WEB 2.0 AND XHTML

Web 2.0 refers to the second generation of the World Wide Web, shifting from static, read-only pages to dynamic, user-generated content and participatory, interactive platforms. It emphasizes social networking, collaboration, and web-based applications (like YouTube, Wikipedia, and Facebook) that allow users to create, share, and edit content rather than merely consume it.

Key Characteristics of Web 2.0

- **User-Generated Content:** Users drive the content, contributing data, opinions, and media, such as blogs, wikis, and social media posts.
- **Interactivity & Collaboration:** Websites are dynamic, allowing users to interact, comment, and collaborate, moving away from the passive viewing of Web 1.0.
- **Web as a Platform:** Applications are delivered through the browser rather than installed on a desktop, enabling web services and social connectivity.
- **Improved Interoperability:** Web 2.0 uses technologies like AJAX and RSS, enabling data sharing and integration across sites.
- **Folksonomy (Tagging):** Users can categorize information collectively, making it easier to search and find content.

Web 1.0 vs. Web 2.0

- **Web 1.0:** Static, personal websites, read-only, content-centric (1990s).
- **Web 2.0:** Dynamic, social media, read-write, user-centric (2000s–present).

Examples

- **Social Networking:** Facebook, Twitter, LinkedIn.
- **Content Sharing:** YouTube, Flickr, Instagram.
- **Collaboration:** Wikipedia, Google Docs.
- **Blogs & Wikis:** WordPress, Blogger.

The term was coined by Darcy DiNucci in 1999 and popularized by O'Reilly Media in 2004, representing a shift toward a more interactive internet

Difference Between Web 1.0, Web 2.0, and Web 3.0

Web 1.0 was all about fetching and reading information. Web 2.0 is all about reading, writing, creating, and interacting with the end user. It was famously called the participative social web. Web 3.0 is the third generation of the World Wide Web, and is a vision of a decentralized web, which is currently a work in progress. It is all about reading, writing, and owning.

What is Web 1.0?

Web 1.0 refers to the first stage of the World Wide Web evolution. Earlier, there were only a few content creators in Web 1.0 with a huge majority of users who are consumers of content. Personal web pages were common, consisting mainly of static pages hosted on ISP-run web servers, or free web hosting services.

- Web 1.0 refers to the early phase of the web, roughly from 1991 to 2004. Advertisements were banned on websites while surfing the internet.
- Platforms like Ofoto allowed users to store, share, view, and print digital photos.
- Web 1.0 worked mainly as a content delivery network (CDN) used to display information on websites, including personal pages. Users were charged based on the number of pages they viewed, and directories helped them retrieve specific information.

Four Design Essentials of a Web 1.0 Site Include:

- Static pages.
- Content is served from the server's file system.
- Pages built using Server Side Includes or Common Gateway Interface (CGI).
- Frames and Tables are used to position and align the elements on a page.

Features of the Web 1.0

- Easy to connect static pages with the system via hyperlinks
- Supports elements like frames and tables with HTML 3.2
- Also has graphics and a GIF button
- Less interaction between the user and the server
- You can send **HTML** forms via mail
- Provides only a one-way publishing medium

What is Web 2.0?

In 1999, Darcy DiNucci coined the term "Web 2.0," which gained fame in 2004 at the First Web 2.0 conference (later Web 2.0 Summit) organized by Tim O'Reilly and Dale Dougherty. Web 2.0 is an enhanced version of Web 1.0, also known as the participative social web.

- **Core Focus**: Emphasizes user-generated content (e.g., blog posts, videos, reviews), usability (intuitive interfaces for all users), and interoperability (sites working seamlessly with each other via APIs and data sharing).
- **Not Technical**: Refers to changes in how web pages are designed and used, not technical specifications (e.g., shifting from static HTML pages to dynamic, user-editable platforms like wikis or social networks).

- **Key Features**: Enables interaction, collaboration, and social media dialogue in virtual communities (e.g., commenting, sharing, co-editing content in real time on platforms like YouTube, Wikipedia, or Facebook).
- **Transition**: Beneficial but gradual, without a clear moment of change (e.g., evolved through tools like blogs in the early 2000s, RSS feeds, and AJAX, improving engagement without a single "switch" event).

Web browser technologies are used in Web 2.0 development and it includes **AJAX** and JavaScript frameworks. Recently, AJAX and **JavaScript** frameworks have become very popular means of creating web 2.0 sites.

Features of the Web 2.0

- Free sorting of information, permits users to retrieve and classify the information collectively.
- Dynamic content that is responsive to user input.
- Information flows between the site owner and site users using evaluation & online commenting.
- Developed APIs to allow self-usage, such as by a software application.
- Web access leads to concerns different, from the traditional Internet user base to a wider variety of users.

Usage of Web 2.0

The social Web contains several online tools and platforms where people share their perspectives, opinions, thoughts, and experiences. Web 2.0 applications tend to interact much more with the end user. As such, the end-user is not only a user of the application but also a participant in these 8 tools mentioned below:

- Podcasting
- Blogging
- Tagging
- Curating with RSS
- Social bookmarking
- Social networking
- Social media
- Web content voting

What is Web 3.0?

