Markup Languages of the Web

Péter Jeszenszky

October 16, 2024

Markup Languages of the Web

- HTML
- SVG
- MathML
- Markdown (discussed separately)

HTML

- "HTML is the World Wide Web's core markup language."
- "[...] a semantic-level markup language and associated semantic-level scripting APIs for authoring accessible pages on the Web ranging from static documents to dynamic applications."
  - See: HTML Living Standard

The Birth of HTML

- Tim Berners-Lee (TBL) invented and created HTML.
- The first public website: http://info.cern.ch/ (launched on: August 6, 1991)
  - See: Restoring the first website
- The first publicly available technical documentation about HTML: https://info.cern.ch/hypertext/WWW/MarkUp/Tags.html

Major Versions of HTML (1)

- HTML 4.01 (superseded)
- XHTML 1.0 (superseded)
- XHTML 1.1 (superseded)
- HTML5 (current version)

Major Versions of HTML (2)

Version statistics:

- W3Techs: Usage statistics and market share of HTML for websites
  HTML5 is used by 96.3% of all the websites who use HTML.

HTML 4.01 (1)

HTML 4.01 Specification (W3C Recommendation, 24 December 1999; superseded: 27 March 2018)

- The last SGML-based HTML version.
- Media type: `text/html`

HTML 4.01 (2)

Document type declarations:

- **Strict**:

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

- **Transitional**:

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

- **Frameset**:

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

XHTML

- HTML defined as an XML application imposes stricter constraints on documents, thus, they can be processed more easily.
- This is particularly important for devices that have limited capabilities compared to desktop computers (e.g., mobile devices).
- XHTML and its modularization enable the combination of XHTML with other XML applications.
    - For example, embedding MathML and SVG in XHTML documents – however, the resulting documents are no longer considered to be XHTML documents.

XHTML 1.0 (1)

XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0 (W3C Recommendation, 26 January 2000; superseded: 27 March 2018)

- A reformulation of HTML 4 as an XML application.
- Media type: `application/xhtml+xml`

XHTML 1.0 (2)

Document type declarations:

- **Strict**:

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

- **Transitional**:

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

- **Frameset**:

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

XHTML 1.1 (1)

XHTML Modularization 1.1 – Second Edition (W3C Recommendation, 29 July 2010; superseded: 27 March 2018)

- Modularization provides a means for subsetting and extending XHTML.
- Can be implemented using either DTD or XML Schema.
- Provides a set of standard modules.
  - Examples: *Frames* (`frame`, `frameset`, `noframes` elements), *Hypertext* (`a` element), *Text* (`div`, `h1`, `p`, ... elements), ...
- Combining multiple modules enables the creation of so-called hybrid document types.

XHTML 1.1 (2)

XHTML Basic 1.1 – Second Edition (W3C Recommendation, 23 November 2010; superseded: 27 March 2018)

- A subset of XHTML suitable to be used on a wide range of devices (e.g., mobile phones, PDAs, e-book readers, set top boxes).
- The document type definition is implemented using XHTML modules as defined in the *XHTML Modularization* recommendation.
- Document type declarations:

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML Basic 1.1//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
```

- Media type: `application/xhtml+xml`

XHTML 2.0

XHTML 2.0 (W3C Working Group Note, 16 December 2010)

- A new language that was not compatible with the earlier HTML and XHTML languages.
- The W3C decided to not continue the development of XHTML 2.0.
- HTML5 has taken over its originally intended role.
- See: Frequently Asked Questions (FAQ) about the future of XHTML

HTML5: History

See: HTML Living Standard – Introduction – History

HTML5: Development (1)

- Originally, the HTML5 specification was developed by the WHATWG.
- The W3C joined to the development of HTML5 in 2007.
- See:
  - Ian Hickson. The WHATWG Blog – W3C restarts HTML effort. 7 March 2007.
  - W3C Relaunches HTML Activity. 7 March 2007.

HTML5: Development (2)

Between July 2012 and June 2019, the WHATWG and the W3C had been working on separate specifications that followed a different development model.

- The W3C started working on the next version (HTML 5.1) after the publication of the HTML 5.0 specification as a Recommendation.
- The WHATWG specification will never be completed, it is under continuous development ("living standard").

HTML5: Development (3)

- Until June 2019, within the W3C the Web Platform Working Group developed the specifications related to the HTML language.
- The W3C's specifications was based on the WHATWG's specification.
  - The W3C had published certain parts in separate documents.

HTML5: Development (4)

- On May 28, 2019, the two organizations signed an agreement to collaborate on the development of a single version of the HTML and DOM specifications.
  - See: Jeff Jaffe. W3C and WHATWG to work together to advance the open Web platform. 28 May 2019.
- Collaboration agreement:
  - HTML and DOM shall be developed principally in the WHATWG.
  - W3C intends to approve and release the WHATWG specifications as W3C recommendations.
  - See: Memorandum of Understanding Between W3C and WHATWG. May 28, 2019.
- In the following, within the W3C the HTML Working Group is responsible for the development of HTML.

HTML5 Standard

- **WHATWG**:
  - HTML Living Standard
  - HTML: The Living Standard – Edition for Web Developers
    - Removes information that only browser vendors need to know.
- **W3C**: Currently, the URI https://www.w3.org/TR/html is redirected to the WHATWG specification.

HTML Elements (1)

- Elements, attributes, and attribute values in HTML are defined to have certain meanings (semantics).
  - For example, the `ol` element represents an ordered list, and the `lang` attribute represents the language of the content.
- Authors must not use elements, attributes, or attribute values for purposes other than their appropriate intended semantic purpose.

HTML Elements (2)

- The majority of presentational features from previous versions of HTML are no longer allowed.
- The problems of presentational markup:
  - The use of presentational elements leads to poorer accessibility.
  - Higher cost of maintenance.
  - Larger document sizes.

HTML Elements (3)

The only remaining presentational markup features in HTML are the `style` attribute and the `style` element.

HTML Elements (4)

The following elements that were previously presentational have been redefined to be media-independent:

Element	HTML 4.01, XHTML 1.0	HTML5
`b`	Bold font	Keywords
`i`	Italic font	Alternate voice
`hr`	Horizontal rule	Thematic break
`small`	Smaller font	Side comment
`s`	Strike-through	Inaccurate text
`u`	Underline	Unarticulated annotation (e.g., misspelt text)

HTML Elements (5)

- Each element defined in this specification has a content model: a description of the element's expected contents.
- An HTML element must have contents that match the requirements described in the element's content model.

HTML Elements (6)

Each element in HTML falls into zero or more of the following categories:

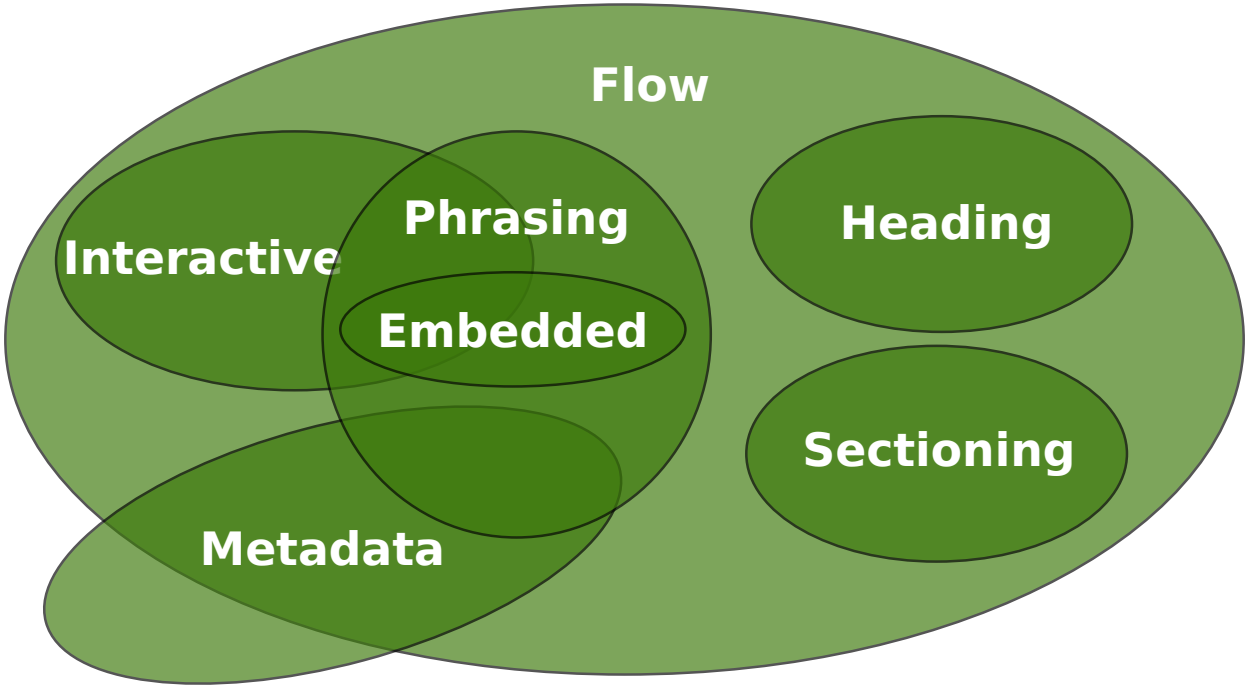

Figure 1: Source: https://html.spec.whatwg.org/images/content-venn.svg

HTML Elements (7)

Foreign elements: Elements from the MathML namespace and the SVG namespace.

HTML Elements (8)

Elements introduced in HTML5 for better structuring:

- `article`
- `aside`
- `figure`
- `footer`
- `header`
- `hgroup`
- `nav`
- `section`

HTML Elements (9)

Other new elements introduced in HTML5:

- `audio`
- `canvas`
- `data`
- `dialog`
- `math`
- `meter`
- `picture`
- `progress`
- `summary/details`
- `time`
- `video`
- …

HTML Elements (10)

All HTML elements:

- HTML Standard – Index – Elements
- MDN Web Docs – HTML Elements Reference

HTML Elements (11)

Example: https://html-basics.surge.sh/en

Semantic HTML (1)

Example:

```
<h1>This is a top-level heading</h1>
<span style="font-size: 2em; margin: 0.67em 0;">
  This is not a top-level heading,
  although it looks like that
</span>
```

Source: Semantics in HTML

Semantic HTML (2)

Benefits include:

- Good for Search Engine Optimization (SEO)
- Improves maintainability
- Improves web accessibility

See: HTML: A good basis for accessibility

Semantic HTML (3)

Related concept: **web accessibility**

- Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them.
- See: https://www.w3.org/mission/accessibility/

Global HTML Attributes

Attributes that may be specified on all HTML elements:

- `class`
- `dir`
- `id`
- `lang`
- `style`
- `title`
- `xml:lang` (should be used only in the XML syntax)
- Custom data attributes
- ...

See:
https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes

`class` Attribute

- Authors can use the `class` attribute to extend elements, effectively creating their own elements, while using the most applicable existing "real" HTML element.
- When specified on HTML elements, the `class` attribute must have a value that is a set of space-separated tokens representing the various classes that the element belongs to.
- Examples:

```
<p class="author">...</p>
<p class="note">...</p>
<p class="warning">...</p>
<p class="note important">...</p>
```

Custom Data Attributes (1)

- A custom data attribute is an attribute whose name starts with the string `data-` and has at least one character after the hyphen.
- Intended to store custom data, state, annotations, and similar, private to the page or application, for which there are no more appropriate attributes or elements.
- Every HTML element may have any number of custom data attributes specified, with any value.

Custom Data Attributes (2)

Example:

```
<form>
  <p>
    <label for="house">House:</label>
    <select name="house" id="house" required>
      <option value="">--Please choose an option--</option>
      <option value="gryffindor"
        data-color-primary="red"
        data-color-secondary="gold">Gryffindor</option>
      <!-- ... --->
    </select>
  </p>
</form>
```

Custom Data Attributes (3)

Example:

```
document.querySelectorAll('a').forEach(function (element) {
  element.addEventListener('mouseenter', function (event) {
    const a = event.currentTarget;
    const timer = setTimeout(function () {
      window.location.href = a.href;
    }, 3000);
    // Storing the timer in the data-timer attribute:
    a.dataset.timer = timer;
  });
  element.addEventListener('mouseleave', function (event) {
    // Getting the timer from the data-timer attribute:
    const timer = event.currentTarget.dataset.timer;
    clearTimeout(timer);
  });
});
```

HTML Syntaxes (1)

HTML defines an abstract language for describing documents and two concrete syntaxes that can be used to transmit resources that use this abstract language.

HTML Syntaxes (2)

- **HTML syntax**:
  - While it bears a close resemblance to SGML and XML, it is a separate language with its own parsing rules.
  - It is compatible with most legacy web browsers.
  - File extension: `.html`, `.htm`
  - Media type: `text/html`
- **XML syntax**:
  - A syntax based on the XML 1.0 and the *Namespaces in XML 1.0* standards.
  - Does not define any syntax-level requirements beyond those defined for XML.
  - Also called as XHTML syntax.
  - File extension: `.xhtml`, `.xht`
  - Media type: `application/xhtml+xml`

HTML: the HTML Syntax (1)

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Sample Page</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

HTML: the HTML Syntax (2)

A document type declaration is required.

HTML: the HTML Syntax (3)

- Special characters:
  - Element text must not contain the `'<'` character or an ambiguous `'&'` character.
  - Attribute values must not contain an ambiguous `'&'` character.
- Ambiguous ampersand (`'&'`):
  - An `'&'` character that is followed by one or more ASCII alphanumerics, followed by a `';'`, where these characters do not match any of the named character references defined by the standard (e.g., `&nosuchchar;`).
    - See: Named character references

HTML: the HTML Syntax (4)

Element and attribute names are case-insensitive:

- Element and attribute names, even those for foreign elements, may be written with any mix of lower- and uppercase letters that are an ASCII case-insensitive match for the name of the element/attribute.
- There must never be two or more attributes on the same start tag whose names are an ASCII case-insensitive match for each other.

HTML: the HTML Syntax (5)

Unquoted attribute value syntax:

- If an attribute value other than the empty string does not contain any literal whitespace character it can be specified omitting the attribute value delimiters.
- For example, the following are equivalent:

```
<input value="yes">
<input value=yes>
```

HTML: the HTML Syntax (6)

Boolean attributes:

- A number of attributes are boolean attributes.
- The presence of a boolean attribute on an element represents the true value, and the absence of the attribute represents the false value.
- If the attribute is present, its value must either be the empty string or a value that is a case-insensitive match for the attribute's canonical name, with no leading or trailing white space.
- For example, the following are equivalent:

```
<input type=checkbox checked name=agree disabled>
<input type=checkbox checked=checked name=agree
  disabled=disabled>
<input type='checkbox' checked='' name="agree"
  disabled="">
```

HTML: the HTML Syntax (7)

Void elements:

- Only have a start tag, end tags must not be specified for them.
- Examples: `br`, `img`, `input`, `link`, `meta`, ...

HTML: the HTML Syntax (8)

- Foreign elements must either have a start tag and an end tag, or a start tag that is marked as self-closing, in which case they must not have an end tag.
- For example, the following SVG elements are equivalent:

```
<circle cx="50" cy="50" r="50"></circle>
<circle cx="50" cy="50" r="50"/>
```

HTML: the HTML Syntax (9)

Optional tags:

- The start and end tags of certain elements can be omitted.
- Omitting an element's start tag in the situations described here does not mean the element is not present (it is implied, but it is still there)!
  - For example, an HTML document always has a root `html` element, even if the string `<html>` doesn't appear anywhere in the markup.
- See: Optional tags

HTML: the HTML Syntax (10)

Optional tags (continued):

- An `li` element's end tag may be omitted if it is immediately followed by another `li` element or if there is no more content in the parent element.
  - For example, the following are equivalent:
    - ```
      <ul>
        <li>Apple</li>
        <li>Banana</li>
        <li>Cherry</li>
      </ul>
      ```
    - ```
      <ul>
        <li>Apple
        <li>Banana
        <li>Cherry
      </ul>
      ```

HTML: the HTML Syntax (11)

Optional tags (continued):

- An `html` element's start tag may be omitted if the first thing inside the element is not a comment.
- An `html` element's end tag may be omitted if the element is not immediately followed by a comment.
- ...

HTML: the HTML Syntax (12)

Optional tags (continued):

- If whitespace between elements is not significant, the following are equivalent:
  - ```
    <!DOCTYPE html>
    <html>
      <head>
        <title>Sample Page</title>
      </head>
      <body>
        Hello, World!
      </body>
    </html>
    ```
  - ```
    <!DOCTYPE html>
    <title>Sample Page</title>
    <p>Hello, World!</p>
    ```

HTML: the HTML Syntax (13)

- Namespace declarations are not supported, even in foreign elements.
- CDATA sections can only be used in foreign content (MathML or SVG).

HTML: the XML Syntax

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Page</title>
    <link rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

No DTD for HTML

- DTDs and XML schemas cannot express all the conformance requirements of HTML.
- See, for example, the custom `data-*` attributes.
  - See: Embedding custom non-visible data with the data-* attributes

HTML Document Type Declaration

- In the HTML syntax the document type declaration `<!DOCTYPE html>` is required, whose only purpose is to to ensure that the document is rendered in standards mode.
  - HTML generators that cannot output the short document type declaration above may use the document type declaration `<!DOCTYPE html SYSTEM "about:legacy-compat">` instead.
  - See: The DOCTYPE
- In the XML syntax any document type declaration can be used, but it is not required.
  - Documents transmitted with the `application/xhtml+xml` media type are always rendered in standards mode.
  - See: Writing documents in the XML syntax
- See also: https://www.w3.org/TR/html5-diff/#doctype

DOM (1)

- A DOM tree is an in-memory representation of a document.
- Relevant standard (WHATWG): DOM Living Standard

DOM (2)

- DOM is an API for accessing and manipulating documents (in particular, HTML and XML documents).
- DOM stands for **Document Object Model**.
- Each such document is represented as a tree that consists of nodes.
  - Major node types: `Document`, `DocumentType`, `Element`, `Text`, `ProcessingInstruction`, `Comment`

DOM (3)

Example:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Sample Page</title>
 </head>
 <body>
  <p>Hello, World!</p>
  <!-- This is a comment -->
 </body>
</html>
```

```
├ DOCTYPE: html
└ html lang="en"
  ├ head
  │ ├ #text: ⏎␣␣
  │ ├ title
  │ │ └ #text: Sample Page
  │ └ #text: ⏎␣
  ├ #text: ⏎␣
  └ body
    ├ #text: ⏎␣␣
    ├ p
    │ └ #text: Hello, World!
    ├ #text: ⏎␣␣
    ├ #comment: This is a comment
    └ #text: ⏎␣⏎
```

