HTML Coding Guidelines and Policy for Staff

Effective: August 2023

Version: 1.1

Note, this document is available in alternative formats upon request including electronic or audio format.

Table of Contents

Policy Statement

This Policy provides a set of guidelines to ensure consistency of coding and coding practices. Consistency is absolutely a prerequisite for maximising maintainability and reusability. All users of Organisation IT services are required to comply with the principles outlined in this policy.

Scope

Consistency is absolutely a prerequisite for maximising maintainability and reusability. These general guidelines for coding style can form the basis of a set of standards that will help ensure that all developers in a project - or, better, in all projects across an organisation - write code consistently.

- Use well-formed HTML.
- Pick good names and ID values.
- Indent consistently.
- Limit line length.
- Standardize character case.
- Use comments judiciously.

Use Well-formed HTML

Although Web browsers are generally forgiving and can ignore many mistakes, rendering most HTML as the document author intended, it is still a good idea to use well-formed HTML code, for a number of reasons.

Well-formed markup code is a concept that has gained importance with increased implementation of XML. While browsers did not, in general, enforce HTML language rules very closely, XML parsers do. Code is considered well formed when it is structured according to the rules for XML 1.0. These rules relate to character case, tags, nesting, and attribute values.

In general, when most browsers encounter an unrecognised or extraneous tag, they ignore them. However, different browsers might deliver results in different—and unpredictable—ways. In addition, future versions of browsers might adhere to standards more closely than do current versions. Finally, code that includes such elements can be harder to read and understand, making maintenance more difficult.

Standards

1. Doctype and layout

All documents must be using the HTML5 doctype and the <html> element should have a "lang" attribute. The <head> should also at a minimum include "viewport" and "charset" meta tags.

```
<!DOCTYPE html>
<html lang=”en”>
      <head>
      <meta charset=”utf-8” />
      <meta name=”viewport” content=”width=device-width, initial-scale=1”>
      <title>Example Site</title>
      </head>
<body>
……
</body>
</html>
```

2. Lowercase names

To be well-formed, element and attribute names must be in all **lower case**. In versions through 4.01, HTML is not case-sensitive. However, XML is case-sensitive, and it follows that the HTML5 recommendation is also case-sensitive. So, to ensure that code keeps working and to maximize reusability, this must be planned for.

3. Closing tags

All nonempty elements must have corresponding closing tags. Empty elements, or self-closing elements - those previously signified with a single tag, such as `<hr>` and `<br/>` must end with ”/”. For example, `<hr/>` and `<br/>` are both examples of well-formed code.

4. Nested elements

All nested attributes must be properly nested—for example:

```
<center><b>Some text</b></center>
```

Note that the `<b>` tag and its corresponding closing tag, `</b>`, are both nested inside the `<center>` and `</center>` tags.

If elements overlap, then they are not properly nested, as illustrated in the following code:

```
<center><b>Some text</center></b>
```

While many browsers have accepted overlapping elements and given the expected results, they have always been, strictly speaking, illegal in HTML, and future versions of browsers might not support them

5. Attribute values

Attribute values, even numeric attributes should be quoted—for example:

```
<input name="txtName" type="text" size="1">
```

6. Code validation

Another step toward improving HTML code is to validate it against a formal published grammar and to declare this validation at the beginning of the HTML document. For example, the following line declares validation against the public HTML**5** Final grammar:

```
<!doctype html>
```

A list of formal published grammars is available from the W3C at http://validator.w3.org/sgml-lib/catalog . The W3C also has a public HTML validation service at http://validator.w3.org/

7. Pick Good Names and ID Values

```
<b>Member? </b>
<input type="Checkbox" name="cbIsMember"><br/>
<b>Admin? </b>
<input type="Checkbox" name="cbIsAdministrator"><br/>
<b>Owner? </b>
<input type="Checkbox" name="cbIsOwner"><br/>
```

8. Indent Consistently

Use indentation consistently to enhance the readability of the code. When elements carry over more than one line of code, indent the contents of elements between the

start tag and the end tag. This will make it easy to see where the element begins and ends. Also, use indentation to align code at attribute names (see Listing 3).