Web 3.0 describes the evolution of web usage and interaction across multiple paths, upgrading the backend after Web 2.0's frontend focus.

- **Backend Shift:** Transforms the web into a readable/writable database; integrates Distributed Ledger Technology (DLT) like blockchain for decentralized, tamper-proof storage and transactions (e.g., Ethereum storing data across nodes instead of central servers).
- **Smart Contracts:** Self-executing code on blockchain that automates agreements based on user-defined conditions (e.g., a freelance payment releases only when work is verified, no intermediaries needed).
- **Data Model:** Users retain control/ownership of data via wallets or keys; platforms access it permissionally and compose customized views (e.g., one dataset powers a social feed, NFT marketplace, and analytics dashboard differently for each service).

- **Contrast to Web 2.0:** Web 2.0 enhanced user experience via frontend tools (AJAX for async updates, folksonomy tagging, social APIs); Web 3.0 rebuilds the infrastructure layer for decentralization, trustlessness, and programmable data flows.

The Semantic Web (Web 3.0) aims to organize the world's information in a more meaningful way than current search engines. It focuses on machine understanding rather than human interpretation. To achieve this, it uses declarative ontological languages like OWL to create domain-specific ontologies, allowing machines to reason about data and draw new conclusions instead of just matching keywords.

Web 3.0 is the latest and updated version of the web. Their are many different concepts and new things in this version.

Features of the Web 3.0

- **Semantic Web:** The succeeding evolution of the Web involves the Semantic Web. The semantic web improves web technologies in demand to create, share and connect content through search and analysis based on the capability to comprehend the meaning of words, rather than on keywords or numbers.
- **Artificial Intelligence:** Combining this capability with **natural language processing,** in Web 3.0, computers can distinguish information like humans to provide faster and more relevant results. They become more intelligent to fulfill the requirements of users.
- **3D Graphics:** The three-dimensional design is being used widely in websites and services in Web 3.0. Museum guides, computer games, e-commerce, geospatial contexts, etc. are all examples that use 3D graphics.
- **Connectivity:** With Web 3.0, information is more connected thanks to semantic metadata. As a result, the user experience evolves to another level of connectivity that leverages all the available information.
- **Ubiquity:** Content is accessible by multiple applications, every device is connected to the web, and the services can be used everywhere.
- **DLT and Smart Contracts:** With the help of DLT, we can create a virtually impossible-to-hack database that gives real value to digital content and virtual ownership. This technology enables a trustless society by using smart contracts that work automatically without needing a middleman, relying only on data stored in the DLT. It is a powerful tool that can improve the world and create more opportunities for everyone on the internet.
-

Differences Between the Web 1.0, Web 2.0, and Web 3.0

S. No.	Web 1.0	Web 2.0	Web 3.0
1.	Mostly Read-Only	Wildly Read-Write	Portable and Personal
2.	Company Focus	Community Focus	Individual Focus

S. No.	Web 1.0	Web 2.0	Web 3.0
3.	Home Pages	Blogs / Wikis	Live-streams / Waves
4.	Owning Content	Sharing Content	Consolidating Content
5.	Web Forms	Web Applications	Smart Applications
6.	Directories	Tagging	User behavior
7.	Page Views	Cost Per Click	User Engagement
8.	Banner Advertising	Interactive Advertising	Behavioral Advertising
9.	Britannica Online	Wikipedia	The Semantic Web
10.	HTML/Portals	XML / RSS	RDF / RDFS / OWL
11.	Data was not Focused.	Data of many was controlled by some mediatory.	Data was personalized and no use of mediatory.
12.	Information sharing is the goal.	Interaction is the goal.	Immersion is the goal.
13.	It connects information as its primary goal.	It aims to connect people.	Focuses on relating knowledge.
14.	Static websites	Introduction of web applications	Intelligent web-based functions and apps
15.	A simpler, more passive web.	An enhanced social Web	A semantic web exists.
16.	Web and File Servers, HTML, and Portals are technologies connected to Web 1.0.	AJAX, JavaScript, CSS, and HTML5 are examples of related technology.	Web 3.0 technologies include blockchain, artificial intelligence, and decentralized protocols.

S. No.	Web 1.0	Web 2.0	Web 3.0
17.	**Associated Technologies** • Web and File Servers • Search Engines (including AltaVista and Yahoo!) • E-mail accounts (Yahoo!, Hotmail) • Peer-to-Peer File Sharing (Napster, Bit Torrent) and others.	**Associated Technologies** • Frameworks for Ajax and JavaScript • Microsoft.NET • Blogs • Wikis and others.	**Associated Technologies** • Searching Using Semantics • Databases of Information • Ontologies • Intelligent Digital Personal Assistants and others.

INTRODUCTION TO XHTML

XHTML stands for **EXtensible HyperText Markup Language**. It is a reformulation of the widely used HTML as an application of XML, developed by the World Wide Web Consortium (W3C) to enforce stricter coding standards and improve compatibility with other XML-based technologies and various devices.

