HTML5 differences from HTML4

W3C Working Draft 25 October 2012

This Version:
http://www.w3.org/TR/2012/WD-html5-diff-20121025/
Latest Version:
http://www.w3.org/TR/html5-diff/
Latest Editor's Draft:
http://dev.w3.org/html5/html4-differences/
Previous Versions:
http://www.w3.org/TR/2012/WD-html5-diff-20120329/
http://www.w3.org/TR/2011/WD-html5-diff-20110525/
http://www.w3.org/TR/2011/WD-html5-diff-20110405/
http://www.w3.org/TR/2011/WD-html5-diff-20110405/
http://www.w3.org/TR/2011/WD-html5-diff-20110113/
http://www.w3.org/TR/2010/WD-html5-diff-20101019/
http://www.w3.org/TR/2010/WD-html5-diff-20100624/
http://www.w3.org/TR/2010/WD-html5-diff-20100304/
http://www.w3.org/TR/2009/WD-html5-diff-20090825/
http://www.w3.org/TR/2009/WD-html5-diff-20090423/
http://www.w3.org/TR/2009/WD-html5-diff-20090212/
http://www.w3.org/TR/2008/WD-html5-diff-20080610/
http://www.w3.org/TR/2008/WD-html5-diff-20080122/
Editors:
Anne van Kesteren (Opera Software ASA) <annevk@annevk.nl>
Simon Pieters (Opera Software ASA) <simonp@opera.com>

Abstract

HTML is the core language of the World Wide Web. The W3C publishes HTML5, which is the fifth major revision of HTML. The WHATWG publishes HTML, which is a rough superset of HTML5. "HTML5 differences from HTML4" describes the differences of these documents from HTML4, and calls out cases where HTML is different from HTML5. This document may not provide accurate information as the specifications are still actively in development. When in doubt, always check the specifications themselves. [HTML5] [HTML]

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is the 25 October 2012 W3C Working Draft produced by the HTML Working Group, part of the HTML Activity. The Working Group intends to publish this document as a Working Group Note to accompany the HTML5 specification. The appropriate forum for comments is W3C Bugzilla. (public-html-comments@w3.org, a mailing list with a public archive, is no longer used for tracking comments.)

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

1 Introduction

HTML has been in continuous evolution since it was introduced to the Internet in the early 1990s. Some features were introduced in specifications; others were introduced in software releases. In some respects, implementations and author practices have converged with each other and with specifications and standards, but in other ways, they have diverged.

HTML4 became a W3C Recommendation in 1997. While it continues to serve as a rough guide to many of the core features of HTML, it does not provide enough information to build implementations that interoperate with each other and, more importantly, with a critical mass of deployed content. The same goes for XHTML1, which defines an XML serialization for HTML4, and DOM Level 2 HTML, which defines JavaScript APIs for both HTML and XHTML. HTML5 will replace these documents. [DOM2HTML] [HTML4] [XHTML1]

The HTML5 draft reflects an effort, started in 2004, to study contemporary HTML implementations and deployed content. The draft:

1. Defines a single language called HTML which can be written in HTML syntax and in XML syntax.
2. Defines detailed processing models to foster interoperable implementations.
3. Improves markup for documents.
4. Introduces markup and APIs for emerging idioms, such as Web applications.

1.1 Open Issues

HTML5 is still a draft. The contents of HTML5, as well as the contents of this document which depend on HTML5, are still being discussed on the HTML Working Group and WHATWG mailing lists. The open issues are linked from the HTML5 draft.

1.2 Backwards Compatible

HTML5 is defined in a way that it is backwards compatible with the way user agents handle deployed content. To keep the authoring language relatively simple for authors, several elements and attributes are not included, as outlined in the other sections of this document, such as presentational elements that are better dealt with using CSS.

User agents, however, will always have to support these older elements and attributes and this is why the HTML5 specification clearly separates requirements for authors and user agents. For instance, this means that authors cannot use the `isindex` or the `plaintext` element, but user agents are required to support them in a way that is compatible with how these elements need to behave for compatibility with deployed content.

Since HTML5 has separate conformance requirements for authors and user agents there is no longer a need for marking features "deprecated".

1.3 Development Model

The HTML5 specification will not be considered finished before there are at least two complete implementations of the specification. A test suite will be used to measure completeness of the implementations. This approach differs from previous versions of HTML, where the final

specification would typically be approved by a committee before being actually implemented. The goal of this change is to ensure that the specification is implementable, and usable by authors once it is finished.

2 Syntax

HTML5 defines an HTML syntax that is compatible with HTML4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML4, such as processing instructions and shorthand markup as these are not supported by most user agents. Documents using the HTML syntax are served with the `text/html` media type.

HTML5 also defines detailed parsing rules (including "error handling") for this syntax which are largely compatible with HTML4-era implementations. User agents must use these rules for resources that have the `text/html` media type. Here is an example document that conforms to the HTML syntax:

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Example document</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

The other syntax that can be used for HTML5 is XML. This syntax is compatible with XHTML1 documents and implementations. Documents using this syntax need to be served with an XML media type and elements need to be put in the `http://www.w3.org/1999/xhtml` namespace following the rules set forth by the XML specifications. [XML]

Below is an example document that conforms to the XML syntax of HTML5. Note that XML documents must be served with an XML media type such as `application/xhtml+xml` or `application/xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Example document</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

2.1 Character Encoding

For the HTML syntax, authors are required to declare the character encoding. There are three ways to do that:

- At the transport level. By using the HTTP `Content-Type` header for instance.
- Using a Unicode Byte Order Mark (BOM) character at the start of the file. This character provides a signature for the encoding used.
- Using a `meta` element with a `charset` attribute that specifies the encoding within the first 1024 bytes of the document. For instance, `<meta charset="UTF-8">` could be used to specify the UTF-8 encoding. This replaces the need for `<meta http-`

`equiv="Content-Type" content="text/html; charset=UTF-8">` although that syntax is still allowed.

