# Web Applications Markup Language 1.0

## Working Draft 2 June 2004


**This version:**

http://www.whatwg.org/specs/web-apps/current-work/

**Latest version:**

http://www.whatwg.org/specs/web-apps/current-work/

**Editor:**

Ian Hickson, Opera Software, ian@hixie.ch



---

## Abstract

...

### Status of this document

**This is a work in progress!** This document is changing on a daily if not hourly basis in response to comments and as a general part of its development process. Comments are very welcome, please send them to whatwg@whatwg.org. Thank you.

It is very wrong to cite this as anything other than a work in progress. Do not implement this in a production product. It is not ready yet! At all!

This document is the result of a loose collaboration between interested parties in the context of the Web hypertext application technology working group. To become involved in the development of this document, please send comments to the address given above. **Your input will be taken into consideration.**

This is a working draft and may therefore be updated, replaced or rendered obsolete by other documents at any time. It is inappropriate to use Working Drafts as reference material or to cite them as other than "work in progress".

This draft may contain namespaces that use the `data:` URI scheme. These are temporary and will be changed before this specification is ready to be implemented.

To find the latest version of this working draft, please follow the "Latest version" link above.

## Table of contents



---

## 1. Introduction

The World Wide Web's markup language has always been HTML. HTML is a language designed primarily for semantically describing scientific documents, although its general design and adaptations over the years has enabled it to be used to describe a number of other types of documents.

The main area that has not been adequately addressed by HTML is a vague subject referred to as Web Applications. This specification attempts to address this.

The scope of this specification is not to describe an entire operating system. In particular, office productivity applications, image manipulation, and other applications that users would be expected to use on a daily basis are out of scope. This specification is targetted specifically at applications that would be expected to be used by users on an occasional basis. For instance online purchasing systems, searching systems, games (especially multiplayer online games), public telephone books or address books, etc.

For sophisticated cross-platform applications, there already exist several proprietary solutions (such as Mozilla's XUL and Macromedia's Flash). These solutions are evolving faster than any standards process could follow, and the requirements are evolving even faster. These systems are also significantly more complicated to specify, and are orders of magnitude more difficult to achieve interoperability with, than the solutions described in this document.

## 1.1. Requirements and ideas

HTML, CSS, DOM, and JavaScript provide enough power that Web developers have managed to base entire businesses on them. What is required are extensions to these technologies to provide much-needed features such as:

- Native pop-up menus and context menus.
- Inline markup for pop-up windows, for example for dialog boxes or tool palettes, so that dialogs need not be defined in separate files.
- Command updating: applications that have several access points for the same feature, for instance a menu item and a tool-bar button, would benefit from having to disable such commands only once, instead of having to keep each access point synchronized with the feature's availability at all times. Similarly menu items or tool-bar buttons that represent a toggle state could automatically stay synchronized whenever toggled.
- Server-sent events: triggering DOM3 Events from the server-side, for example for tickers or status updates.
- Client-server communications methods that do not require page loads, enabling on-demand data retrieval (where the UA automatically fetches data from the server as required), remote procedure calls (where script can invoke code on the server side and get an XML fragment in return), etc.
- More device-independent DOM events: The DOM event set needs device-independent events, such as events that fire when a button or link is activated, whether via the mouse or the keyboard. `DOMActivate` is a start,

but it lacks equivalent HTML attributes, and additional events may be needed.

- Sortable and multicolumn tree views and list views with rich formatting.
- Ability to define custom widgets cleanly, for example using XBL and APIs to query and control focus state, widget state, the position and state of input devices, etc.
- Rich text editing: an underlying architecture upon which domain-specific editors can be created, including things like control over the caret position.
- A predefined HTML editor based on the rich text editing architecture.
- Drag and drop APIs.
- Text selection manipulation APIs.
- Clipboard APIs (if the security and privacy concerns can be addressed).