Key Characteristics

- **Strict Syntax:** Unlike HTML, which is lenient with errors, XHTML requires documents to be "well-formed," meaning all elements must be properly nested and closed. A single syntax error may prevent the browser from displaying the page correctly.
- **Case-Sensitive:** All element and attribute names in XHTML must be in lowercase.
- **Quoted Attributes:** All attribute values must be enclosed in quotation marks (e.g., `width="220"` instead
- **No Attribute Minimization:** Attributes cannot be minimized; they must be written in full attribute-value pairs (e.g., `<option selected="selected">` instead of `<option selected>`).
- **Document Type Definition (DTD):** Every XHTML document must include a `<!DOCTYPE>` declaration that references one of three DTDs (Strict, Transitional, or Frameset), specifying the rules for the markup language.
- **XML Compatibility:** Being XML-based, XHTML documents can be readily viewed, edited, and validated with standard XML tools and integrate seamlessly with other XML-based technologies.

Why was it developed?

XHTML was created to address the inconsistencies in how different web browsers handled the flexible syntax of older HTML versions. By adhering to the rigorous rules of XML, XHTML aimed to create a more consistent, reliable, and future-proof standard for web content that could be easily processed by both machines and various devices, including early mobile phones and Braille readers.

Current Relevance

While XHTML was once considered the successor to HTML, its strictness and lack of support for true XML parsing in older versions of Internet Explorer limited its widespread adoption. The web development

community largely moved to HTML5, which offers greater flexibility, new features (like native audio and video), and robust error handling, making it the dominant standard today.
However, XHTML's principles of writing clean, well-structured, and semantically correct code continue to influence modern web development best practices

STANDARD XHTML DOCUMENT STRUCTURE

A standard XHTML document has a strict, well-formed structure that includes an optional XML declaration, a mandatory **DOCTYPE declaration**, and a core structure of `<html>`, `<head>`, and `<body>` elements.

Example of a Standard XHTML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Document Title</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<body>
    <!-- Your content goes here -->
    <p>This is a paragraph.</p>
</body>
</html>
```

Required Structural Components

- **XML Declaration:** While optional if using UTF-8 or UTF-16 encoding, it is strongly encouraged by the W3C and typically the first line of the document. It specifies the XML version and character encoding.
- **DOCTYPE Declaration:** This mandatory declaration appears before the root element and refers to a Document Type Definition (DTD), which defines the rules for the markup language. The three DTDs for XHTML 1.0 are Strict, Transitional, and Frameset.
- **`<html>` (Root Element):** This encloses the entire document and must include the `xmlns` attribute to specify the XHTML namespace (`http://www.w3.org/1999/xhtml`). It is also recommended to include `xml:lang` and `lang` attributes to declare the document's primary language.
- **`<head>`:** Contains meta-information about the document, such as the title, links to stylesheets, and scripts. This information is for machine processing and does not appear in the visible body of the page.
- **`<title>`:** A mandatory element within the `<head>` that specifies the title of the document, which is displayed in the browser's title bar or tab.

- **`<body>`:** This element contains all the visible content of the web page, including text, images, links, and other structural elements (e.g., `<p>`, `<div>`, `<h1>`).

Key XHTML Syntax Rules

XHTML is stricter than HTML and requires adherence to specific XML syntax rules.

- **Lowercase Tags and Attributes:** All element and attribute names must be in lowercase.
- **Proper Nesting:** Elements must be properly nested (e.g., `<b><i>text</i></b>` is correct, `<b><i>text</b></i>` is incorrect).
- **Closed Tags:** All non-empty elements must have a closing tag (e.g., `<p>text</p>`). Empty elements must be self-closing (e.g., `<br />` or `<img src="image.jpg" alt="Description" />`).
- **Quoted Attribute Values:** All attribute values must be enclosed in quotation marks (e.g., `<a href="page.html">Link</a>`, not `<a href=page.html>Link</a>`).
- **No Attribute Minimization:** Attributes must be written in full attribute-value pairs (e.g., `<option selected="selected">`, not `<option selected>`).
    - **Elements of XHTML:**

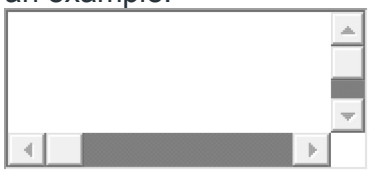
XHTML Element	Description
<!DOCTYPE>	Used to declare the Document Type Definition (DTD), specifying the rules for the markup language, ensuring proper rendering in browsers.
<html>	Encloses the entire HTML or XHTML document, serving as the root element.
<head>	Contains meta-information about the document, such as the title, character set, linked stylesheets, and other essential elements.
<title>	Nested within the head section, specifies the title of the document, displayed in the browser's title bar or tab.
<body>	Encloses the content of the web page, including text, images, links, and other HTML elements. It represents the visible part of the document displayed in the browser.

- When creating an XHTML web page, it is necessary to include a DTD (Document Type Definition) declaration. There are three types of DTD which are discussed below:
- ## Transitional DTD:
- It is supported by the older browsers which do not have inbuilt cascading style sheets supports. Several attributes are enclosed in the body tag which are not allowed in strict DTD.
- **Syntax:**

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">

<html xmlns="https://www.w3.org/1999/xhtml/" xml:lang="en"
lang="en">
```