For the XML syntax, authors have to use the rules as set forth in the XML specifications to set the character encoding.

2.2 The Doctype

The HTML syntax of HTML5 requires a doctype to be specified to ensure that the browser renders the page in standards mode. The doctype has no other purpose. [DOCTYPE]

The doctype declaration for the HTML syntax is `<!DOCTYPE html>` and is case-insensitive. Doctypes from earlier versions of HTML were longer because the HTML language was SGML-based and therefore required a reference to a DTD. With HTML5 this is no longer the case and the doctype is only needed to enable standards mode for documents written using the HTML syntax. Browsers already do this for `<!DOCTYPE html>`.

To support legacy markup generators that cannot generate the preferred short doctype, the doctype `<!DOCTYPE html SYSTEM "about:legacy-compat">` is allowed in the HTML syntax.

The strict doctypes for HTML 4.0, HTML 4.01, XHTML 1.0 as well as XHTML 1.1 are also allowed (but are discouraged) in the HTML syntax.

In the XML syntax, any doctype declaration may be used, or it may be omitted altogether. Documents with an XML media type are always handled in standards mode.

2.3 MathML and SVG

The HTML syntax of HTML5 allows for MathML and SVG elements to be used inside a document. An `math` or `svg` start tag causes the HTML parser to switch to a special insertion mode which puts elements and attributes in the appropriate namespaces, does case fixups for elements and attributes that have mixed case, and supports the empty-element syntax as in XML. The syntax is still case-insensitive and attributes allow the same syntax as for HTML elements. Namespace declarations may be omitted. CDATA sections are supported in this insertion mode.

Some MathML and SVG elements cause the parser to switch back to "HTML mode", e.g. `mtext` and `foreignObject`, so you can use HTML elements or a new `math` or `svg` element.

For instance, a very simple document using some of the minimal syntax features could look like:

```
<!doctype html>
<title>SVG in text/html</title>
<p>
 A green circle:
 <svg> <circle r="50" cx="50" cy="50" fill="green"/> </svg>
</p>
```

2.4 Miscellaneous

There are a few other changes in the HTML syntax worthy of mentioning:

- The `&lang;` and `&rang;` named character references now expand to U+27E8 and U+27E9 instead of U+2329 and U+232A, respectively.
- Many new named character references have been added, including all from MathML.
- Void elements (known as "EMPTY" in HTML4) are allowed to have a trailing slash.
- The ampersand (&) may be left unescaped in some more cases compared to HTML4.

- Attributes have to be separated by at least one whitespace character.
- Attributes with an empty value may be written as just the attribute name omitting the equals sign and the value, even if it's not a boolean attribute. (HTML4 actually allowed using only the attribute *value* and omitting the attribute name, for enumerated attributes, but this was not supported in browsers.)
- Attributes omitting quotes for the value are allowed to use a larger set of characters compared to HTML4.
- The `optgroup` end tag is now optional.
- The `colgroup` start tag is now optional and is inferred by the HTML parser.

3 Language

This section is split up in several subsections to more clearly illustrate the various differences there are between HTML4 and HTML5.

3.1 New Elements

The following elements have been introduced for better structure:

- `section` represents a generic document or application section. It can be used together with the `h1`, `h2`, `h3`, `h4`, `h5`, and `h6` elements to indicate the document structure.
- `article` represents an independent piece of content of a document, such as a blog entry or newspaper article.
- `aside` represents a piece of content that is only slightly related to the rest of the page.
- `hgroup` represents the header of a section.
- `header` represents a group of introductory or navigational aids.
- `footer` represents a footer for a section and can contain information about the author, copyright information, etc.
- `nav` represents a section of the document intended for navigation.
- `figure` represents a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.

  ```
  <figure>
   <video src="example.webm" controls></video>
   <figcaption>Example</figcaption>
  </figure>
  ```

  `figcaption` can be used as caption (it is optional).

Then there are several other new elements:

- `video` and `audio` for multimedia content. Both provide an API so application authors can script their own user interface, but there is also a way to trigger a user interface provided by the user agent. `source` elements are used together with these elements if there are multiple streams available of different types.
- `track` provides text tracks for the `video` element.
- `embed` is used for plugin content.

- `mark` represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
- `progress` represents a completion of a task, such as downloading or when performing a series of expensive operations.
- `meter` represents a measurement, such as disk usage.
- `time` represents a date and/or time.
- WHATWG HTML has `data` which allows content to be annotated with a machine-readable value.
- WHATWG HTML has `dialog` for showing a dialog.
- `ruby`, `rt`, and `rp` allow for marking up ruby annotations.
- `bdi` represents a span of text that is to be isolated from its surroundings for the purposes of bidirectional text formatting.
- `wbr` represents a line break opportunity.
- `canvas` is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
- `command` represents a command the user can invoke.
- `details` represents additional information or controls which the user can obtain on demand. The `summary` element provides its summary, legend, or caption.
- `datalist` together with the a new `list` attribute for `input` can be used to make comboboxes:

```
<input list="browsers">
<datalist id="browsers">
 <option value="Safari">
 <option value="Internet Explorer">
 <option value="Opera">
 <option value="Firefox">
</datalist>
```

- `keygen` represents control for key pair generation.
- `output` represents some type of output, such as from a calculation done through scripting.

The `input` element's `type` attribute now has the following new values:

- `tel`
- `search`
- `url`
- `email`
- `datetime`
- `date`
- `month`
- `week`
- `time`
- `datetime-local`
- `number`
- `range`

- color

The idea of these new types is that the user agent can provide the user interface, such as a calendar date picker or integration with the user's address book, and submit a defined format to the server. It gives the user a better experience as his input is checked before sending it to the server meaning there is less time to wait for feedback.

