A reference guide to HTML, CSS and JavaScript

A blank webpage template that passes validation (remember to indent when copying and pasting to a text editor)

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title></title>
  <style type="text/css">
  </style>
</head>
<body>
<p></p>
</body>
</html>
```

What goes where?

Everything visible in the webpage (i.e. that will show up in a browser) will go between the `<BODY>` and `</BODY>` tags. CSS will either go in between the STYLE tags, or it will go "in-line" in the body (see CSS section below).

Start a tag, close a tag

Let's say we're adding a paragraph inside the BODY section of our webpage. To open a paragraph tag, just type in the tag like:

```html
<P>
```

After you type in some text, you'll need to close the tag. To close a tag, add a slash just after the opening "<" symbol like:

```html
</P>
```

With a few exceptions like the DOCTYPE, META, and IMG tags, EVERY tag you open you have to close.

Example:
A sample paragraph with paragraph tags in our BODY section might look like:

```html
<P> This is a paragraph. </P>
```

The tags we've learned and how to use them

If any of these tags below are not defined clearly, check out the reference at:  
[http://www.w3schools.com/tags/default.asp](http://www.w3schools.com/tags/default.asp)

- `<H1>`, `<H2>`, ... , `<H6>` Each of these are heading tags with `<H1>` being the largest heading and `<H6>` being the smallest heading.
- `<P>` Paragraph tags enclose simple text paragraphs in our BODY.
- `<TITLE>` This tag appears in the HEAD section of our webpage. This applies a title to appear in the title bar (the blue bar at the top of the browser).
- `<DIV>` Divides up the webpage into sections. Handy to use to apply centering style to multiple paragraphs in the BODY.
- `<UL>` Unordered list. Each list item will appear with a dot next to it.
- `<OL>` Ordered list. Each list item will appear with a number next to it.
- `<LI>` List item. Only use these tags between `<OL>` </OL> tags or `<UL>` </UL> tags. These designate items in your list.
- `<TABLE>` Starts a table.
- `<TR>` Table row. Each time you want to add a new row to your table, use this tag. Only use these tags between `<TABLE>` </TABLE> tags.
<TD> Table data. These designate cells in your table. Only use these tags within a row, i.e. between <TR> </TR> tags.

<TH> Table headers. These designate cells in your table, but apply a **bold** formatting to your data. These are often used in the first row of your table.

<IMG> tag:
looks like: `<IMG src= "X" alt= "Y">` where X should be replaced with the file name of your image, and Y should be replaced with the alternative text (or caption).
Example:
`<IMG src= "mypic.jpg" alt= "A picture of me!">`

<A> tag:
Makes a hyperlink in your webpage.
Looks like: `<A href= "X">` where X should be replaced with either the html file name to link to, or the full URL of a webpage to link to.
Example1:
`<A href= "http://www.google.com"> A link to google </a>`
In the browser, the text in between the A-tags will appear blue and underlined, available to click on.

Example2:
`<A href= "subpage.html"> A link to my subpage </a>`

<INPUT> tag: creates a button, select box, text box, or other input area. Use `<INPUT type="Button" onclick="…">` to create a button with a click handler.

CSS

There are three slightly different ways of applying style to your webpage: 1) **inside the <STYLE> tags in the HEAD of our webpage**, 2) "in-line" in our BODY, 3) named styles.

Way #1: **inside the <STYLE> tags in the HEAD of our webpage**

Where it goes:
Style definitions to be applied to a tag **every time** it appears in the BODY of the webpage should go in between the <STYLE> and </STYLE> tags in the HEAD section of our webpage.

The template:
TAG { property:value }

The "TAG" is actually one of the tags (H1, P, IMG, etc) we want to apply style formatting to, the "property" is the CSS property to add or change (text-align, color, background-color, etc), and the "value" is what to change the property to. Notice that we do not have the "<" or ">" symbols wrapped around our TAG.

Example:
Apply green font color to all **paragraph** tags.

How to do it:
Notice the area in the blank webpage at the beginning of this reference guide:
`<STYLE type= "text/css">`

`</STYLE>`

In the blank line between <STYLE> and </STYLE> we follow our template above
P { color:green }
alternatively:
P { color:#00FF00 }

Thus, our STYLE area of our blank webpage would then look like:
`<STYLE type= "text/css">`
If we wanted to change all the H1 tags to also be green font color, our STYLE area would change to something like:

```html
<STYLE type= "text/css">
P { color:green }
H1 {color:green }
</STYLE>
```

Notice we didn't need any more <STYLE> tags in the HEAD section.

**MULTIPLE Style Attributes for the same element**
Separate multiple style properties for the same element with semicolons.

Template:
TAG{ property:value ; property:value ; ... }

Example:
If we wanted the paragraph tags to be both green font color AND have a blue background color, then our STYLE area would change to something like:

```html
<STYLE type= "text/css">
P { color:green ; background-color:blue }
</STYLE>
```

**Way #2: "In-line"**

"In-line" is a way of adding a style definition that is to be applied only to one tag in our body.

The template:

```html
<TAG STYLE= "property:value">
```

The "TAG" is actually one of the tags (H1, P, IMG, etc) we want to apply style formatting to, the "property" is the CSS property to add or change (text-align, color, background-color, etc), and the "value" is what to change the property to.

Example:

**Center the first paragraph.**

How to do it:
This example would require a paragraph in our BODY section of our webpage. We find the paragraph we want to center and apply the template above.

Thus, before our "in-line" CSS is added, the BODY section of our webpage might look something like:

```html
<BODY>
P>This is a paragraph.</P>
</BODY>
```

Afterwards, the paragraph changes to:

```html
<BODY>
P STYLE= "text-align:center">This is a paragraph.</P>
</BODY>
```

**MULTIPLE In-Line Style Attributes for the same element**
Separate multiple in-line style properties for the same element with semicolons.

Template:
Example:
If we wanted the paragraph tags to be both centered AND have a blue background color, then our above example would change to something like:

```html
<BODY

<P STYLE= "text-align:center ; background-color:blue">This is a paragraph.</P>
</BODY>
```

Way #3: Named styles.

Named styles are a mix of the other two CSS methods above.

Step 1) Define and name your style.

In the HEAD section of our webpage, in between the <STYLE> and </STYLE> tags, a named style looks something like this:

```html
.name { property:value }
```

Where "name" is any name you'd like, as long as it is preceded by a period, "property" is the CSS property to add or change (text-align, color, background-color, etc), and "value" is what to change the property to.

Step 2) Apply your named style to a tag (or multiple tags) in the BODY section of your webpage

```html
template:
<TAG class= "name">

Example:
Let's say we want to center the first and last paragraphs, but not all paragraphs, in our body.

We first define our named style in the STYLE area of the webpage (and let's assume we name the style "example1")

Our STYLE area would change to something like:

```html
<style type= "text/css">
.example1 { text-align:center }
</style>
```

The BODY section of our webpage would change to:

```html
<BODY

<P class= "example1">This is paragraph one.</P>
<P>This is paragraph two.</P>
<P class= "example1">This is paragraph three.</P>

</BODY>
```

MULTIPLE Named Style Attributes
Separate multiple style properties for the same named style definition with semicolons.

Template:

```html
.name { property:value ; property:value ; ...}
```

Example:
Using the previous example, if we wanted a particular paragraph to be aligned center and font colored green, the above example's named style in our STYLE area would change to something like:

```html
<style type= "text/css">
.example1 { text-align:center ; color:green}
</style>
```
If you don't know the property name or value for CSS, you'd need to look it up in the CSS section of blooberry.com

JavaScript

The formula for making JavaScript happen:
1) Make a SCRIPT section in your webpage
2) Define and implement a function in the SCRIPT section
3) Apply an event handler in the BODY section of your webpage that runs your function when the event happens

The SCRIPT section of your webpage
The SCRIPT section for JavaScript goes in your HEAD section of your webpage. Inside, you could define JavaScript functions.

```html
<SCRIPT type='text/javascript'>
/* This is a comment. */
function function_name( parameters ) {
  What to do.
}
</SCRIPT>
```

'This is a comment.' is a placeholder for a specification to the reader or code writer of what the function does. A function must start out being defined with the word 'function', followed by the function's name. The function_name can be anything you want to name your function. 'parameters' above is a placeholder for a list of things you want to pass the function (see Function Parameters section below). 'What to do.' is a placeholder for the code you want your function to run (see the What to do in a function section below).

A function runs when it is called. We can call functions using Event Handlers.

Event Handlers
An 'event' can be a mouse click, rolling the mouse over a section of text, etc. When an event happens, you can run a function.

A List of the event handlers we've learned:
- onclick
- onmouseover
- onmouseout

Each of these can be an attribute of any tag, e.g. `<img onclick="handle_click()"... >`, where handle_click is a function.

What to do in a function:

Object references to page elements
If we want to affect elements in our webpage, we need a handle to them. A “handle” is called an object reference to the page element.

To create an object reference to a specific page element, use the id attribute. You should not use the same id in two different tags.

Template:
```
<TAG id='Steve'>
Then:
```
- `my_object = document.getElementById('Steve')` creates an object reference to the page element whose id is ‘Steve’
- `my_object.className='newStyle'` changes the style of the page element to the named CSS style .newStyle.

**Function parameters**

We can make functions more flexible by providing parameters. These are input values that give the function some details about how to do its job. For example, instead of hard coding our function above to only 'highlight' the 'Steve' element, we can pass in the ID and the Style name as a parameter. The example above would then look like:

```javascript
/*
 * Inputs:
 *    theElement: id of an element to be changed
 * Result:
 *    className of theElement is changed to “newStyle”
 */
function change_style(theElement) {
    obj = document.getElementById(theElement);
    obj.className="newStyle";
}
```

- `<input type='button' value='Change to highlight' onclick="change_style('Steve','highlight')">`

**Predefined functions:**

- `window.alert(message)` where message is a quoted string
- `window.Prompt(message)` – same as alert, but returns a value entered by the user
- `document.getElementById(id)` – see object references to page elements, above
- `window.open("content", "window-name", "window-features")`
  - “content” is the location of the Webpage that will open in the new window. This can be a Web URL starting with http:// or the name of a local file.
  - “window_name” should be either "_blank" if you want to open a new window or "_self" if you want to replace the contents of the existing window.
  - “window-features” is a string giving the size and location of the window. For example "left=100,top=200, height=350,width=500" creates a window that is offset 100 pixels from the left of the screen and 200 pixels from the top of the screen, and is 500 pixels wide by 350 pixels high.