

An HTML5 Learning Tool


Context:

DECO1400/7140 is an introductory course that teaches web design & development. The specific web technologies covered are HTML/CSS/JavaScript. This course is core for the Bachelor of Multimedia Design and Bachelor of Information Technology and the Masters Interaction Design. In its current form, the course teaches HTML4.01 and given the stability of HTML5 the teaching staff would like to update content to cover the newer specification of HTML.

Problem:

The issue with teaching HTML5 is that the specifications do not restrict or validate code for structure or semantics or syntax. As HTML5 is designed to be flexible in syntax, and to be backwards compatible, existing validators do not enforce correct tag closure, correct page structure, or deprecated tags from previous versions. These common mistakes do generate errors when validated against earlier versions of the HTML specification. This means that, in a learning environment, students are not made aware of errors in their code, leading to poor practice and understanding of the web environment.


Validation Output: 10 Errors

 **Line 6, Column 8: required attribute "TYPE" not specified**

`<script></script>`

The attribute given above is required for an element that you've used, but you have omitted it. For instance, in most HTML and XHTML document types the "type" attribute is required on the "script" element and the "alt" attribute is required for the "img" element.


Typical values for type are type="text/css" for <style> and type="text/javascript" for <script>.

 **Line 9, Column 7: element "EMBED" undefined**

`<embed>`

You have used the element named above in your document, but the document type you are using does not define an element of that name. This error is often caused by:

- incorrect use of the "Strict" document type with a document that uses frames (e.g. you must use the "Frameset" document type to get the "<frameset>" element),
- by using vendor proprietary extensions such as "<spacer>" or "<marquee>" (this is usually fixed by using CSS to achieve the desired effect instead).
- by using upper-case tags in XHTML (in XHTML attributes and elements must be all lower-case).

 **Line 17, Column 6: required attribute "ACTION" not specified**

`<FORM>`

The attribute given above is required for an element that you've used, but you have omitted it. For instance, in most HTML and XHTML document types the "type" attribute is required on the "script" element and the "alt" attribute is required for the "img" element.

Typical values for type are type="text/css" for <style> and type="text/javascript" for <script>.

Example of error output from validator.w3.org

This isn't to say that the validators for earlier version were entirely suited to the beginner web developer. Errors & warnings in the existing W3C validator can be difficult to interpret, understand and fix. The errors are listed according to the line & column placement in the source code, requiring the developer to find these themselves. Error messages are delivered in technical language which can be difficult to understand for new developers.

What do we want:

A web-based learning tool that allows students to "validate" their code in order to see where they are making structural, semantic, or syntactical errors. This tool should allow for them to upload HTML files, and have these files parsed for errors. This tool should then

provide clear feedback to the students about the errors discovered and suggestions for fixing these and if possible provide links to information on the tag.

The most common errors seen by those new to HTML are:

- Structural/syntactical:
 - Multiple instances of singular tags - html, head, body, footer
 - Incorrect page structure (html, head, body, footer - where tags are missing or in the wrong order)
 - Form elements not being contained in a form object
 - Failure to close tags that require a closing tag
 - Incorrect nesting of tags - resulting in overlapping html tags
 - Incorrect table structures - cells not in rows, different numbers of cells in rows where colspans are not specified
 - Missing title tag in head
 - Missing required attributes (src for img, href or name for a, href for link etc)
 - Use of short tags - self-closing tags not having /
 - Use of PHP in a html file (not using php extension)
 - Form elements
 - incorrect type attributes for inputs (or misspelling of)
 - missing value attributes,
 - radio inputs with the same id,
 - inputs missing name attribute - causes issues when accessing via JavaScript or PHP
- Deprecated elements:
 - Use of frames
 - Use of deprecated, presentational tags (b, i, small etc)
- Accessibility:
 - Form elements not having labels
 - Missing alt tags on images - accessibility standards not followed
- Poor practice/Miscellaneous:
 - Using tables for layout
 - Semantic issues - multiple H1's, incorrect use of headings
 - Multiple elements with the same value for the id attribute - causes issues when they begin to work with JavaScript and the DOM.
 - Special characters used or non-ASCII character used.

In addition, it would be good to provide a mechanism whereby the students file structure is checked by the system. This would allow students to upload a fully linked site that is then checked for:

- Incorrect linking to local files - images, css, javascript and other HTML files. This could be due to files being in a different location to the link specified or due a mismatch in the case used in the filepath.
- Presence of an index.html file. This is something that students regularly forget which causes issues when they publish to a web server.
- Cleanliness of file structure - placement of CSS files into a CSS directory, of image files into an images directory etc.