- **Example:** In this example we will see the code for writing an XHTML document with an example.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">
<html xmlns="https://www.w3.org/1999/xhtml/" xml:lang="en"
lang="en">

<head>
    <title>Transitional DTD XHTML</title>
</head>

<body bgcolor="#dae1ed">
    <div style="color:#090;font-size:40px;
                font-weight:bold;text-align:center;
                margin-bottom:-25px;">GeeksforGeeks</div>
    <p style="text-align:center;font-size:20px;">
        A computer science portal</p>
    <p style="text-align:center;font-size:20px;">
        Option to choose month:
        <select name="month">
            <option selected="selected">January</option>
            <option>February</option>
            <option>March</option>
            <option>April</option>
            <option>May</option>
            <option>June</option>
```

```
            <option>July</option>
            <option>Augusy</option>
            <option>September</option>
            <option>October</option>
            <option>November</option>
            <option>December</option>
        </select>
    </p>
</body>

</html>
```

- **Output:**

Strict DTD:

- Strict DTD is used when XHTML page contains only markup language. Strict DTD is used together with cascading style sheets, because this attribute does not allow CSS property in body tag.
- **Syntax:**

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">
<html xmlns="https://www.w3.org/1999/xhtml/" xml:lang="en"
lang="en">
```

- **Example 2:** In this example we will see the code for writing an XHTML document with an example for strict DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">
<html xmlns="https://www.w3.org/1999/xhtml/" xml:lang="en"
lang="en">

<head>
    <title>Strict DTD XHTML</title>
</head>

<body>
    <div style="color:#090;font-size:40px;
                font-weight:bold;text-align:center;
                margin-bottom:-25px;">GeeksforGeeks</div>
    <p style="text-align:center;font-size:20px;">
        A computer science portal</p>
    <p style="text-align:center;font-size:20px;">
        Option to choose month:
        <select name="month">
            <option selected="selected">January</option>
            <option>February</option>
            <option>March</option>
            <option>April</option>
            <option>May</option>
            <option>June</option>
            <option>July</option>
            <option>Augusy</option>
            <option>September</option>
            <option>October</option>
            <option>November</option>
            <option>December</option>
        </select>
    </p>
</body>

</html>
```

Frameset DTD:

The frameset DTD is used when XHTML page contains frames. This DTD is identical to the HTML 4.01 Transitional DTD except for the content model of the HTML element.

Syntax:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"DTD/xhtml1-frameset.dtd">
<html xmlns="https://www.w3.org/1999/xhtml/" xml:lang="en"
lang="en">
```

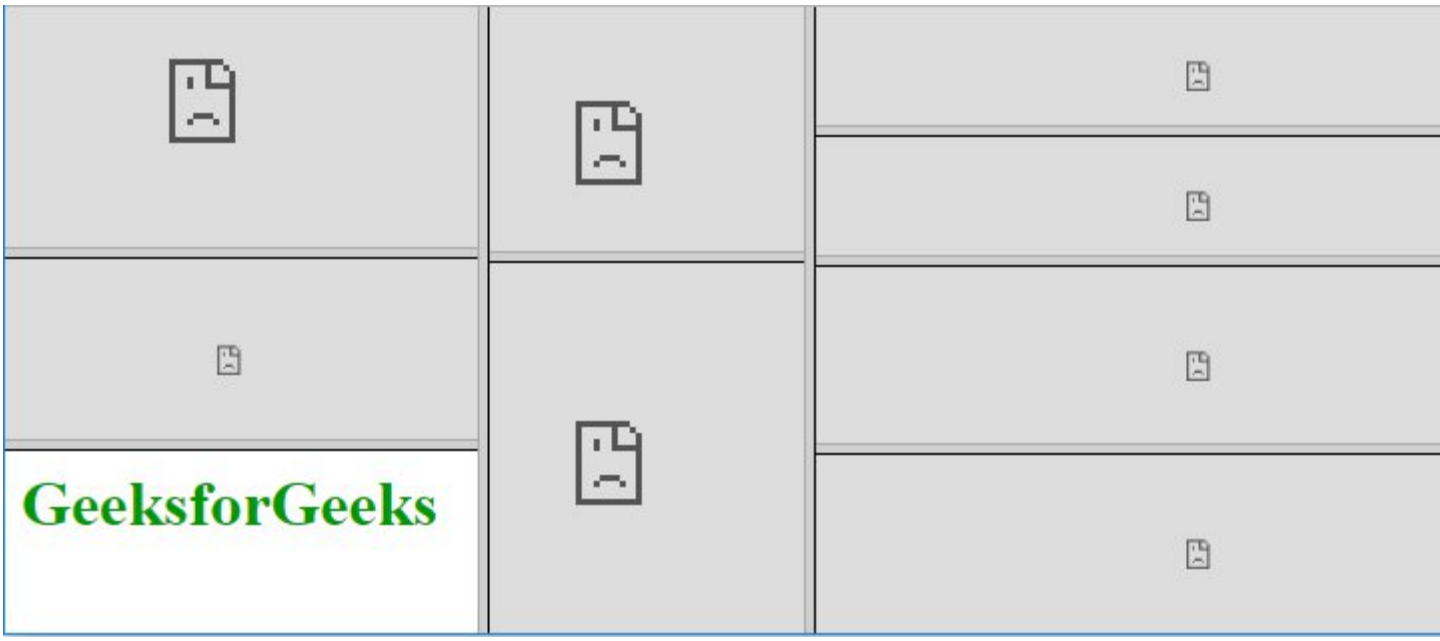
Example 2: In this example we will see the code for writing an XHTML document with an example for frameset DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"https://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="https://www.w3.org/1999/xhtml/"
      xml:lang="en" lang="en">