3.2 New Attributes

Several attributes have been introduced to various elements that were already part of HTML4:

- The a and area elements now have a media attribute for consistency with the link element. WHATWG HTML also has the download and ping attributes.
- The area element, for consistency with the a and link elements, now also has the hreflang, type and rel attributes.
- The base element can now have a target attribute as well, mainly for consistency with the a element. (This is already widely supported.)
- The meta element has a charset attribute now as this was already widely supported and provides a nice way to specify the character encoding for the document.
- A new autofocus attribute can be specified on the input (except when the type attribute is hidden), select, textarea and button elements. It provides a declarative way to focus a form control during page load. Using this feature should enhance the user experience compared to focusing the element with script as the user can turn it off if the user does not like it, for instance.
- A new placeholder attribute can be specified on the input and textarea elements. It represents a hint intended to aid the user with data entry.

  ```
  <input type=email placeholder="a@b.com">
  ```

- The new form attribute for input, output, select, textarea, button, label, object and fieldset elements allows for controls to be associated with a form. These elements can now be placed anywhere on a page, not just as descendants of the form element, and still be associated with a form.

  ```
  <table>
   <tr>
    <th>Key
    <th>Value
    <th>Action
   <tr>
    <td><form id=1><input name=1-key></form>
    <td><input form=1 name=1-value>
    <td><button form=1 name=1-action value=save>✓</button>
        <button form=1 name=1-action value=delete>✗</button>
   ...
  </table>
  ```

- The new required attribute applies to input (except when the type attribute is hidden, image or some button type such as submit), select and textarea. It indicates that the user has to fill in a value in order to submit the form. For select, the first option element has to be a placeholder with an empty value.

  ```
  <label>Color: <select name=color required>
   <option value="">Choose one
  ```

```
 <option>Red
 <option>Green
 <option>Blue
</select></label>
```

- The `fieldset` element now allows the `disabled` attribute which disables all descendant controls (excluding those that are descendants of the `legend` element) when specified, and the `name` attribute which can be used for script access.
- The `input` element has several new attributes to specify constraints: `autocomplete`, `min`, `max`, `multiple`, `pattern` and `step`. As mentioned before it also has a new `list` attribute which can be used together with the `datalist` element. It also now has the `width` and `height` attributes to specify the dimensions of the image when using `type=image`.
- The `input` and `textarea` elements have a new attribute named `dirname` that causes the directionality of the control as set by the user to be submitted as well.
- The `textarea` element also has two new attributes, `maxlength` and `wrap` which control max input length and submitted line wrapping behavior, respectively.
- The `form` element has a `novalidate` attribute that can be used to disable form validation submission (i.e. the form can always be submitted).
- The `input` and `button` elements have `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget` as new attributes. If present, they override the `action`, `enctype`, `method`, `novalidate`, and `target` attributes on the `form` element.
- In WHATWG HTML, the `input` and `textarea` have an `inputmode` attribute.
- The `menu` element has two new attributes: `type` and `label`. They allow the element to transform into a menu as found in typical user interfaces as well as providing for context menus in conjunction with the global `contextmenu` attribute.
- The `style` element has a new `scoped` attribute which can be used to enable scoped style sheets. Style rules within such a `style` element only apply to the local tree.
- The `script` element has a new attribute called `async` that influences script loading and execution.
- The `html` element has a new attribute called `manifest` that points to an application cache manifest used in conjunction with the API for offline Web applications.
- The `link` element has a new attribute called `sizes`. It can be used in conjunction with the `icon` relationship (set through the `rel` attribute; can be used for e.g. favicons) to indicate the size of the referenced icon. Thus allowing for icons of distinct dimensions.
- The `ol` element has a new attribute called `reversed`. When present, it indicates that the list order is descending.
- The `iframe` element has three new attributes called `sandbox`, `seamless`, and `srcdoc` which allow for sandboxing content, e.g. blog comments.
- The `object` element has a new attribute called `typemustmatch` which allows safer embedding of external resources.
- The `img` element has a new attribute called `crossorigin` to use CORS in the fetch and if

it is successful, allows the image data to be read with the `canvas` API. In WHATWG HTML, there is also a new attribute called `srcset` to support multiple images for different resolutions and different images for different viewport sizes.

Several attributes from HTML4 now apply to all elements. These are called global attributes: `accesskey`, `class`, `dir`, `id`, `lang`, `style`, `tabindex` and `title`. Additionally, XHTML 1.0 only allowed `xml:space` on some elements, which is now allowed on all elements in XHTML documents.

There are also several new global attributes:

- The `contenteditable` attribute indicates that the element is an editable area. The user can change the contents of the element and manipulate the markup.
- The `contextmenu` attribute can be used to point to a context menu provided by the author.
- The `data-*` collection of author-defined attributes. Authors can define any attribute they want as long as they prefix it with `data-` to avoid clashes with future versions of HTML. These are intended to be used to store custom data to be consumed by the Web page or application itself. They are *not* intended for data to be consumed by other parties (e.g. user agents).
- The `draggable` and `dropzone` attributes can be used together with the new drag & drop API.
- The `hidden` attribute indicates that an element is not yet, or is no longer, relevant.
- WHATWG HTML has the `inert` attribute, intended to make `dialog` elements modal.
- The `role` and `aria-*` collection attributes which can be used to instruct assistive technology.
- The `spellcheck` attribute allows for hinting whether content can be checked for spelling or not.
- The `translate` attribute gives a hint to translators whether the content should be translated.

HTML5 also makes all event handler attributes from HTML4, which take the form `on`*`event`*, global attributes and adds several new event handler attributes for new events it defines. For instance, the `onplay` event handler attribute for the `play` event which is used by the API for the media elements (`video` and `audio`).

