JavaScript Code Conventions

Reference

- Code Conventions for the JavaScript Programming Language

Formatting

Line Length

Avoid lines longer than 80 characters. When a statement will not fit on a single line, it may be necessary to break it. Place the break after an operator, ideally after a comma. A break after an operator decreases the likelihood that a copy-paste error will be masked by semicolon insertion. The next line should be indented 8 spaces.

Quote

Use double quote.

Indentation

JavaScript code should be indented 4 spaces.

Variables

All variable names should begin with a letter and use camel case (first letter of each

new word is capitalized). Do not use underscores in variable names. Example:

```
personName
```

Variables should indicate their type in one of the following three ways:

Type	Usage	Example	Notes
HTMLElement	variable**El**	bodyEl	
Node	variable**Node**	bodyNode	
NodeList?	variable**Nodes**	itemNodes	
Array	variable**Arr**	itemArr	
Boolean	**is**Variable, **has**Variable	isLocked	
Date	variable**Time**	startTime	
Event Handler	variable**Handler**	linkClickHandler	
Object	(N/A)		
Int	(N/A)		
String	(N/A)		

Constants

Variables or property names that are intended to be constant values (that is, they should not be changed later), should be in all uppercase with words separate by underscores. Example:

```
MAXIMUM_TIMEOUT
```

Constants should always be used whenever dealing with:

- URLs - these may change in the future, it's best to keep them out of your business logic.
- Settings - things that control the UI or other settings that may change in the future.
- Repeated strings - any string that is used in your code more than once should be a constant.

Objects

Properties

Public property names should be formatted in camel case but not in Hungarian notation. This helps to keep the public API clean. For example:

```
this.name = "My";  //correct
this.sName = "My";  //avoid!!
```

Private property names should be formatted in the same way but should be prefixed with an underscore:

```
this._name = "My";
```

Module properties are considered to be public so using the underscore is not necessary.

Methods

Public method names should be formatted in camel case but not in Hungarian notation. For example:

```
MyObject.prototype.getName = function () { ... }    //correct
MyObject.prototype.fnGetName = function () { ... }  //avoid!!
```

All methods should include trace information using **Y.log**. The format should be as follows:

```
Y.log("methodName(): message", "debug", "source");
```

The method name should always appear first, followed be a message (such as "entering"). The second argument should be kept as "debug". The third argument should be the source of the message (for example, "WeatherModule" or "Application"), which aids in filtering the trace messages.

Public methods, minimally, must have a *Y.log* statement as the first line of the method, specifying the arguments that were passed in. For example:

```
MyObject.prototype.setName = function (name) {
    Y.log("setName(): Setting name to " + name, "debug", "MyObject");
    this.name = name;
}
```

All references to **Y.log()** and **this.Y.log()** will be stripped out as part of the build process.

Coding Practices

English Only

You shouldn't write any non-english characters including comments. For prompting message functions such as alert(), you should use language variables (e.g. MUCHIII.Bar.Lang.navigation.someText) instead of writing it directly.

Declare Variables at the Beginning of Functions

Whenever possible, all variables used in a function should be declared at the beginning of the function using **var**.

Event Delegation

Each view should have only one =onclick= event handler. This can be attached on the view node itself and should handle all clicks for the view:

```
var body = moduleapi.getViewNode("default");
body.on("click", this.handleClick, this, true);
```

Create Shortcut Variables for Object Properties

When an object property is referenced two or more times within a function, create a shortcut variable for it. This reduces the amount of code, enables the YUI Compressor to perform variable replacement on it, and reduces the lookup time for the value (each property lookup is more expensive than a property lookup). For example:

```
this.body.on("click", this.handleClick, this, true);
this.body.firstChild.set("class", "selected");
```

This code should be rewritten as follows:

```
var body = this.body;
body.on("click", this.handleClick, this, true);
body.firstChild().set("class", "selected");
```

Avoid Using Globals

Global variables such as **window** and **document** should not be used or assumed to

be available. It's possible that modules may need to work in a locked-down environment where such variables are not accessible. You should be able to do everything necessary using just facilities provided by modules.

Condition Operator Usage

The condition operator (=?:=) should be used only to assign variables and never for code that doesn't assign a value.

```
//proper use
var value = (condition ? 100 : 200);

//AVOID!!! Doesn't assign a value - use if statement instead
(condition ? doSomething() : null);
```

Any time a condition doesn't return a value, an **if** statement should be used instead.

Function Declarations

All functions should be declared before they are used. Inner functions should follow the var statement. This helps make it clear what variables are included in its scope.

There should be no space between the name of a function and the ((left parenthesis) of its parameter list. There should be one space between the) (right parenthesis) and the { (left curly brace) that begins the statement body. The body itself is indented four spaces. The } (right curly brace) is aligned with the line containing the beginning of the declaration of the function.

```
function getData() {
}

var getData = function () {
}
```

=== and !== Operators.

It is almost always better to use the === and !== operators. The == and != operators do type coercion. In particular, do not use == to compare against falsy values.

Statements

All block statements, regardless of the number of lines, should use curly braces. There should be a line break after the left curly brace and the lines inside should be

indented. For example:

```
if (true) {
    doSomething();
}

if (true) doSomething();    //avoid!!

if (true)
    doSomething();            //avoid!!
```

Take if Statement as example, the class of statements should have the following form:

```
if (condition) {
    statements
}

if (condition) {
    statements
} else {
    statements
}

if (condition) {
    statements
} else if (condition) {
    statements
} else {
    statements
}
```

A for class of statements should have the following form:

```
for (initialization; condition; update) {
    statements
}

for (var i = 0, j = items.length; i < j; i++) {
    statements
}

for (var variable in object) {
    statements
}
```