<head>
    <title>Frameset DTD XHTML</title>
</head>
<frameset cols="30%, 20%, *">
    <frameset rows="40%, 30%, *">
        <frame src="gfg.html" />
        <frame src="gfg1.html" />
        <frame src="geeks.html" />
    </frameset>
    <frameset rows="40%, 60%">
        <frame src="g4g.html" />
        <frame src="g4g1.html" />
    </frameset>
    <frameset rows="20%, 20%, 30%, *">
        <frame src="geeksforgeeks.html" />
        <frame src="geeksforgeeks1.html" />
        <frame src="geeksforgeeks2.html" />
        <frame src="geeksforgeeks3.html" />
    </frameset>
</frameset>

</html>
```

Output:



Why use XHTML?

- XHTML documents are validated with standard XML tools.
- It is easily to maintain, convert, edit document in the long run.
- It is used to define the quality standard of web pages.
- XHTML is an official standard of the W3C, your website becomes more compatible and accurate with many browsers.

Benefits of XHTML:

- All XHTML tags must have closing tags and are nested correctly. This generates cleaner code.
- XHTML documents are lean which means they use less bandwidth. This reduces cost particularly if your web site has 1000s of pages.
- XHTML documents are well formatted well–formed and can easily be transported to wireless devices, Braille readers and other specialized web environments.
- All new developments will be in XML (of which XHTML is an application).
- XHTML works in association with CSS to create web pages that can easily be updated.

Difference Between HTML and XHTML:

HTML	XHTML
HTML or HyperText Markup Language is the main markup language for creating web pages	XHTML (Extensible HyperText Markup Language) is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML)
Flexible framework requiring lenient HTML specific parser	Restrictive subset of XML which needs to be parsed with standard XML parsers
Proposed by Tim Berners-Lee in 1987	World Wide Web Consortium Recommendation in 2000.
Application of Standard Generalized Markup Language (SGML).	Application of XML
Extended from SGML.	Extended from XML, HTML

BASIC SYNTAX OF XHTML

XHTML syntax is a stricter, cleaner version of HTML based on XML rules. Adhering to these rules ensures well-formed, interoperable documents.

Core XHTML Syntax Rules

- **Mandatory `DOCTYPE` Declaration:** Every XHTML document must start with a `DOCTYPE` declaration.
- **Mandatory Elements:** The `<html>`, `<head>`, `<title>`, and `<body>` elements are mandatory and must be present.
- **Root Element with Namespace:** The entire document content must be enclosed within an `<html>` root element which must also specify the XML namespace using the `xmlns` attribute.
- **Lowercase Tags and Attributes:** All element and attribute names must be in lowercase, as XHTML is case-sensitive.
- **Properly Nested Elements:** Elements must always be properly nested. For example, `<b><i>text</i></b>` is correct, while `<b><i>text</b></i>` is not.
- **All Tags Must Be Closed:** Every element must have a closing tag.
    - Non-empty elements use a start and end tag pair, such as `<p>This is a paragraph</p>`.

- Empty elements (which do not wrap content) must be self-closing with a trailing slash and space, such as `<br />` or `<img src="image.gif" alt="Description" />`.

- **Attribute Values Must Be Quoted:** All attribute values must be enclosed in quotation marks.
- **Attribute Minimization Forbidden:** Attributes cannot be minimized. For example, use `checked="checked"` instead of just `checked`.

Minimal XHTML Document Example

A basic, valid XHTML 1.0 Transitional document follows this structure:

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Document Title</title>
</head>
<body>
    <!-- Your content goes here -->
</body>
</html>
```

XHTML syntax is very similar to HTML syntax and almost all the valid HTML elements are valid in XHTML as well. But when you write an XHTML document, you need to pay a bit extra attention to make your HTML document compliant to XHTML.

Here are the important points to remember while writing a new XHTML document or converting existing HTML document into XHTML document –

- Write a DOCTYPE declaration at the start of the XHTML document.
- Write all XHTML tags and attributes in lower case only.
- Close all XHTML tags properly.
- Nest all the tags properly.
- Quote all the attribute values.
- Forbid Attribute minimization.
- Replace the **name** attribute with the **id** attribute.
- Deprecate the **language** attribute of the script tag.