Technology for Tool

A web-based application would be ideal so that it is not location-dependent or platform-dependent. The specific web technology is up to you - PHP/MySQL is first thought but it could be anything that supports the tool & its users.

Other works

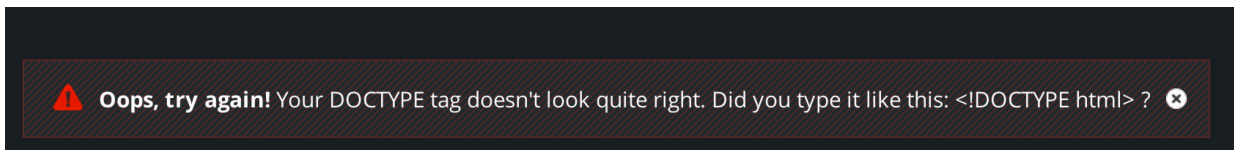
It is important to review existing HTML4.01 & HTML5 validators such as <http://validator.w3.org> or <http://html5.validator.nu/> for presentation of errors & functionality. In terms of learning tools, the CodeAcademy Web Fundamentals course at <http://www.codecademy.com> would be worth reviewing.

Example of error representations from CodeAcademy.

Key Considerations:

Representation of errors back to the student

The way that errors are represented to the student needs to be clear, obvious and should use beginner level language. Students should be able to ascertain what kind of error it is and its level



of severity. Errors should be displayed on top of the source code, rather than as a list denoted by line/column numbers and with mouse overs to explain & suggested fix. This should be reinforced by the list of errors underneath or as a separate tab for reinforcement.

Errors should be coded that student understand what kind of error it is - semantic, structural, syntactical or best practice. This could be colour-coded or denoted by icon. The language used in the interface & error reporting should be plain english & should be easy to understand. Syntax highlighting would be useful when viewing code for errors/success.

Multiple files & Site-wide Checking

The system should allow for multiple files to be uploaded at the one time. This allows students to check multiple files at once but also to allow students to visualise file structure.

Testing expectations

As this tool is intended for learning, user testing will be key for evaluating & informing the representation of errors, use of the system and understanding of errors and system. User testing should be carried out frequently during the build process, and with appropriate representatives of the target audience - people with little/no experience with HTML. It is assumed that the team will have a strategy for functional testing.

Interface Sections

Landing Page

This should explain the tool and provide simple instructions for use with a link through to uploading source code.

Help Section

There should be a help section that describes how to use the tool, what the encoding of the errors means and the various methods for checking files. It could also provide links to well-documented HTML5 learning resources such as <http://htmldog.com/> or <http://www.codecademy.com/>.

Upload Source Code Page

Students should be able to submit source code for checking using the following approaches:

- Direct Input - copy and paste HTML code directly into the browser for processing.
- Upload Single File - upload a single HTML file
- Upload Multiple HTML files - ability to select multiple files to be processed.

- Upload Site - ability to upload a zip file containing all files for a site to be processed. This will include images, JavaScript files, CSS files and HTML files. It is not necessary to store the non-HTML files, only filenames & path will be required for checking the site file structure and linking.

File List Page

This page is displayed when multiple HTML only files are uploaded. This should provide an overview of the files uploaded, how many errors have been detected and of what kind (see sketch titled Uploaded Files). This page can also be used as a progress indicator as each file is processed. A good example of this is the upload progress screen that Flickr displays.

Clicking on a filename will take the student to the Error Page for that file.

File Structure View

This page is displayed when a site is uploaded. This will represent the file structure of the zip file to the student. Mousing over or clicking on an HTML file icon will show the working links to other files and where links are broken will list those and make an attempt to suggest which file might be required. For example, if a HTML file has the path “images/fred.jpg” as the broken link and the images directory contains a Fred.jpg file, the system could suggest this as the intended file. This would highlight the case-sensitive restrictions on file linking. (see sketch titled File Structure View).

This page would also highlight files that fall out of standard practice for file structure. For example, if a JavaScript file is not contained within a js directory.

Students would also be able to click through to the File List Page which would outline the errors on the HTML files.

Clicking on a HTML file icon would take students to the Error Page for that file.

