Understanding the Functionality of HTML

The HTML5 standard is primarily authored and maintained as a "living standard" by the *Web Hypertext Application Technology Working Group* (WHATWG). The *World Wide Web Consortium* (W3C) publishes stable snapshots of this living standard as formal W3C Recommendations.

Web browsers are very forgiving and will make a best effort to display an HTML file, even if there are errors. However, writing correct, standards-conforming HTML is crucial for ensuring predictable rendering across different browsers and devices, as well as for accessibility, search engine optimization, and long-term maintainability.

Note: inline styling should not be used when designing modern web pages; however it can be an excellent way to save time when writing the Pearson exam – you can quickly add a style to a specific tag without needing to create a separate `<style>` section or CSS file.

1. Vital Concepts

1.1 Separation of Concerns

Separation of Concerns (SoC) is a design principle that means you should organize a system into distinct sections, each handling a single, specific job. This makes things easier to build, debug, and improve.

For web page design, the functionality of a web page is separated into the following concerns:

- **HTML: structure and content**
  Defines the skeleton and semantics of the page; the hierarchy of elements like headings, paragraphs, lists, and form controls.
- **CSS: presentation and style**
  Controls the visual appearance and layout; everything from colors, fonts, and spacing to responsive grid systems and animations.
- **JavaScript: behavior and logic**
  Manages the interactivity and dynamic functionality; responding to user actions, fetching data, and manipulating the content or style in real time.

1.2 Inline, Internal, and External

HTML forms the core structure and only essential element of a web page. It is loaded first by the browser. CSS for styling and JavaScript for behavior can then be added via three methods: *inline*, *internal*, or *external*.

External is the only recommended and standard practice for production code due to its maintainability, reusability, and performance. Other methods are acceptable for quick prototypes, testing, or when a single, isolated rule or script is absolutely necessary. Although you should avoid the other methods in general, **on a Pearson exam, adding *internal* or *inline* CSS or JavaScript may save a lot of time, so it is useful for you to learn these**.

Examples of how to add CSS and JavaScript to your web pages will be covered in Topics 8 and 9, respectively.

2. Basic HTML Document Structure

Below is an example of a short HTML Document. In this section we will briefly discuss the elements shown in **bold font**: the doctype declaration, and the html elements `<html>`, `<head>` and `<body>`.

The doctype declaration must be the first line of the file, and there must be only one of each of the above listed elements per HTML document, the `<html>` element must enclose both the `<head>` and `<body>` elements, and the `<head>` element must precede the `<body>` element.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6            content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <h1>Web Page Heading</h1>
11     <p>This is the first paragraph.</p>
12     <p>This is the second paragraph.</p>
13     <p>
14         <a href="https://www.nielsenedu.com/index.html">
15             A link to another web page
16         </a>
17     </p>
18 </body>
19 </html>
```

Below is a basic overview of these high-level parts of the template HTML document shown above.

`<!DOCTYPE html>`

This declaration should be the first line of every modern HTML file. This one simple declaration tells the browser to render the page in modern standards mode. It is not case-sensitive, so capitalization does not matter.

While you might occasionally encounter older, more complex doctypes, like the example below, there is no longer any reason to use anything other than the current, simplified version for any new webpage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
                      "http://www.w3.org/TR/html4/strict.dtd">
```

The doctype declaration is not an HTML tag, but rather a necessary instruction to the browser.

`<html lang="en"> … </html>`

The `<html>` element is the root element that wraps all other HTML content on a page. There must be only one `<html>` element in each HTML document.

The only part of your HTML file that exists outside the `<html>` element is the very first line: the `<!DOCTYPE html>` declaration.

Including the `lang` attribute use on the `<html>` tag is strongly advised as it is considered a fundamental best practice for accessibility and technical correctness.

- Screen readers use it to select the correct voice and pronunciation.
- Search engines can better index and serve the page to the correct language audience
- Translation tools can use it to determine the language to translate
- Browsers can apply language-specific typography rules, such as correct quotation marks or hyphenation

The `<html>` element should only have two direct children: `<head>` and `<body>` elements. Any other element placed inside the `<html>` element will be automatically moved by the browser into the `<body>`.

`<head> … </head>`

The `<head>` element holds essential information about the page that is not the visible content itself, but is still crucial for the browser, search engines, and other services.

Attributes are rarely applied to the `<head>` element because the contents of this element are not displayed.

The contents of the `<head>` element will be expanded on in section 3, below.