Here is the detail explanation of the above XHTML rules –

DOCTYPE Declaration

All XHTML documents must have a DOCTYPE declaration at the start. There are three types of DOCTYPE declarations, which are discussed in detail in XHTML Doctypes chapter. Here is an example of using DOCTYPE –

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Case Sensitivity

XHTML is case sensitive markup language. All the XHTML tags and attributes need to be written in lower case only.

```
<!-- This is invalid in XHTML -->
<A Href="/xhtml/xhtml_tutorial.html">XHTML Tutorial</A>

<!-- Correct XHTML way of writing this is as follows -->
<a href="/xhtml/xhtml_tutorial.html">XHTML Tutorial</a>
```

In the example, **Href** and anchor tag **A** are not in lower case, so it is incorrect.

Closing the Tags

Each and every XHTML tag should have an equivalent closing tag, even empty elements should also have closing tags. Here is an example showing valid and invalid ways of using tags –

```
<!-- This is invalid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.

<!-- This is also invalid in XHTML -->
<img src="/images/xhtml.gif" >
```

The following syntax shows the correct way of writing above tags in XHTML. Difference is that, here we have closed both the tags properly.

```
<!-- This is valid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.</p>

<!-- This is also valid now -->
<img src="/images/xhtml.gif" />
```

Attribute Quotes

All the values of XHTML attributes must be quoted. Otherwise, your XHTML document is assumed as an invalid document. Here is the example showing syntax –

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" width=250 height=50 />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif" width="250" height="50" />
```

Attribute Minimization

XHTML does not allow attribute minimization. It means you need to explicitly state the attribute and its value. The following example shows the difference –

```
<!-- This is invalid in XHTML -->
<option selected>

<!-- Correct XHTML way of writing this is as follows -->
<option selected="selected">
```

Here is a list of the minimized attributes in HTML and the way you need to write them in XHTML –

HTML Style	XHTML Style
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"

noresize	noresize="noresize"

The *id* Attribute

The id attribute replaces the name attribute. Instead of using name = "name", XHTML prefers to use id = "id". The following example shows how −

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" name="xhtml_logo" />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif" id="xhtml_logo" />
```

The *language* Attribute

The language attribute of the script tag is deprecated. The following example shows this difference −

```
<!-- This is invalid in XHTML -->

<script language="JavaScript" type="text/JavaScript">
   document.write("Hello XHTML!");
</script>

<!-- Correct XHTML way of writing this is as follows -->

<script type="text/JavaScript">
   document.write("Hello XHTML!");
</script>
```

Nested Tags

You must nest all the XHTML tags properly. Otherwise your document is assumed as an incorrect XHTML document. The following example shows the syntax −

```
<!-- This is invalid in XHTML -->
<b><i> This text is bold and italic</b></i>

<!-- Correct XHTML way of writing this is as follows -->
<b><i> This text is bold and italic</i></b>
```

Element Prohibitions

The following elements are not allowed to have any other element inside them. This prohibition applies to all depths of nesting. Means, it includes all the descending elements.

Element	Prohibition
<a>	Must not contain other <a> elements.
<pre>	Must not contain the <img>, <object>, <big>, <small>, <sub>, or <sup> elements.
<button>	Must not contain the <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe> or <isindex> elements.
<label>	Must not contain other <label> elements.
<form>	Must not contain other <form> elements.

A Minimal XHTML Document

The following example shows you a minimum content of an XHTML 1.0 document –

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/TR/xhtml1" xml:lang="en" lang="en">
   <head>
      <title>Every document must have a title</title>
   </head>

   <body>
      ...your content goes here...
   </body>
</html>
```

IMAGES IN XHTML

To embed an image in XHTML, you use the **`<img>` element** with a self-closing tag syntax and essential attributes like `src` and `alt`. The general format is:

xhtml

```
<img src="image_path.jpg" alt="Description of the image" />
```

Key Attributes

The `<img>` tag uses several important attributes:

- **`src`**: Specifies the path or URL to the image file. This attribute is required.
  - Example using a relative path: `src="images/logo.png"`.
  - Example using an absolute URL: `src="https://www.example.com"`.
- **`alt`**: Provides alternate text for the image, which is displayed if the image cannot be loaded or is being accessed by a screen reader for accessibility. This attribute is also required in XHTML.
- **`width` and `height`**: These attributes define the dimensions of the image in pixels, helping the browser render the page layout more quickly.
  - Example: `width="100" height="75"`.
  - It is often recommended to use CSS for sizing to separate presentation from structure.
- **`title`**: Adds a tooltip that appears when the user hovers their mouse over the image.

XHTML-Specific Syntax Notes

- **Self-closing tag**: In XHTML, all elements must be properly closed. The `<img>` tag, which does not wrap content, is "self-closing" and must end with `/>`. The space before the slash is recommended for maximum browser compatibility.
- **Case Sensitivity**: XHTML is case-sensitive and typically uses lowercase for all element and attribute names (e.g., `<img>` not `<IMG>`).

Examples

- **Basic image insertion**:

xhtml

```
<img src="mypic.jpg" alt="A photo of a sunset" width="400" height="300" />
```

- **Image as a link**: To make an image clickable, nest the `<img>` tag inside an anchor `<a>` tag:

xhtml