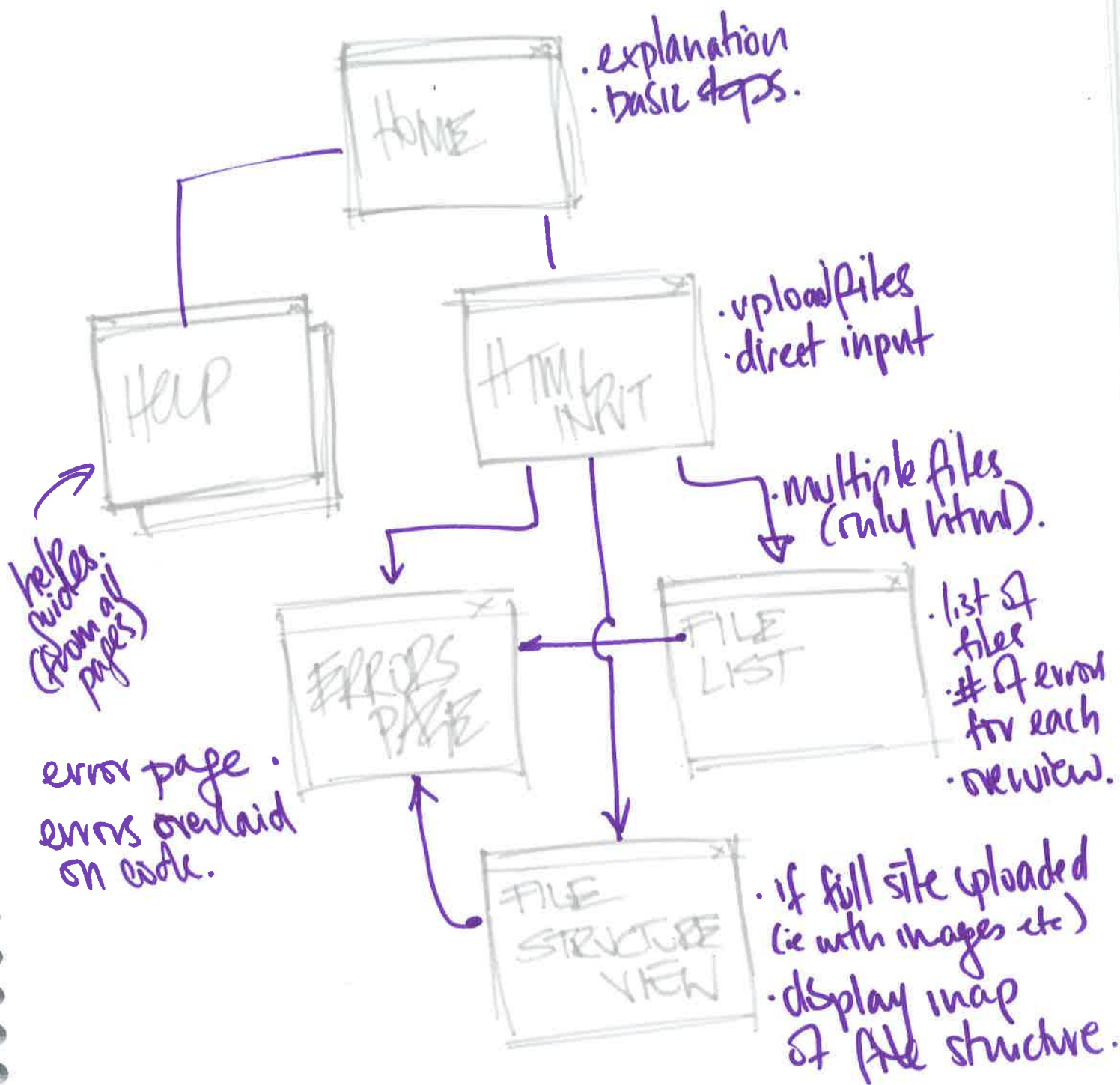
Error Page

This page would represent the errors discovered when processing a single HTML file. The source code would be displayed to the students with errors highlighted in the colour of their category (ie. Syntax errors in red, Accessibility errors in blue etc). Students can mouse over the error to view an explanation of that error, with a suggested fix or link to further information.

A full list of errors, listed according to line number, should also be displayed under the source code view. This list should provide explanations of the errors.

Interface Mockups

On the following pages are sketches for the overall interface mapping & major interface screens.



FILE LIST PAGE

Rendered when multiple files uploaded. Only in HTML format.

click to view full errors for page.

Quick visual of number & type of errors discovered.