It is a good idea to use **one TAB for each level in indentation**. If possible, set up the development tool to convert tabs to spaces so that the indentation will be the same when the source is viewed in different editors or as printed output.

```
<video autoplay="autoplay" poster="poster_image.jpg">
    <source src="foo.ogg" type="video/ogg">
</video>
```

9. Limit line length

Break up lines when they run too long. It is much easier to read and understand code when you can see the entire line at once. When lines of code are so long that the reader must scroll right and left to read them, it requires much more cognitive effort to understand what the code is doing. Alternatively, in some applications, long lines might wrap to the next line at the nearest word break. In either case, source code is much easier to read and understand if the developer takes explicit control of line length.

HTML is not sensitive to line breaks, so the developer can break lines at will between keywords for readability. For example, the code snippet below in which two elements have word-wrapped to the next line because they were two long for the editor window.

```
<td valign="Center">
    <input type="Text" Length="45" name="txtName" language=
        "JavaScript" onclick="return NameValid();"><br/>
    <input type="Text" Length="35" name="txtAddress"
        language="JavaScript" onclick="return AddrValid();">
</td>
```

10. Use Comments Judiciously

Good comments can be invaluable for understanding and maintaining code. However, the unique nature of HTML introduces a trade-off between the value of thorough comments and the efficiency of the Web application.

The Web server reads in the HTML code and sends it as a stream of text over the network to the browser. Only after arriving at the client does the browser parse and interpret the HTML code, displaying the visible elements and ignoring the comments. The obvious implication is that the comments add nothing to the document as the browser displays it, yet they add to the processing overhead on both the server and

client computers, and they increase the amount of data transferred. Below is an example of good commenting

```
<!-- Form for input of security groups -->
<form name="Form1" action="https://www.mydomain.com/input.php">
      <b>Member? </b><input type="Checkbox" name="CB1"><br/>
      <b>Admin? </b><input type="Checkbox" name="CB2"><br/>
      <b>Owner? </b><input type="Checkbox" name="CB3"><br/>
      <input type="Submit" Value="Submit">  
      <input type="Reset" Value="Clear">
</form>
```

Definitions and Acronyms

Term	Definition
HTML5	HTML5 (**Hypertext Markup Language 5**) is a markup language used for structuring and presenting hypertext documents on the World Wide Web. HTML stands for Hyper Text Markup Language. HTML is the standard markup language for creating Web pages. HTML5 is the current standard at time of writing this document.
Element	HTML describes the structure of a Web page. HTML consists of a series of elements. HTML elements tell the browser how to display the content. An element is a set of opening and closing tags. An example of this would be the set of `<article>` and `</article>`.
Tag	a tag is a piece of HTML code. An example of this would be `<article>`
XML	**Extensible Markup Language** is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. HTML5 is a subset of XML.

Related polices and other relevant documents

Staff Code of Conduct

Related legislation

Copyright in computer code Coding rights are covered by the *Copyright Act 1968 (Cth) (Act)* and include the right to reproduce, publish and adapt the work. Computer programs or compilations of computer programs are also protected by the Act.

Who owns the code? Who owns the computer code is an important issue not just for employers but also IT contractors, who may utilise their own code libraries, and business associates embarking on joint software development projects

Section 10 of the Copyright Act 1968 (Cth) (Act) defines the term 'literary work' to include tables, compilations, figures and symbols expressed in words, as well as a computer program or a compilation of computer programs.

Section 47 of the Act further defines “computer program” to include any literary work that is:

- incorporated in, or associated with, a computer program; and
- essential to the effective operation of a function of that computer program.

Review Date

June 2024

Contact Information

Chief Information Officer

Revision History

Version No.	Approved/ Amended/ Rescinded	Date	Approval Authority	Amendments
1.0	Approved	August 2023	ITC MD	
1.1	Amended	January 2024		Corrected grammar errors

https://docs.ckan.org/en/2.9/contributing/html.html