3.3 Changed Elements

These elements have slightly modified meanings in HTML5 to better reflect how they are used on the Web or to make them more useful:

- The `address` element is now scoped by the nearest ancestor `article` or `body` element.
- The `b` element now represents a span of text to which attention is being drawn for utilitarian purposes without conveying any extra importance and with no implication of an alternate voice or mood, such as key words in a document abstract, product names in a review, actionable words in interactive text-driven software, or an article lede.
- The `cite` element now solely represents the title of a work (e.g. a book, a paper, an essay, a poem, a score, a song, a script, a film, a TV show, a game, a sculpture, a painting, a theatre production, a play, an opera, a musical, an exhibition, a legal case report, etc). Specifically

the example in HTML4 where it is used to mark up the name of a person is no longer considered conforming.

- The `dl` element now represents an association list of name-value groups, and is no longer said to be appropriate for dialogue.
- The `hr` element now represents a paragraph-level thematic break.
- The `i` element now represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose in a manner indicating a different quality of text, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, a thought, or a ship name in Western texts.
- For the `label` element the browser should no longer move focus from the label to the control unless such behavior is standard for the underlying platform user interface.
- The `menu` element is redefined to be useful for toolbars and context menus.
- The `noscript` element is no longer said to be rendered when the user agent doesn't support a scripting language invoked by a `script` element earlier in the document.
- The `s` element now represents contents that are no longer accurate or no longer relevant.
- The `script` element can now be used for scripts or for custom data blocks.
- The `small` element now represents side comments such as small print.
- The `strong` element now represents importance rather than strong emphasis.
- The `u` element now represents a span of text with an unarticulated, though explicitly rendered, non-textual annotation, such as labeling the text as being a proper name in Chinese text (a Chinese proper name mark), or labeling the text as being misspelt.

3.4 Changed Attributes

Several attributes have changed in various ways.

- The `accept` attribute on `input` now allows the values `audio/*`, `video/*` and `image/*`.
- The `accesskey` global attribute now allows multiple characters to be specified, which the user agent can choose from.
- The `action` attribute on `form` is no longer allowed to have an empty URL.
- In WHATWG HTML, the `method` attribute has a new keyword `dialog`, intended to close a `dialog` element.
- The `border` attribute on `table` only allows the values "1" and the empty string. In WHATWG HTML, the `border` attribute is obsolete.
- The `colspan` attribute on `td` and `th` now has to be greater than zero.
- The `coords` attribute on `area` no longer allows a percentage value of the radius when the element is in the circle state.
- The `data` attribute on `object` is no longer said to be relative to the `codebase` attribute.
- The `defer` attribute on `script` now explicitly makes the script execute when the page has finished parsing.

- The `dir` global attribute now allows the value `auto`.
- The `enctype` attribute on `form` now supports the value `text/plain`.
- The `width` and `height` attributes on `img`, `iframe` and `object` are no longer allowed to contain percentages. They are also not allowed to be used to stretch the image to a different aspect ratio than its intrinsic aspect ratio.
- The `href` attribute on `link` is no longer allowed to have an empty URL.
- The `href` attribute on `base` is now allowed to contain a relative URL.
- All attributes that take URLs, e.g. `href` on the `a` element, now support IRIs if the document's encoding is UTF-8 or UTF-16.
- The `http-equiv` attribute on `meta` is no longer said to be used by HTTP servers to create HTTP headers in the HTTP response. Instead, it is said to be a pragma directive to be used by the user agent.
- The `id` global attribute is now allowed to have any value, as long as it is unique, is not the empty string, and does not contain space characters.
- The `lang` global attribute takes the empty string in addition to a valid language identifier, just like `xml:lang` does in XML.
- The `media` attribute on `link` now accepts a media query and defaults to "all".
- The event handler attributes (e.g. `onclick`) now always use JavaScript as the scripting language.
- The `value` attribute of the `li` element is no longer deprecated as it is not presentational. The same goes for the `start` and `type` attributes of the `ol` element.
- The `style` global attribute now always uses CSS as the styling language.
- The `tabindex` global attribute now allows negative values which indicate that the element can receive focus but cannot be tabbed to.
- The `target` attribute of the `a` and `area` elements is no longer deprecated, as it is useful in Web applications, e.g. in conjunction with `iframe`.
- The `type` attribute on `script` and `style` is no longer required if the scripting language is JavaScript and the styling language is CSS, respectively.
- The `usemap` attribute on `img` no longer takes a URL, but instead takes a valid hash-name reference to a `map` element.

The following attributes are allowed but authors are discouraged from using them and instead strongly encouraged to use an alternative solution:

- The `border` attribute on `img`. It is required to have the value "`0`" when present. Authors can use CSS instead.
- The `language` attribute on `script`. It is required to have the value "`JavaScript`" (case-insensitive) when present and cannot conflict with the `type` attribute. Authors can simply omit it as it has no useful function.
- The `name` attribute on `a`. Authors can use the `id` attribute instead.

3.5 Obsolete Elements

The elements in this section are not to be used by authors. User agents will still have to support them and various sections in HTML5 define how. E.g. the obsolete `isindex` element is handled by the parser section.

The following elements are not in HTML5 because their effect is purely presentational and their function is better handled by CSS:

- `basefont`
- `big`
- `center`
- `font`
- `strike`
- `tt`

The following elements are not in HTML5 because using them damages usability and accessibility:

- `frame`
- `frameset`
- `noframes`

The following elements are not included because they have not been used often, created confusion, or their function can be handled by other elements:

- `acronym` is not included because it has created a lot of confusion. Authors are to use `abbr` for abbreviations.
- `applet` has been obsoleted in favor of `object`.
- `isindex` usage can be replaced by usage of form controls.
- `dir` has been obsoleted in favor of `ul`.

Finally the `noscript` element is only conforming in the HTML syntax. It is not allowed in the XML syntax. This is because in order to not only hide visually but also prevent the content to run scripts, apply style sheets, have submittable form controls, load resources, and so forth, the HTML parser parses the content of the `noscript` element as plain text. The same is not possible with an XML parser.

