

## Simple Use Case: Profiler Object

To use Profiler, register your target functions with the `YAHOO.tool.Profiler` object and then call the function as you would normally:

```
var object = {
    method: function(){
    }
};

//register the function
YAHOO.tool.Profiler.registerFunction(
    "object.method", object);

//call the function
object.method();
```

## Usage: YAHOO.registerFunction()

```
YAHOO.tool.Profiler.registerFunction(str name[,
    obj owner])
```

*Arguments:*

- (1) **name:** A string containing the fully-qualified name of the function (e.g. `myobject.mymethod`). Profiler knows to extract everything after the last dot as the short function name.
- (2) **owner:** The object that owns the function (e.g. `myobject` for `myobject.mymethod`). This argument may be safely omitted if the `name` exists in the global scope.

**Note:** Only functions that exist on objects can be profiled. Global functions are considered properties of the window object, so they can be registered but functions declared inside of other functions cannot be registered unless attached to an object.

## Usage: YAHOO.registerConstructor()

```
YAHOO.tool.Profiler.registerConstructor(str
    name[, obj owner])
```

*Arguments:*

- (1) **name:** A string containing the fully-qualified name of the constructor (e.g. `YAHOO.widget.Menu`). Profiler knows to extract everything after the last dot as the short constructor name.
- (2) **owner:** The object that owns the function (e.g. `YAHOO.widget` for `YAHOO.widget.Menu`). This argument may be safely omitted if the `name` exists in the global scope.

**Note:** Only constructors that exist on objects can be profiled. Global functions are considered properties of the window object, so they can be registered but functions declared inside of other functions cannot be registered unless attached to an object.

## Function Report Object

When `YAHOO.tool.getFunctionReport()` is called, an object with the following properties is returned.

Member	Type	Description
avg	float	The average amount of time (in milliseconds) that the function took to execute.
calls	int	The number of times that the function was called.
min	float	The average amount of time (in milliseconds) that the function took to execute.
max	float	The average amount of time (in milliseconds) that the function took to execute.
points	float[]	An array containing the actual execution times (in milliseconds) of the function.

## Usage: YAHOO.registerObject()

```
YAHOO.tool.Profiler.registerObject(str name[, obj
    object[, bool recurse]])
```

Use `registerObject` to register all of the methods on an object (use `registerFunction` to register a single method).

*Arguments:*

- (1) **name:** A string containing the fully-qualified name of the object (e.g. `myobject` or `YAHOO.util.Dom`).
- (2) **object:** The object represented by the `name`. This argument may be safely omitted if the `name` exists in the global scope.
- (3) **recurse:** Indicates if object properties should also be should also have their methods registered.

## Solutions

The basic use case of Profiler is to register one or more functions, run the application as you normally would, retrieve information about specific functions (or a complete report), and then unregister the functions (a necessary step to clean up memory if you wish to persist the browser session).

```
//register the function
YAHOO.tool.Profiler.registerFunction(
    "object.method", object);

//call the function
object.method();

//get specific function information
var calls =
    YAHOO.tool.Profiler.getCallCount("object.method");
var avg = YAHOO.tool.Profiler.getAverage("object.method");
var min = YAHOO.tool.Profiler.getMin("object.method");
var max = YAHOO.tool.Profiler.getMax("object.method");

//get all function information
var report =
    YAHOO.tool.Profiler.getFunctionReport("object.method");
```

## YAHOO.tool.Profiler Registration Methods

**registerConstructor**(string *name*, func *owner*) registers a constructor for profiling

**registerFunction**(string *name*, func *owner*) registers a function for profiling

**registerObject**(string *name*, obj *object*, bool *recurse*) registers all methods on an object for profiling

**unregisterConstructor**(string *name*) unregisters a constructor that was previously registered

**unregisterFunction**(string *name*) unregisters a function that was previously registered

**unregisterObject**(string *name*) unregisters all methods on an object that were previously registered

## YAHOO.tool.Profiler Reporting Methods

**getAverage**(str *name*) returns the average amount of time (in ms) the function with the given name took to execute

**getCallCount**(str *name*) returns the number of times that the given function was called

**getMax**(str *name*) returns the maximum amount of time (in ms) the function with the given name took to execute

**getMin**(str *name*) returns the minimum amount of time (in ms) the function with the given name took to execute

**getFunctionReport**(str *name*) returns an object containing all information about a given function including call count and average, min, and max calls times

**getFullReport**(func *filter*) returns an object containing profiling information for all registered functions

## Dependencies

Profiler requires only the YAHOO Global Object.