DOM (4)

- Each node is represented by an object with an API so that they can be manipulated.
- DOM interfaces are described in Web IDL.

DOM (5)

- Web IDL is an interface definition language that can be used to describe interfaces that are intended to be implemented in web browsers.
- Current standard (WHATWG): Web IDL Living Standard
- Example of use: `Node` interface

DOM (6)

- The HTML specification defines additional interfaces that extend DOM interfaces for representing HTML elements.
- Examples:
  - `meta` element
  - `img` element

DOM (7)

- The DOM is not just an API, the conformance criteria of HTML implementations are defined in terms of operations on the DOM.
- Example: The `Navigator` object
- The specifications are mostly phrased in terms of DOM trees, instead of markup.

DOM (8)

- A DOM tree can be manipulated from scripts in the page.
- Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>DOM Example</title>
  </head>
  <body>
    <p>User agent: <span id="ua"></span></p>
    <script>
      document.getElementById('ua').innerHTML = navigator.userAgent;
    </script>
  </body>
</html>
```

DOM (9)

Tools:

- Live DOM Viewer

Further information:

- MDN Web Docs – Document Object Model (DOM)
- The Modern JavaScript Tutorial – DOM tree

Working with the DOM (1)

```
// Setting values of CSS properties:
document.body.style.backgroundColor = 'aliceblue';
document.body.style.margin = '2rem';
document.getElementById('copyright').style.fontStyle = 'italic';

// Accessing the computed value of a CSS property:
const fontSize = window.getComputedStyle(document.body).fontSize;
console.log(`Computed font size of body: ${fontSize}`);

// Accessing classes of an element:
console.log(document.body.className);
document.getElementById('copyright').classList.add('important');
```

Working with the DOM (2)

Hiding and showing an element:

```
<aside>
  <div id="help">
    <!-- ... -->
  </div>
  <button onclick="toggleHelp()">Hide/Show Help</button>
</aside>
<script>
  function toggleHelp() {
    const element = document.getElementById('help');
    element.hidden = !element.hidden;
  }
</script>
```

However, note that onevent attributes are not recommended to be used!

Working with the DOM (3)

Hiding and showing an element:

```
<aside>
  <div id="help">
    <!-- ... -->
  </div>
  <button id="toggle-help-btn">Hide/Show Help</button>
</aside>
<script>
  function toggleHelp(event) {
    const element = document.getElementById('help');
    element.hidden = !element.hidden;
  }
  document.getElementById('toggle-help-btn')
      .addEventListener('click', toggleHelp);
</script>
```

Working with the DOM (4)

Creating elements dynamically:

```
<div id="container"></div>
<button id="add-item-btn">Add Item</button>
<script>
  function addItem(event) {
    const container = document.getElementById('container');
    const item = document.createElement('div');
    item.className = 'item';
    container.appendChild(item);
  }
  document.getElementById('add-item-btn')
    .addEventListener('click', addItem);
</script>
```

Working with the DOM (5)

Removing elements dynamically:

```
<div id="container"></div>
<button id="remove-children-btn">Remove Children</button>
<script>
  function removeChildren(event) {
    const container = document.getElementById('container');
    while (container.firstChild) {
      container.removeChild(container.firstChild);
    }
  }
  document.getElementById('remove-children-btn')
    .addEventListener('click', removeChildren);
</script>
```

HTML: DOM, HTML Syntax, and XML Syntax

The DOM, the HTML syntax, and the XML syntax cannot all represent the same content:

	HTML syntax	XML syntax	DOM
Namespaces	no	yes	yes
`noscript`	yes	no	no
The string `-->` in comments	no	no	yes

HTML APIs

See: MDN Web Docs – Web APIs

- Console API
- Fetch API
- Web Storage
- ...

Rendering Modes of Browsers (1)

Layout engines render HTML documents in the following modes:

- Quirks mode
- Standards mode
- Almost standards modes

Rendering Modes of Browsers (2)

- The historical reason for the existence of rendering modes is that early web standards were not compatible with the behavior of web browsers existing at that time.
- Web browsers introduced a new rendering mode to comply with web standards while still being able to render existing older existing content properly.
- Thus, modern web pages conforming to current web standards and obsolete web pages are rendered differently.

Rendering Modes of Browsers (3)

- **Quirks mode**: mimicking (emulating) the behavior of legacy web browsers in a way that violates current web standards for rendering obsolete web pages.
- **Standards mode**: rendering web pages according to current web standards.
- **Almost standards mode**: some web browsers also have a third rendering mode that differs from standards mode only in some height calculations that affects, e.g., the layout of images inside table cells.

Rendering Modes of Browsers (4)

- Earlier, browser engines used slightly different quirks modes, however, the WHATWG has standardized the quirks mode in the Quirks Mode specification.
- This specification does not enumerate all quirks that currently exist in browsers, a number of quirks are specified in other WHATWG specifications.

Rendering Modes of Browsers (5)

- The DOM specification renames standards mode and almost standards mode:
  - Renames standards mode to **no-quirks mode**.
  - Renames almost standards mode to **limited-quirks mode**.

Rendering Modes of Browsers (6)

Determining which rendering mode to use for a HTML document:

- For documents transmitted with the `text/html` media type the document type declaration determines the rendering mode.
- Documents transmitted with the `application/xhtml+xml` media type are always rendered in standards mode.
  - In this case the document is parsed with an XML parser that also checks the document for well-formedness.

Rendering Modes of Browsers (7)

How to determine which rendering mode is used?

- The `compatMode` attribute of the `Document` DOM interface returns the string `"BackCompat"` if the document's rendering mode is quirks, and `"CSS1Compat"` otherwise.
  - See: DOM – Living Standard – Interface Document
- On the browser UI:
  - Firefox: Firefox Page Info window (available from the Tools menu)

Rendering Modes of Browsers (8)

Further useful links:

- MDN Web Docs – Quirks Mode and Standards Mode
- Henri Sivonen. Activating Browser Modes with Doctype.

Error Handling (1)

- Unlike previous versions HTML, the current specification defines in some detail the required processing for invalid documents as well as valid documents.
- Actually, web browsers are very tolerant of errors, they automatically fix invalid content.
  - However, this primarily applies to documents in the HTML syntax!

Error Handling (2)

Example:

```
<html>
  <foo>
  <p bar class=>Does it <b>work,<i> or</b> not</i>?
  <
  </body>
</html>
```

Examine the DOM tree in either the browser DevTools or the Live DOM Viewer!

Error Handling (3)

Conformance checker (also known as a validator):

- A software tool that verifies whether an HTML document conforms to the conformance criteria of the HTML specification.
- Example:
  - Nu Html Checker (platform: Java; license: MIT License)
    https://validator.github.io/validator/
    https://github.com/validator/validator
    - Web interface: https://validator.w3.org/nu/

Error Handling (4)

See:

- HTML Living Standard – Parsing HTML documents
  - An introduction to error handling and strange cases in the parser
- Paul Irish, Tali Garsiel. How browsers work.
  - Browsers' error tolerance

Browser Support

- Supported by modern web browser.
- See:
  - *HTML5test – How well does your browser support HTML5?* (inactive)
    https://html5test.com/
    https://github.com/WebPlatformTest/HTML5test
    - Unofficial updated version: https://html5test.co/
      https://github.com/niutech/html5test
  - Can I use... Support tables for HTML5, CSS3, etc

Browser Developer Tools

- Chromium, Google Chrome, Opera: Chrome DevTools
- Firefox: Firefox DevTools User Docs
- Safari: Safari Developer Tools
- Chromium-based Edge: Microsoft Edge DevTools documentation

Developer Editions of Browsers

- Google Chrome: Google Chrome for developers
- Firefox: Firefox Browser Developer Edition

HTML Editors (1)

Free and open source software:

- Visual Studio Code (platform: Linux, macOS, Windows; license: MIT License) https://code.visualstudio.com/ https://github.com/Microsoft/vscode
  - See: https://code.visualstudio.com/docs/languages/html
  - Recommended extensions:
    - Live Preview https://marketplace.visualstudio.com/items?itemName=ms-vscode.live-server https://github.com/microsoft/vscode-livepreview

HTML Editors (2)

Emmet (written in: JavaScript; license: MIT License) https://emmet.io/ https://github.com/emmetio/emmet

- A set of text editor plugins for boosting HTML and CSS code writing.
- Available for many text editors such as: Eclipse, NetBeans, Notepad++, Visual Studio Code, IntelliJ IDEA, ...
  - See: https://emmet.io/download/
- Documentation: https://docs.emmet.io/
  - Customization: https://docs.emmet.io/customization/
- See also: Emmet in Visual Studio Code

HTML Editors (3)

Examples of Emmet abbreviations:

- `ul>li*3`

  is expanded to:

```
<ul>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

- `section.chapter>h1{Introduction}+p`

  is expanded to:

```
<section class="chapter">
  <h1>Introduction</h1>
  <p></p>
</section>
```

HTML Editors (4)

Examples of Emmet abbreviations (continued):

```
ul>li*5>a[href=#chapter$]{Chapter $}
```

is expanded to:

```
<ul>
  <li><a href="#chapter1">Chapter 1</a></li>
  <li><a href="#chapter2">Chapter 2</a></li>
  <li><a href="#chapter3">Chapter 3</a></li>
  <li><a href="#chapter4">Chapter 4</a></li>
  <li><a href="#chapter5">Chapter 5</a></li>
</ul>
```

HTML: Other Software (1)

Free and open source software:

- HTML5 Boilerplate (platform: browser; license: MIT License)
  https://html5boilerplate.com/
  https://github.com/h5bp/html5-boilerplate
- http-server (platform: Node.js; license: MIT License)
  https://www.npmjs.com/package/http-server
  https://github.com/http-party/http-server
- Tidy (platform: Linux, macOS, Windows; license: Tidy License)
  https://www.html-tidy.org/ https://github.com/htacg/tidy-html5

HTML: Other Software (2)

http-server: a simple static HTTP server

- Usage:

```
$ npx http-server -o index.html

$ npm install -g http-server
$ http-server -o index.html
```

- The document is available at http://localhost:8080/index.html.

HTML: Online Services

Surge: free static HTML publishing from the command line

- Website: https://surge.sh/
- Documentation: https://surge.sh/help/
- Installation and usage:

```
$ npm install --global surge
$ surge --domain example.surge.sh
```

- The deployed website is available at https://example.surge.sh/.

SVG (1)

- A language for describing two-dimensional vector graphics in XML.
- Also supports interactive graphics and animation.
- Developed by the W3C SVG Working Group.

SVG (2)

- Current standard: Scalable Vector Graphics (SVG) 1.1 (Second Edition) (W3C Recommendation, 16 August 2011)
- The forthcoming next version of the standard: Scalable Vector Graphics (SVG) 2 (W3C Candidate Recommendation, 4 October 2018)
- A profile of SVG for mobile devices: Scalable Vector Graphics (SVG) Tiny 1.2 Specification (W3C Recommendation, 22 December 2008)

SVG (3)

- SVG content can be embedded inline within other documents.
  - HTML supports direct embedding (see the `svg` element).
- Schema (DTD): https://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd
- File extension: `.svg`
- Media type: `image/svg+xml`

SVG (4)

Browser support:

- All modern web browsers support SVG natively.
- See: https://caniuse.com/svg

SVG (5)

Free and open source software: editors

- Inkscape (platform: Linux, macOS, Windows; license: GPLv2)
  https://inkscape.org/ https://gitlab.com/inkscape/inkscape
- macSVG (platform: macOS; license: MIT License)
  https://macsvg.org/ https://github.com/dsward2/macSVG
- SVG-Edit (written in: JavaScript; license: MIT License)
  https://github.com/SVG-Edit/svgedit
- tldraw (written in: TypeScript; license: Apache License 2.0)
  https://www.tldraw.com/ https://github.com/tldraw/tldraw

SVG (6)

Free and open source software: libraries

- Apache Batik (written in: Java; license: Apache License 2.0)
  https://xmlgraphics.apache.org/batik/
- Frappe Charts (written in: JavaScript; license: MIT License)
  https://frappe.io/charts https://github.com/frappe/charts
- Rough.js (written in: TypeScript; license: MIT License)
  https://roughjs.com/ https://github.com/rough-stuff/rough
- Snap.svg (written in: JavaScript; license: Apache License 2.0)
  http://snapsvg.io/
  https://github.com/adobe-webplatform/Snap.svg/
- SVG.js (written in: JavaScript; license: MIT License)
  https://svgjs.dev/ https://github.com/svgdotjs/svg.js

SVG (7)

Examples:

- Bootstrap Icons (license: MIT License)
  https://icons.getbootstrap.com/ https://github.com/twbs/icons/
- Feather – Simply beautiful open source icons (license: MIT License)
  https://feathericons.com/ https://github.com/feathericons/feather
- Inkscape Gallery https://inkscape.org/gallery/
- Super Tiny Icons (license: MIT License)
  https://github.com/edent/SuperTinyIcons
- Tabler Icons (license: MIT License) https://tabler.io/icons
  https://github.com/tabler/tabler-icons

SVG (8)

Further useful links:

- MDN Web Docs – SVG
  - SVG Tutorial

MathML (1)

- A language for describing mathematical notation in XML.
- Its goal is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text.
- Can be used for a number of purposes: for example, displaying mathematical content on the Web, exporting mathematical formulas from computer algebra systems, ...
- Developed by the W3C Math Working Group.

MathML (2)

- Current standard: Mathematical Markup Language (MathML) Version 3.0 2nd Edition (W3C Recommendation, 10 April 2014)
  - Also ratified as an ISO standard: ISO/IEC 40314:2016: Information technology – Mathematical Markup Language (MathML) Version 3.0 2nd Edition
- The next version of the standard currently under development is MathML 4: https://www.w3.org/TR/mathml4/

MathML (3)

- Can be used to describe both the presentation and the meaning of mathematical expressions.
  - Separate sets of elements elements serve the two distinct purposes (presentation elements, content elements).
- MathML content can be embedded within other documents.
  - HTML supports direct embedding (see the `math` element).

MathML (4)

- Schema (RELAX NG):
  - https://www.w3.org/Math/RelaxNG/mathml3/mathml3.rng
  - http://www.w3.org/Math/RelaxNG/mathml3/mathml3.rnc
- File extension: `.mml`
- Media type: `application/mathml+xml`, `application/mathml-presentation+xml`, `application/mathml-content+xml`

MathML (5)

Browser support:

- **Blink** (Chromium, Google Chrome, Opera, Chromium-based Edge): supported since Chromium version 109 released in January 2023.
  - See: https://www.igalia.com/2023/01/10/Igalia-Brings-MathML-Back-to-Chromium.html
- **Gecko** (Firefox): supported
- **WebKit** (Safari): supported

See: Can I use... Support tables for HTML5, CSS3, etc

MathML (6)

- Many other applications provides MathML support, such as Maple, Wolfram Mathematica, LibreOffice (MathML import), ...
- A collection of these can be found here: https://www.w3.org/wiki/Math_Tools

MathML (7)

Free and open source software:

- MathJax (written in: JavaScript; license: Apache License 2.0)
  https://www.mathjax.org/ https://github.com/mathjax/MathJax
  - Display engine written in JavaScript.

MathML (8)

Samples:

- https://developer.mozilla.org/en-US/docs/Web/MathML/Examples
- W3C MathML Test Suite

MathML (9)

Further useful links:

- MDN Web Docs – MathML
- Planet MathML (W3C)
- MathML in Web Browsers