```
<a href="destination_page.html">
    <img src="button.gif" alt="Clickable button to destination page" />
</a>
```

- **Using CSS for styling**: Instead of the `align` attribute (which is deprecated), use CSS for positioning, such as floating the image:

xhtml

```
<img src="mypic.jpg" alt="A photo" style="float:left; margin:10px;" />
```

HYPER TEXT LINKS IN XHTML

Hypertext links in XHTML are created using the **<a> (anchor) element** and the **href attribute**, which functions similarly to HTML but with stricter syntax rules (e.g., all elements must be properly nested and closed). The `href` attribute specifies the destination URI (Uniform Resource Identifier).

Syntax and Structure

The basic syntax for a hypertext link is:

xhtml

```
<a href="url_of_destination">Link text or image</a>
```

- **<a>...**: The anchor element defines the start and end of the link.
- **href attribute**: This is a mandatory attribute in a link definition, which specifies the destination address or fragment identifier.
- **Content**: The content between the opening and closing `<a>` tags (text or an image) is the clickable part of the link displayed to the user.

Types of Links

- **External Links**: Link to a resource on a different website using an absolute URL (e.g., `<a href="https://www.w3.org/">W3C Web site</a>`).
- **Internal Links (Relative URLs)**: Link to another page within the same website using a relative URL (e.g., `<a href="second.htm">Go to next page.</a>`).
- **Anchor Links (Fragment Identifiers)**: Link to a specific section within the same document or another document by using the `#` symbol followed by the target element's `id`. For example:
  - To link to an ID on the same page: `<a href="#section2">Go to Section 2</a>`.
  - To link to an ID on another page: `<a href="otherpage.htm#section2">Go to Other Page Section 2</a>`.

Key XHTML Attributes

XHTML links support several attributes to control behavior and provide additional information:

W3Schools +2

Attribute	Description	Example
`href`	Specifies the destination URI of the link.	`<a href="page.xhtml">...</a>`
`target`	Specifies where to open the linked document (e.g., `_blank` for a new window/tab, `_self` for the current frame).	`<a href="url" target="_blank">...</a>`
`title`	Provides extra information about the	`<a href="url" title="Visit our`

	link, often shown as a tooltip on hover.	`tutorial">...</a>`
`id`	Creates a unique identifier for an anchor target within a document, which can be linked to via a fragment identifier.	`<p id="section2">...</p>`
`hreflang`	Specifies the language of the linked resource.	`<a href="url" hreflang="fr">...</a>`
`type`	Gives an advisory hint as to the content type of the destination (e.g., "text/html").	`<a href="url" type="application/pdf">...</a>`

XHTML Specific Rules

XHTML, being an application of XML, enforces stricter rules than HTML:

- Elements and attribute names must be in **lowercase**.
- All non-empty elements must have a **closing tag** (e.g., `</a>`).
- Documents must be **well-formed**, meaning all tags are properly nested.

These requirements ensure that XHTML documents can be parsed as a strict XML tree structure.

LIST AND TABLES IN XHTML

In XHTML, lists and tables are used to organize content logically and display data in structured formats. Both follow the strict syntax rules of XHTML, such as using lowercase tags, quoting all attributes, and properly closing all elements.

XHTML Lists

XHTML supports three main types of lists, created using specific tag combinations:

- **Unordered Lists (`<ul>`):** Used for items where the order does not matter (e.g., a shopping list). Items are typically marked with bullets.
  - `<ul>` defines the unordered list.
  - `<li>` defines each individual list item within the list.

xhtml

```
<ul>
  <li>First item</li>
  <li>Second item</li>
</ul>
```

- **Ordered Lists (`<ol>`):** Used for items where the sequence is important (e.g., steps in a recipe). Items are typically enumerated with numbers or letters.
  - `<ol>` defines the ordered list.

- `<li>` defines each individual list item within the list.

```xhtml
<ol>
  <li>Step one</li>
  <li>Step two</li>
</ol>
```

- **Definition Lists (`<dl>`):** Used to display terms and their corresponding definitions or explanations (e.g., a glossary).
- `<dl>` defines the description list.
- `<dt>` defines the definition term.
- `<dd>` defines the description/definition for the term, which is typically indented.

```xhtml
<dl>
  <dt>HTML</dt>
  <dd>A language used to create webpages.</dd>
  <dt>CSS</dt>
  <dd>Used for styling web pages.</dd>
</dl>
```

XHTML Tables

Tables are used to organize data into rows and columns. Key elements for creating a table in XHTML include:

- **`<table>`**: The container element that defines the entire table.
- **`<caption>`**: Defines a title or caption for the table, usually appearing above it.
- **`<tr>`**: Defines a single table row.
- **`<th>`**: Defines a table header cell. Text within this is typically bold and centered.
- **`<td>`**: Defines a standard table data cell.
- **`<thead>`, `<tbody>`, `<tfoot>`**: Optional elements used to group rows into a table head, body, and foot section, respectively. This allows browsers to scroll the body independently or repeat the head/foot when printing long tables.

A basic XHTML table structure looks like this:

```xhtml
<table>
  <caption>Monthly Expenses</caption>
  <thead>
    <tr>
      <th>Category</th>
      <th>Cost</th>
    </tr>
```

