JAVASCRIPT TEMPLATE

About the example templates

While the W3C and ECMAscript (Javascript) standards continue to evolve, browser compliance with any particular version, at any given time, is a moving target. All vendors have added their own tweaks (additional features?) to the standards, which by their very definition, are non standard; so should be avoided if the goal is cross browser compatibility.
The examples therefore focus on standard features, ignoring any cross browser differences, except where specifically mentioned. They should work with the majority of *modern* browsers. The extent to which these examples work with your favourite browser, is perhaps indicative of its standards compliance

Since Javascript needs to run in a web page, the following HTML code will be used as a container for the various examples. This is very similar to the code most browsers provide when you open a blank page. The main change to make is to add the script tags.

In the examples which follow, where you see the script and magnifying glass icon, you should be able to click on this and run the examples in a separate window. Initially though you should type out examples yourself, in order to get practice at debugging!.
If having done this the script still doesn't work, then try running the included example, to confirm whether you have a bug (or two) or not. If the included script doesn't work either, it may be because you have a 'non standard' browser, or of course maybe I messed up!

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>javascript test page</title>
<link href="./allStyles.css" rel="stylesheet" type="text/css" />
<script type="text/javascript">
 /* this is where your javascript starts */

</script>
</head>
<body>
<h2>Subject</h2>
<h3>Introduction</h3>
<!-- This is where your HTML code starts -->
<h4>Program output</h4>
<div id="output">
```

```
 </div>
 </body>
 </html>
```

The sample page below is designed for the XHTML standard. You can use either, but if you don't intent to use XHTML then the simpler version above is perfectly adequate. It also includes some additions to the HTML code, which are used in subsequent examples. Some simple styling has been added as well.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Example template</title>
<link href="./allStyles.css" rel="stylesheet" type="text/css" />
<script type="text/javascript">
//<![CDATA[
function test()
  /* this is where your javascript starts */
//]]>
</script>
<style type="text/css">
/*<![CDATA[*/
 /* additional style information here */
/*]]>*/
</style>
</head>
<body>
<h2>Subject</h2>
<h3>Introduction</h3>
 <!-- This is where your HTML code starts -->
<h4>Program output</h4>
<div id="output">;
</div>
</body>
</html>
```

The stylesheet used contains the following style rules. You should create this and place in the same folder as your coding examples.

```
h2      {margin-top:-8px;height:50px;text-align:center;
         font-style:italic; font-weight:bold;
         background:#aaccff;
                    }
form    {background:#eee;padding:2px;
         border:1px solid black;
```

```
}
```

> Javascript is case sensitive, this cannot be emphasised enough! So var my_Variable **is not the same as** my_variable. Check your spelling as well.

You may also want to include the simple form with test button in your template as a default. In many of the examples the function name is test. you may wish to change this to something more meaningful, in which case, make sure the onclick event of the test button is also changed to call the right function.

> The alert box is often used as debugging aid to show messages as the program executes. If you have Firebug installed you can use console.log() in place of alert(), however this **only works in the Firefox browser**. To use it, you must have the Firebug displayed in the window as shown below.
> The use of console.log() therefore is not recommended, a better solution if you don't like alert() boxes, is to set up a ... box on the page to collect *progress* messages.

Place this HTML code just above the ***program output*** heading for example.

```
<form>
  <input type="button" value = "test" onclick="test()" />
</form>
```

Javascript Extended (project) exercises

If you want to learn a scripting language, and you do, if you want to turn static HTML pages into D(ynamic)HTML. The best way to learn Javascript is to start from what you know, I.e. a knowledge of HTML, then select real DHTML problem that's not to simple, nor to complex to help identify the things you need to know. There is no single path to learning any programming language, sure there are some basic concepts you need to get out of the way first, but after that you learn the bits that are relevant to the current problem. All you need is a browser and a text editor, such as gedit, or better still, the Bluefish editor since it has language support for HTML,CSS and PHP.

The javascript examples focus on the feature being demonstrated and so are deliberately kept as short and simple as possible. Once you are past the basics structures of sequence, selection and repetition, then the focus of the Javascript projects is primarily on problem specification and the development of potential solutions. This includes the Human Computer Interface (or Interaction), HCI for short. Any program can be greatly influenced by how the information is presented to user. In this case it that means the HTML and CSS design.

You will find the projects more challenging than the simple exercises, because they are designed to help develop the ancillary skills you will need as a programmer, not just the *mechanics* of any particular programming language. The projects have been chosen to exercise different aspects of the javascript language, so depending on which project you choose to follow, you will be directed to the relevant background material. The order in which you learn will therefore inevitably be different to the order presented in the main menu. That way you will learn just those parts of the language that maybe immediately relevant to you.

As far as possible the projects are listed in order of difficulty, starting with the easiest first. Each project will identify the prerequisite knowledge you will need to tackle it. As well as the *mechanics* of javascript, the projects will involve some related topics such as, data structures, the Document Object Model (DOM).

Javascript libraries

There are various Javascript libraries on the web, some better than others. One of the more popular is JQuery. The main rationale behind these libraries is,

- Don't reinvent the wheel
- Productivity reasons
- Encapsulation

They are supplementing the javascript language by providing common functions that have already been tried and tested. This is all well and good, however while learning to program, my advice is treat them with a degree of caution; as you may end up more confused. Unless you are an advanced user, its likely the libraries are a coding style and programming techniques which are alien to you, better to initially build up your own library, which can be supplemented later on with external libraries.

> If you are familiar with Python, or PHP, these languages are build in a more modular fashion, so the language can be customised to your particular needs. As a general observation, having modules under a

> central coding authority, should mean a higher degree of standardisation and coding proficiency, than you might otherwise find from *independent*libraries. This is perhaps particularly true when it comes to documentation.

Encapsulation (or Information hiding) is part of the Object Orientated Development (OOD) methodology. As programs get bigger you need a way of reducing their complexity to a level that is more manageable. The idea is that once you have build a function or object, you can forget about how it works and just focus on its interface. You can then use this function to build higher level functions, so limiting the number of building blocks (functions) you have to deal with. A good example of this philosophy is the Javascript Math object. When you plug a radian value into a sin() cos() function, you really don't need to know how the return value was calculated, only that its right.

> Flow charts can, to some extend, be adapted for large programs, the technique quickly breaks down for all but quite modest programs.

Source : http://www.soslug.org/node/1716