a message to the default console and to `console.log()`, if enabled; the source is "mysource" and the category is "error". Custom categories and sources can be added on the fly.

Constructor (LogWriter)

Creates a separate, named bucket for your log messages:

```
YAHOO.widget.LogWriter(str sSource);
```

Arguments:

(1) **Source (string):** The source of log messages. The first word of the string will be used to create a LogReader filter checkbox. The entire string will be prepended to log messages so they can be easily tracked by their source.

Solutions

Log a message using a pre-styled logging category:

```
YAHOO.log("My log message.", "warn");
```

Create a new logging category on the fly:

```
YAHOO.log("My log message.", "myCategory");
```

Style a custom logging category in CSS:

```
.yui-log .myCategory {background-color:#dedede;}
```

Log a message, **creating a new "source" on the fly**:

```
YAHOO.log("My log message.", "warn", "newSource");
```

In script, **hide and show** the logging console:

```
myLogReader.hide();
myLogReader.show();
```

In script, **pause and resume** output to the console:

```
myLogReader.pause();
myLogReader.resume();
```

Instantiate your own LogWriter to write log messages categorized by their source:

```
MyClass.prototype.myLogWriter = new
   YAHOO.widget.LogWriter("MyClass of MyApp");
var myInstance = new MyClass();
myInstance.myLogWriter.log("This log message can now
   be filtered by its source, MyClass."); //"MyClass
   of MyApp", the full name of the source, will be
   prepended to the actual log message
```

YAHOO.widget.Logger Static Properties:

loggerEnabled (b)
maxStackEntries (int)

YAHOO.widget.Logger Static Methods:

log(sMsg, sCategory, sSource)
disableBrowserConsole()
enableBrowserConsole()
getStack()
getStartTime()
reset()

YAHOO.widget.Logger Custom Events:

categoryCreateEvent
sourceCreateEvent
newLogEvent
logResetEvent

LogReader Properties:

verboseOutput (b)
newestOnTop (b)
thresholdMax (int)
thresholdMin (int)
outputBuffer (int)

LogReader Methods:

hide()/show()
pause()/resume()
collapse()/expand()
clearConsole()
hideCategory()/ showCategory()
hideSource()/ showSource()

LogWriter Methods:

log(sMsg, sCategory, sSource**)**

Categories

info
warn
error
time
window

(Pass in other categories to `log()` to add to this list.)