3.6 Obsolete Attributes

Some attributes from HTML4 are no longer allowed in HTML5. The specification defines how user agents should process them in legacy documents, but authors must not use them and they will not validate.

HTML5 has advice on what you can use instead.

- `rev` and `charset` attributes on `link` and `a`.
- `shape` and `coords` attributes on `a`.
- `longdesc` attribute on `img` and `iframe`.
- `target` attribute on `link`.
- `nohref` attribute on `area`.
- `profile` attribute on `head`.
- `version` attribute on `html`.
- `name` attribute on `img` (use `id` instead).
- `scheme` attribute on `meta`.

- `archive`, `classid`, `codebase`, `codetype`, `declare` and `standby` attributes on `object`.
- `valuetype` and `type` attributes on `param`.
- `axis` and `abbr` attributes on `td` and `th`.
- `scope` attribute on `td`.
- `summary` attribute on `table`.

In addition, HTML5 has none of the presentational attributes that were in HTML4 as their functions are better handled by CSS:

- `align` attribute on `caption`, `iframe`, `img`, `input`, `object`, `legend`, `table`, `hr`, `div`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `p`, `col`, `colgroup`, `tbody`, `td`, `tfoot`, `th`, `thead` and `tr`.
- `alink`, `link`, `text` and `vlink` attributes on `body`.
- `background` attribute on `body`.
- `bgcolor` attribute on `table`, `tr`, `td`, `th` and `body`.
- `border` attribute on `object`.
- `cellpadding` and `cellspacing` attributes on `table`.
- `char` and `charoff` attributes on `col`, `colgroup`, `tbody`, `td`, `tfoot`, `th`, `thead` and `tr`.
- `clear` attribute on `br`.
- `compact` attribute on `dl`, `menu`, `ol` and `ul`.
- `frame` attribute on `table`.
- `frameborder` attribute on `iframe`.
- `height` attribute on `td` and `th`.
- `hspace` and `vspace` attributes on `img` and `object`.
- `marginheight` and `marginwidth` attributes on `iframe`.
- `noshade` attribute on `hr`.
- `nowrap` attribute on `td` and `th`.
- `rules` attribute on `table`.
- `scrolling` attribute on `iframe`.
- `size` attribute on `hr`.
- `type` attribute on `li`, and `ul`.
- `valign` attribute on `col`, `colgroup`, `tbody`, `td`, `tfoot`, `th`, `thead` and `tr`.
- `width` attribute on `hr`, `table`, `td`, `th`, `col`, `colgroup` and `pre`.

4 Content Model

Content model is what defines how elements may be nested — what is allowed as children (or descendants) of a certain element.

At a high level, HTML4 had two major categories of elements, "inline" (e.g. `span`, `img`, text), and "block-level" (e.g. `div`, `hr`, `table`). Some elements did not fit in either category.

Some elements allowed "inline" elements (e.g. `p`), some allowed "block-level" elements (e.g. `body`), some allowed both (e.g. `div`), while other elements did not allow either category but only allowed other specific elements (e.g. `dl`, `table`), or did now allow any children at all (e.g. `link`, `img`, `hr`).

Notice the difference between an element itself being in a certain category, and having a content

model of a certain category. For instance, the `p` element is itself a "block-level" element, but has a content model of "inline".

To make it more confusing, HTML4 had different content model rules in its Strict, Transitional and Frameset flavors. For instance, in Strict, the `body` element allowed only "block-level" elements, but in Transitional, it allowed both "inline" and "block-level".

To make things more confusing still, CSS uses the terms "block-level element" and "inline-level element" for its visual formatting model, which is related to CSS's 'display' property and has nothing to do with HTML's content model rules.

HTML5 does not use the terms "block-level" or "inline" as part of its content model rules, to reduce confusion with CSS. However, it has more categories than HTML4, and an element can be part of none of them, one of them, or several of them.

- Metadata content, e.g. `link`, `script`.
- Flow content, e.g. `span`, `div`, text. This is roughly like HTML4's "block-level" and "inline" together.
- Sectioning content, e.g. `aside`, `section`.
- Heading content, e.g. `h1`, `hgroup`.
- Phrasing content, e.g. `span`, `img`, text. This is roughly like HTML4's "inline". Elements that are phrasing content are also flow content.
- Embedded content, e.g. `img`, `iframe`, `svg`.
- Interactive content, e.g. `a`, `button`, `label`. Interactive content is not allowed to be nested.

As broad changes from HTML4, HTML5 no longer has any element that only accepts what HTML4 called "block-level" elements; e.g. the `body` element now allows flow content. This is thus closer to HTML4 Transitional than HTML4 Strict.

Further changes include:

- The `address` element now allows flow content, but with no heading content descendants, no sectioning content descendants, and no `header`, `footer`, or `address` element descendants.
- HTML4 allowed `object` in `head`. HTML5 does not.
- WHATWG HTML allows `link` and `meta` as descendants of `body` if they use microdata attributes.
- The `noscript` element was a "block-level" element in HTML4, but is phrasing content in HTML5.
- The `table`, `thead`, `tbody`, `tfoot`, `tr`, `ol`, `ul` and `dl` elements are allowed to be empty in HTML5.
- Table elements have to conform to the table model (e.g. two cells are not allowed to overlap).
- The `table` element now does not allow `col` elements as direct children. However, the HTML parser implies a `colgroup` element, so this change should not affect `text/html` content.
- The `table` element now allows the `tfoot` element to be the last child.
- The `caption` element now allows flow content, but with no descendant `table` elements.
- The `th` element now allows flow content, but with no `header`, `footer`, sectioning

content, or heading content descendants.