`<body> … </body>`

The `<body>` element represents the visible content of an HTML document. Everything users see and interact with in the browser window, such as text, images, links, forms, and multimedia must be inside the `<body>` element.

Attributes are rarely applied to the `<body>` tag; perhaps an `id` or `class` attribute for styling.

Traditionally, some event handlers were included in the `<body>` tag, such as `onclick` or `onload`. This is discouraged in modern HTML because, for separation of concerns, functionality should be implemented in the JavaScript, not the HTML. Unlike adding inline CSS, adding inline JavaScript is not really any quicker, so I would recommend sticking with best practices. However, you should be able to read and understand those attributes in case code given on the exam includes it.

3. Inside the <head> Tag

`<title> … </title>`

A valid HTML file is required to have a `<title>` element within the `<head>` element (if not present, browsers will usually generate one for you). The contents of the `<title>` element is shown in the browser tab and search engine results.

`<meta charset="utf-8">`

Declares the character encoding for the document (e.g., UTF-8). This element should be the first element after `<title>`.

Over 90% of web pages use UTF-8 encoding. Most modern text editors default to UTF-8 encoding when saving a file. Just FYI, other possible encoding schemes include UTF-16, ISO-8859-1, Shift-JIS (Japanese), and GBK (Chinese). Except for perhaps some niche or legacy pages, all modern web pages should use UTF-8 encoding.

`<meta name="viewport" ...>`

This tag controls the viewport's size and scaling for responsive design. Without it, a mobile browser may render the page as if it were on a wide desktop screen, resulting in a poor, zoomed-out user experience. The standard, recommended tag that provides a good mobile-friendly baseline without restricting user control follows:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The "`width=device-width`" sets the viewport width to match the device's screen width, while "`initial-scale=1.0`" sets the initial zoom level when the page loads to 1.0 (100%), preventing the page from loading zoomed in or out.

Modern web browsers do not default to these setting to maintain backwards compatibility with older sites that were designed only for desktop. This tag basically tells the web browser: yes, the web site has been designed to be responsive to mobile.

`<link rel="stylesheet" href="...">`

The standard way to link an external CSS file (***external CSS***). This is the preferred method of adding CSS and we will be discuss this when we study CSS.

`<style> … </style>`

For embedding CSS directly in the HTML document (***internal CSS***). This should generally <u>not</u> be included in production web design. However, it is much better than using *inline* CSS, and is useful for quick prototyping or situations where you just need a quick solution, such as when writing an exam about web design.

`<script src="app.js" defer></script>`

For linking an external JavaScript file (***external JavaScript***). When placed in `<head>`, often uses `defer` attribute to delay the execution until after the page has been fully parsed and the DOM tree built.

Without the `defer` attribute, the page loading is paused while the script executes, and if the script refers to anything in the DOM (say using `selectElementById` or `querySelector`), it will not be found because the DOM tree has not been built. Thus, the `defer` attribute is usually included. The script will be loaded asynchronously when the `<script … defer>` is encountered.

`<script> … </script>`

For embedding JavaScript directly in the HTML document (***internal JavaScript***). This should generally not be included in production web design. However, it is considered much better than using *inline* JavaScript, and is useful for quick prototyping or situations where you need a quick solution, such as when writing an exam about web design. Since code cannot be deferred, `<script>` tags are generally placed near the end of the HTML file, just before the closing `</body>` tag so that the DOM tree will be built and the JavaScript can access the necessary portions of the DOM.

`<meta name="description" content="This web site...">`

Search engines like Google often display this text as the snippet below your page's title in search results. A good description can improve your click-through rate, so write a compelling, accurate summary that includes primary keywords naturally. Aim for about 150to 160 characters, as search engines may truncate longer descriptions. Each page on your site should have a unique description relevant to its specific content.

`<meta http-equiv="refresh" content="5">`

This is an old, obsolete method to either:

- Refresh the current page after a set number of seconds.
- Redirect the user to a different URL after a delay.

The number of seconds is given in the `content` attribute. To redirect to another, add a `url` to the `content` attribute, as shown in this example:

```
<meta http-equiv="refresh" content="3; url=https://example.com/new-page">
```

The proper way to redirect to another page is for the server to send a HTTP return code that is in the 300's, which will instruct the browser to redirect. Search engines may penalize or distrust pages that use client-side (HTML) redirects. ***This redirect method has been included here because it has been presented in past papers of Pearson Edexcel.***