```
  </thead>
  <tbody>
    <tr>
      <td>Housing</td>
      <td>$1000</td>
    </tr>
    <tr>
      <td>Food</td>
      <td>$400</td>
    </tr>
  </tbody>
</table>
```

CREATION FORM IN XHTML

Creating a form in XHTML is very similar to HTML, but requires strict adherence to XML syntax rules, such as all tags being properly nested and closed, and attributes enclosed in quotes.

Essential XHTML Document Structure

Every XHTML document must include a `DOCTYPE` declaration and the root `<html>` element with an `xmlns` attribute.

xhtml

```
<?xml version="1.0" encoding="utf-8"?>
<!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" xml:lang="en">
<head>
    <title>Creation Form</title>
</head>
<body>

    <!-- Form content goes here -->

</body>
</html>
```

Form Creation

The form itself is defined using the `<form>` element, which acts as a container for various input elements. The two most important attributes of the `<form>` tag are `action` and `method`.

- **`action`**: Specifies the URL of the server-side script or application that will process the form data (e.g., a PHP or Perl script).
- **`method`**: Specifies the HTTP method to be used when submitting the data, typically `"post"` to hide the information during transfer or `"get"` to append it to the URL.

Example XHTML Form

xhtml

```
<form action="process_form.php" method="post">
    <!-- Form elements go here -->
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" /><br />

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" /><br />

    <label for="comments">Comments:</label>
    <textarea id="comments" name="comments" rows="5" cols="40"></textarea><br
/>

    <input type="submit" value="Submit" />
    <input type="reset" value="Reset" />
</form>
```

Key Form Elements

The `<form>` element contains various input elements defined by tags such as `<input>`, `<textarea>`, and `<select>`. All form widgets should have a `name` attribute to identify the data when it is sent to the server.

- **`<input type="text" />`**: Creates a single-line text input field.
- **`<input type="password" />`**: Creates a single-line password input field where characters are masked.
- **`<textarea></textarea>`**: Defines a multi-line text input area.
- **`<input type="radio" />`**: Creates a radio button for selecting one option from many choices.
- **`<input type="checkbox" />`**: Creates a checkbox for selecting zero or more options.
- **`<select></select>`**: Creates a dropdown list for selecting options, with options defined by `<option>` tags inside.
- **`<input type="submit" />`**: Creates a button that submits the form data to the server.
- **`<label for="id">`**: Associates a label with an input element, improving accessibility. The `for` attribute's value must match the `id` of the related input element.

INTERNAL LINKING AND META ELEMENTS XHTML

In XHTML, **internal linking** allows navigation within the same document using anchor elements and `id` attributes, while **meta elements** provide hidden, machine-readable information about the document's metadata. Both are essential for structured, accessible, and search-engine-friendly web pages.

Internal Linking in XHTML

Internal links allow users to jump to specific sections within a single, long XHTML document. This is achieved using two components: the destination target and the hyperlink.

1. Defining the Destination

An `id` attribute is used to uniquely identify an element (like a heading or a paragraph) within the document. The `id` value must be unique to the page.

xhtml

```
<h2 id="section2">Section Two Heading</h2>
```

In older HTML versions, the `name` attribute on an `<a>` tag was used, but in XHTML, the `id` attribute is the standard mechanism.

2. Creating the Link

The anchor (`<a>`) element is used to create the hyperlink. The `href` attribute value consists of a hash symbol (`#`) followed by the `id` of the destination element.

xhtml

```
<a href="#section2">Jump to Section Two</a>
```

If the link is to a section in a *different* document, the URL of that document is added before the hash symbol (e.g., `<a href="document.xhtml#section2">...</a>`).

Meta Elements in XHTML

`meta` elements provide metadata (data about data) that is not displayed as visible content on the webpage. They are placed within the `<head>` section of the document and help browsers, search engines, and other web services understand how to handle and display the page.

In XHTML, `meta` tags must be properly closed, typically using the empty-element tag syntax `<meta ... />`.

Common Meta Elements and Attributes:

- **`name` and `content`**: Define a property name and its value.
  - **Description**: Used by search engines as the snippet in search results.

    xhtml

    ```
    <meta name="description" content="A description of the webpage content" />
    ```

  - **Keywords**: A list of keywords relevant to the page content (less important for modern SEO).

    xhtml

    ```
    <meta name="keywords" content="xhtml, internal linking, meta elements" />
    ```

  - **Author**: Specifies the author of the page.

    xhtml

    ```
    <meta name="author" content="John Doe" />
    ```

  - **Viewport**: Essential for responsive design, instructing the browser on how to control page dimensions and scaling on mobile devices.

    xhtml

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

- **`http-equiv` and `content`**: Emulate HTTP headers.
  - **Content-Type**: Specifies the character set of the document (though in HTML5, the `charset` attribute is more common).

xhtml

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

- **Refresh**: Automatically refreshes the document after a specified number of seconds.

xhtml

```
<meta http-equiv="refresh" content="30" />
```

- **`charset`**: A shorthand for specifying character encoding in HTML5, which is also allowed for migration to/from XHTML.

xhtml

```
<meta charset="UTF-8" />
```