- The a element now has a transparent content model (except it does not allow interactive content descendants), meaning that it has the same content model as its parent. This means that the a element can now contain e.g. div elements, if its parent allows flow content.
- The ins and del elements also have a transparent content model. HTML4 had similar rules in prose that could not be expressed in the DTD.
- The object element also has a transparent content model, after its param children.
- The map element also has a transparent content model. The area element is considered phrasing content if there is a map element ancestor, which means that they do not need to be direct children of map.
- The fieldset element no longer requires a legend child.

5 APIs

HTML5 has introduced many new APIs and have extended, changed or obsoleted some existing APIs.

5.1 New APIs

HTML5 introduces a number of APIs that help in creating Web applications. These can be used together with the new elements introduced for applications:

- Media elements (video and audio) have APIs for controlling playback, syncronising multiple media elements, and timed text tracks (e.g. subtitles).
- An API for form constraint validation (e.g. the setCustomValidity() method).
- An API for commands that the user can invoke (used together with the command element among others).
- An API that enables offline Web applications, with an application cache.
- An API that allows a Web application to register itself for certain protocols or media types, using registerProtocolHandler() and registerContentHandler() .
- Editing API in combination with a new global contenteditable attribute.
- Drag & drop API in combination with a draggable attribute.
- An API that exposes the components of the document's URL and allows scripts to navigate, redirect and reload (the Location interface).
- An API that exposes the session history and allows scripts to update the document's URL without actually navigating, so that applications don't need to abuse the fragment component for "Ajax-style" navigation (the History interface).
- An API for base64 conversion (atob() and btoa() methods).
- An API to schedule timer-based callbacks (setTimeout() and setInterval()).
- An API to prompt the user (alert(), confirm(), prompt(), showModalDialog()).
- An API for printing the document (print()).

- An API for handling search providers (`AddSearchProvider()` and `IsSearchProviderInstalled()`).
- The `Window` object has been defined.

WHATWG HTML has further APIs that are not in HTML5 but are separate specifications at the W3C:

- An API for microdata.
- An API for immediate-mode bitmap graphics (the `2d` context for the `canvas` element).
- An API for cross-document messaging and channel messaging (`postMessage()` and `MessageChannel`).
- An API for runnings scripts in the background (`Worker` and `SharedWorker`).
- An API for client-side storage (`localStorage` and `sessionStorage`).
- An API for bidirectional client-server communication (`WebSocket`).
- An API for server-to-client data push (`EventSource`).

5.2 Changed APIs

The following features from DOM Level 2 HTML are changed in various ways:

- `document.title` now collapses whitespace on getting.
- `document.domain` is made settable, which can change the document's effective script origin.
- `document.open()` now either clears the document (if invoked with two or less arguments), or acts like `window.open()` (if invoked with three or four arguments). In the former case, throws an exception in XML.
- `document.close()`, `document.write()` and `document.writeln()` throw an exception in XML. The latter two now support variadic arguments; they can add text to the document's input stream while it is still being parsed, or can imply a call to `document.open()` or be ignored altogether in some cases.
- `document.getElementsByName()` now returns all HTML elements with a `name` attribute matching the argument.
- `elements` on `HTMLFormElement` now returns an `HTMLFormControlsCollection` of `button`, `fieldset`, `input`, `keygen`, `object`, `output`, `select` and `textarea` elements. `length` returns the number of nodes in `elements`.
- `add()` on `HTMLSelectElement` now also accepts an integer as its second argument.
- `remove()` on `HTMLSelectElement` now removes the first element in the collection if the argument is out of bounds.
- `a` and `area` elements now stringify to their `href` attribute.
- The `click()`, `focus()` and `blur()` methods are now available on all HTML elements.

5.3 Extensions to `Document`

DOM Level 2 HTML had an `HTMLDocument` interface that inherited from `Document` and provided HTML-specific members on documents. HTML5 has moved these members to the `Document` interface, and extended it in a number of ways. Since all documents use the `Document` interface, the HTML-specific members are now available on all documents, so they are usable in e.g. SVG documents as well. It also has several new members:

- `location`, `lastModified` and `readyState` to help resource metadata management.
- `dir`, `head`, `embeds`, `plugins`, `scripts`, `commands`, and a generic name getter, to access various parts of the DOM tree. WHATWG HTML also has `getItems()` for microdata and `cssElementMap` to accompany the CSS `element()` feature.
- `activeElement` and `hasFocus` to determine which element is currently focused and whether the `Document` has focus respectively.
- `designMode`, `execCommand()`, `queryCommandEnabled()`, `queryCommandIndeterm()`, `queryCommandState()`, `queryCommandSupported()`, `queryCommandValue()` for the editing API.
- All event handler IDL attributes. Also, `onreadystatechange` is a special event handler IDL attribute that is only available on `Document`.

Existing scripts that modified the prototype of `HTMLDocument` should continue to work because `window.HTMLDocument` now returns the `Document` interface object.

5.4 Extensions to `HTMLElement`

The `HTMLElement` interface has also gained several extensions in HTML5:

- `translate`, `hidden`, `tabIndex`, `accessKey`, `draggable`, `dropzone`, `contentEditable`, `contextMenu`, `spellcheck` and `style` reflect content attributes.
- `classList` is a convenient accessor for `className`. The object it returns, exposes methods (`contains()`, `add()`, `remove()`, and `toggle()`) for manipulating the element's classes.
- `dataset` is a convenience feature for handling the `data-*` attributes, which are exposed as camel-cased properties. For instance, *`elm`*`.dataset.fooBar = 'test'` sets the `data-foo-bar` content attribute on *`elm`*.
- WHATWG HTML has `itemScope`, `itemType`, `itemId`, `itemRef`, `itemProp`, `properties` and `itemValue` for microdata.
- `click()`, `focus()` and `blur()` allows scripts to simulate clicks and moving focus.
- `accessKeyLabel` gives the shortcut key that the user agent has assigned for the element, which the author can influence with the `accesskey` attribute.
- `isContentEditable` returns true if the element is editable.
- `commandType`, `commandLabel`, `commandIcon`, `commandHidden`, `commandDisabled` and `commandChecked` is part of the command API.
- All event handler IDL attributes.