UPLOADED FILES

X FILES UPLOADED. 4 ERRORS FOUND

FILENAME	ERRORS
~~~~~	<del>5</del> <del>5</del> <del>2</del>
~~~~~	<del>10</del> <del>2</del> <del>5</del>
~~~~~	<del>1</del> <del>7</del>
~~~~~	<del>1</del> <del>8</del>
~~~~~	<del>0</del>



## FILE STRUCTURE VIEW

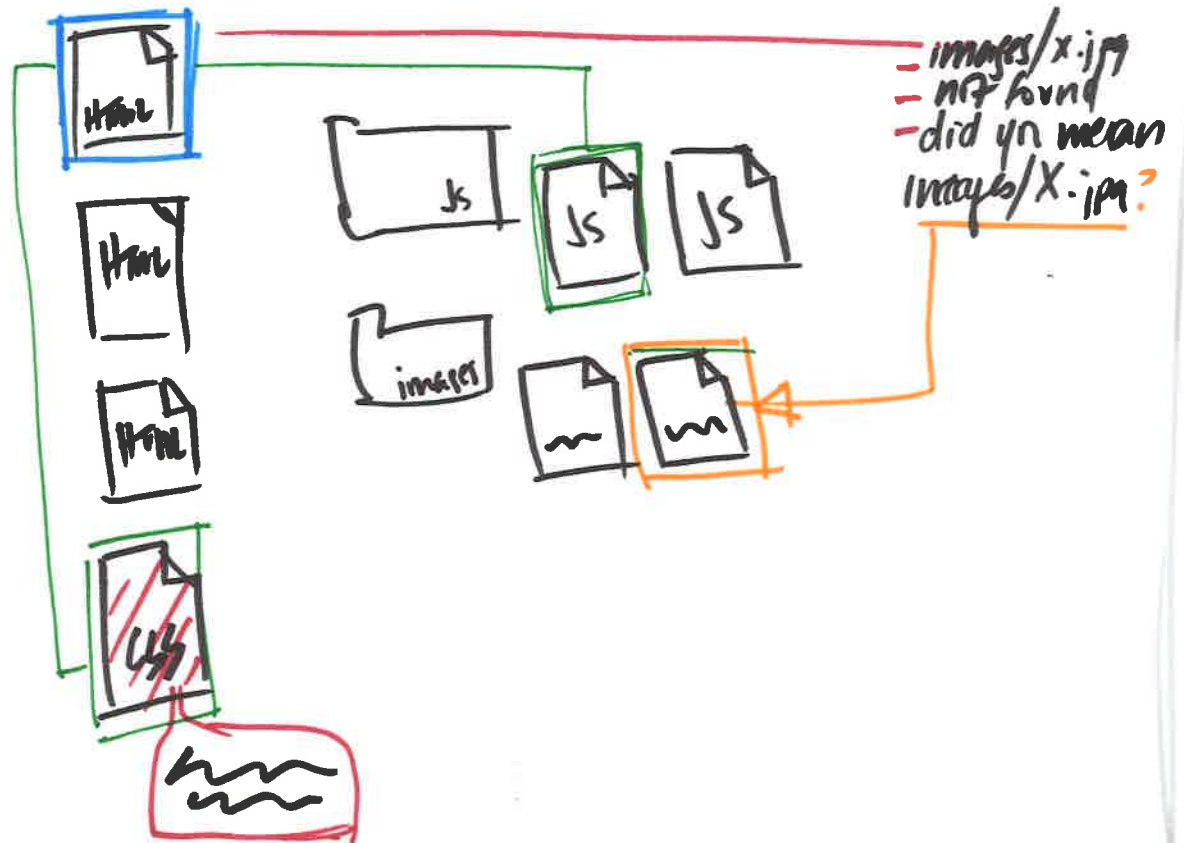
Rendered when linked  
HTML/CSS/JS uploaded  
as site structure.

Click on the HTML  
file to highlight links  
from that.

Functioning links highlighted.

Broken links listed with  
suggestions for what it  
might be (case-sensitive).

## FILE STRUCTURE VIEW



## ERROR PAGE

- source code with errors highlighted
- overview of error + types at top
- click on to highlight only that type
- list of errors under.

## ERRORS LISTING/PAGE

FILENAME: ~~~~~

# ERRORS 5 7 2

click to show only those errors

~~<html>~~

<head>

<title>

</head>

<body>

<h1>

<p>

<div>

<p>

</div>

<form>

<input type="text">

</form>

</body>

</html>

mouse over.

Plus closed before internal tag.

label messy.