Some less important features would be good to have as well:

- Window-based state management (so that new windows don't interfere with existing sessions), for example implemented as a per-domain, per-window "file system". This would allow multiple instances of the same application (from the same site) to run without the instances overwriting each other's cookies.
- Elements for semantics commonly found in applications, such as <byline>, <footer>, <section>, <navigation>, etc.
- Markup to denote mutually exclusive sections (as in the commonly seen wizard interfaces).
- Better defined user authentication state handling. (Being able to "log out" of sites reliably, for instance, or being able to integrate the HTTP authentication model into the Web page.)

Several of the features in these two lists have been supported in non-standard ways by some user agents for some time.

## 1.2. Relationship to HTML 4.01 and XHTML 1.1

This specification adds a number of features to HTML. For XML-based documents, these are added to the XHTML 1.x namespace. This ensures backwards compatibility with, and a simple migration path from, existing HTML and XHTML content.

## 1.3. Relationship to XHTML2

XHTML2 [XHTML2] updates HTML with better features for hyperlinks, multimedia content, annotating document edits, and introduces elements for better describing the semantics of human literary works such as poems.

Unfortunately, it lacks elements to express the semantics of many of the non-document types of content most often seen on the Web. For instance, the very popular forum sites, auction sites, search engines, online shops, and the like, do not fit the document metaphor well.

*This* specification aims to extend HTML so that it is more suitable in these contexts.

XHTML2 and this specification therefore address the needs of two different authoring scenarios, and are expected to be used side by side. XHTML2 is optimised for *documents* — for example help files, essays, articles — whereas this specification is optimised for applications.

## 1.4. Relationship to Web Forms 2.0

This specification is designed to complement Web Forms 2.0. [WF2] Where Web Forms concentrates on input controls, data validation, and form submission, this specification concentrates on client-side user interface features needed to create modern applications.

## 1.5. Relationship to CSS3 UI

The CSS3 UI specification [CSS3UI] introduces a number of properties suitable for Web-based application development. This specification expands on those properties and specifies their interaction with scripting-based environments and the DOM.

## 1.6. Conformance requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Diagrams, examples, and notes are non-normative. All other content in this specification is intended to be normative.

Documents that use the new features described in this specification using HTML over HTTP must be served as `text/html`. Documents that use the new features described in this specification using XHTML or other XML languages over HTTP must be served using an XML MIME type such as `application/xml`

or `application/xhtml+xml`. [RFC3023]

(In other words, for the purposes of [RFC2854], documents conforming to this specification using its HTML formulation should be considered HTML documents, but documents using the XML formulation should not be considered HTML-compatible.)

### 1.7. Terminology

...

## 2. Command updating [provisional title]

...

## 3. Focus

### 3.1. The `DocumentFocus` Interface

The `DocumentFocus` interface contains methods for moving focus around the document.

```
interface DocumentFocus {
  void moveFocusForward()
  void moveFocusBackward()
  void moveFocusUp()
  void moveFocusRight()
  void moveFocusDown()
  void moveFocusLeft()
};
```

The **`moveFocusForward`** method uses `'nav-index'` and `'user-can-focus'` properties to find the next focussable element and focusses it.

The **`moveFocusBackward`** method uses `'nav-index'` and `'user-can-focus'` properties to find the previous focussable element and focusses it.

The **`moveFocusUp`** method uses `'nav-up'` and `'user-can-focus'` properties to find an appropriate focussable element and focusses it.

In a similar manner, the **`moveFocusRight`**, **`moveFocusDown`**, and **`moveFocusLeft`** methods use the `'nav-right'`, `'nav-down'`, and `'nav-left'` properties (respectively), and the `'user-can-focus'` property, to find an appropriate focussable element and focus it.

## 4. Mutually exclusive sections

## References

XXX

## Acknowledgements

Thanks to Håkon Wium Lie, David Hyatt and Brendan Eich for their important support. Thanks to Matthew Mastracci and Steven Garrity for their useful and substantial comments.

Thanks also to the Microsoft blogging community for some ideas, to the attendees of the W3C Workshop on Web Applications and Compound Documents for inspiration, and to the #mozilla crew, the #opera crew, and the #mrt crew for their ideas and support.