5.5 Extensions to Other Interfaces

Some interfaces in DOM Level 2 HTML have been extended.

- `HTMLOptionsCollection` now has a legacy caller, setter creator, and the members `add()`, `remove()` and `selectedIndex`
- `HTMLLinkElement` and `HTMLStyleElement` now implement the `LinkStyle` interface from CSSOM. [CSSOM]
- `HTMLFormElement` now has a named getter and an indexed getter.
- `HTMLSelectElement` now has a getter, `item()` and `namedItem()` methods, a setter creator, `selectedOptions` and `labels` IDL attributes, and members for the form constrain validation API: `willValidate`, `validity`, `validationMessage`, `checkValidity()` and `setCustomValidity()`.
- `HTMLOptionElement` now has a constructor `Option`.
- `HTMLInputElement` now has the members `files`, `height`, `indeterminate`, `list`, `valueAsDate`, `valueAsNumber`, `width`, `stepUp()`, `stepDown()`, the form constraint validation API members, `labels`, members for the text field selection API: `selectionStart`, `selectionEnd`, `selectionDirection`, `setSelectionRange()` and `setRangeText()`.
- `HTMLTextAreaElement` now has the members `textLength`, the form constraint validation API members, `labels` and the text field selection API members.
- `HTMLButtonElement` now has the form constraint validation API members and `labels`.
- `HTMLLabelElement` now has the member `control`.
- `HTMLFieldSetElement` now has the members `type`, `elements` and the form constraint validation API members.
- `HTMLAnchorElement` now has the members `relList`, `text`, the URL decomposition IDL attributes: `protocol`, `host`, `hostname`, `port`, `pathname`, `search` and `hash`. `HTMLLinkElement` and `HTMLAreaElement` also have the `relList` IDL attribute. `HTMLAreaElement` also has the URL decomposition IDL attributes.
- `HTMLImageElement` now has a constructor `Image`, the members `naturalWidth`, `naturalHeight` and `complete`.
- `HTMLObjectElement` now has the members `contentWindow`, the form constraint validation API members and a legacy caller.
- `HTMLMapElement` now has the member `images`.
- `HTMLTableElement` now has the member `createTBody()`.
- `HTMLIFrameElement` now has the member `contentWindow`.

In addition, most new content attributes also have corresponding IDL attributes on the elements' interfaces, e.g. the `sizes` IDL attribute on `HTMLLinkElement` which reflects the `sizes` content attribute.

5.6 Obsolete APIs

Some APIs are now either removed altogether, or marked as obsolete.

All IDL attributes that reflect a content attribute that is itself obsolete, are now also obsolete. For instance, the `bgColor` IDL attribute on `HTMLBodyElement` which reflects the obsolete `bgcolor` content attribute.

The following interfaces are marked obsolete since the elements are obsolete: `HTMLAppletElement`, `HTMLFrameSetElement`, `HTMLFrameElement`, `HTMLBaseFontElement`, `HTMLDirectoryElement` and `HTMLFontElement`.

The `HTMLIsIndexElement` interface is removed altogether since the HTML parser expands an `isindex` tag into other elements.

The following members of the `HTMLDocument` interface (which have now moved to `Document`) are now obsolete: `anchors` and `applets`.

6 HTML5 Changelogs

The changelogs in this section indicate what has been changed between publications of the HTML5 drafts, as well as changes in WHATWG HTML that do not affect HTML5. Rationale for changes can be found in the public-html@w3.org and whatwg@whatwg. org mailing list archives, and the WHATWG Weekly series of blog posts. More fundamental rationale is being collected on the WHATWG Rationale wiki page. Many editorial and minor technical changes are not included in these changelogs. Implementors are strongly encouraged to follow the development of the main specification on a frequent basis so they become aware of all changes that affect them early on.

The changes in the changelogs are in rough chronological order.

6.1 Changes since 29 March 2012

- The content model for `ruby` was changed with regards to nested `ruby` elements.
- Self-closing SVG `script` tags in the HTML syntax now execute.
- The placeholder section for the `find()` API has been dropped.
- An encoding declaration is now required in the HTML syntax even if only ASCII characters are used.
- Some bug fixes in the Drag and Drop API.
- The `inBandMetadataTrackDispatchType` IDL attribute was added to `TextTrack`.
- The `TextTrackCue()` constructor now has fewer arguments.
- The `accept` attribute now supports file extensions as well as MIME types.
- The `initialTime` IDL attribute on media elements has been dropped.
- The `startOffsetTime` IDL attribute on media elements has been renamed to `startDate`.

Further changes to WHATWG HTML that do not affect HTML5:

- Several changes and bug fixes in the Text Track API.
- `addElement()` was dropped from the Drag and Drop API.
- Media queries are now proxied for `iframe` elements with the `seamless` attribute.
- The `:enabled` and `:disabled` pseudo-classes now apply to `input` elements in the Hidden state.
- The `ssh`, `sip` and `magnet` schemes are now in the `registerProtocolHandler()`

whitelist.
- `table` elements now have 'box-sizing:border-box' by default.
- Bug fixes in the "potentially CORS-enabled fetch" algorithm.
- The document outline algorithm now ignores elements with the `hidden` attribute.
- Markup generators that are unable to provide required `alt` text can now use a specific attribute on `img` that makes validators ignore the missing `alt` error.
- Workers and shared workers now support `data:` URLs.
- The `inputmode` attribute has been added to `input` and `textarea`.
- The `autocomplete` attribute has been extended to support prefilling specific things.
- WebSocket supports sending `ArrayBufferView` as well as `ArrayBuffer`.
- The `border` attribute on `table` is non-conforming again.
- The canvas ImageData methods now assume 96dpi, and a set of "HD" methods have been introduced.
- The shared worker `connect` event now also exposes the source port in the `source` IDL attribute.
- Lone surrogates are converted to U+FFFD instead of throwing in WebSocket `send()`.
- The `setRangeText()` method has been added to `input` and `textarea`.
- The `srcset` attribute has been added to `img`.
- Application cache now has an `prefer-online` mode.
- Dialogs are now supported with the `dialog` element, the `inert` global attribute and the `dialog` method on `form`.
- The `resetTransform()` method, `currentTransform` IDL attribute, several IDL attributes for font metrics, `resetClip()` method, `imageSmoothingEnabled` IDL attribute, `addHitRegion()` method, `removeHitRegion()` method, support for dashed lines, have been added to the canvas 2d context.

6.2 Changes from 25 May 2011 to 29 March 2012

- Support for mutation observers was added.
- The `TextTrackCue` members `alignment`, `linePosition`, `textPosition` and `direction` were renamed to `align`, `line`, `position` and `vertical`, respectively.
- The `command` element now has a `command` attribute.
- Drag and drop content is now suggested to be filtered by user agents to prevent XSS attacks.
- The `translate` global attribute was added.
- The `showModalDialog()`, `alert()`, `confirm()` and `prompt()` methods are now allowed to do nothing during `pagehide`, `beforeunload` and `unload` events.
- The `script` element now supports `beforescriptexecute` and `afterscriptexecute` events.
- `window.onerror` now supports a fourth argument for column position.
- The `window.opener` IDL attribute can now return null in some cases.
- The `clearTimeout()` and `clearInterval()` methods were made synonymous.
- The CSS `@global` at-rule was introduced, for use together with `style` elements with the `scoped` attribute.
- The `embed` and `object` elements now have a legacy caller.
- The handling of `window.onerror`'s return value was changed to match reality.
- The `setTimeout()` API is now allowed to be throttled in background tabs.
- The `:valid` and `:invalid` pseudo-classes now apply to `form` elements.

- The `toBlob()` method on `canvas` now honors the origin-clean flag.
- The `activeElement` IDL attribute now points to the relevant browsing context container (e.g. `iframe`) when a child document has focus.
- The `atob()` method now ignores whitespace.
- The `dropzone` attribute was changed to use "`string:`" and "`file:`" instead of "`s:`" and "`f:`".
- The HTML parser was fixed to correctly handle a case involving foreign lands and foster parenting.
- The date-and-time microsyntaxes now allows a single space instead of a "T".
- Application cache no longer checks the MIME type of the cache manifest.
- The `cueAsSource` IDL attribute on `TextTrackCue` got renamed to `text`.
- The `window.onerror` API is now invoked with dummy arguments for cross-origin scripts.
- The `textarea` element's `value` and `textLength` IDL attributes have their newlines normalized to LF.
- The `q` element now has language-specific quotes rendered by default.
- The `data` element was introduced.
- The `time` element was redesigned to make it match how people wanted to use it. Its `pubdate` attribute was dropped.
- The legacy caller on `form` was removed.
- The `location.resolveURL()` method was removed.
- The `track` element now sniffs instead of obeying the MIME type.
- The `load()` method on documents created by `createDocument()` is now defined on the `XMLDocument` interface.
- Members of `HTMLDocument` moved to `Document` and `window.HTMLDocument` now just returns `window.Document`.
- The `MutableTextTrack` and `TextTrack` interfaces were merged and `TextTrackCue` was made more mutable.
- The `selectedOption` IDL attribute on `input` was dropped.
- Attribute values in Selectors are now case sensitive for all attributes.
- The `readyState` IDL attribute moved from `TextTrack` to `HTMLTrackElement`.
- The `text/html-sandboxed` MIME type was dropped.
- Floating point numbers are now allowed to begin with a "`.`" character.
- Navigating to an audio or video resource is now supported.
- Table cells now allow flow content but does not allow `header`, `footer`, sectioning content or heading content descendants.
- Adding a track to a media element now fires an `addtrack` event on the relevant track list objects.
- Setting `currentTime` on media elements before the media has loaded now defers the seek instead of throwing.
- Plugins are no longer disabled in sandboxed `iframe`s if they honor the `sandbox` attribute.
- Some tweaks to history navigation and related events.
- Media elements and `MediaController`s now get paused when they end.
- Events now support constructors and some `init*Event()` methods were removed.
- Media elements now fire a `suspend` event when the resource is loaded.
- Form submission now normalizes newlines to CRLF.
- Some tweaks around bidi and the `br` element.

- Large parts of the Editing section moved to HTML Editing APIs.
- `UndoManager` and related features moved to UndoManager and DOM Transaction.
- `isProtocolHandlerRegistered()`, `isContentHandlerRegistered()`, `unregisterProtocolHandler()` and `unregisterContentHandler()` were added.
- `registerContentHandler()` now has a blacklist of MIME types.
- `registerProtocolHandler()` now has a whitelist of protocols, but also supports any protocol that starts with "`web+`".
- Fragment identifiers for `text/html` resources now don't need to point to an element with a matching ID.
- `audio` elements are now allowed to have zero `source` children.
- There are now some restrictions on the use of bidi formatting characters.
- The `maxlength` and `size` attributes are allowed (but give warnings in validators) on `input` elements with `type=number`.
- The link relation "`shortcut icon`" is now allowed.
- Heading elements are now allowed to have the `heading` and `tab` roles.
- Things that use `EventTarget` now inherit from it instead of using "implements".
- The `setInterval()` API now clamps to 4ms instead of 10ms.
- The `select` element and its `options` collection now have a setter.
- `rel=help` on links now show a help cursor by default.
- Calls to `window.print()` before the document is loaded defers the print until it is loaded.
- Application cache gained an `abort()` method.
- `HTMLCollection`, `DOMTokenList`, `getElementsByClassName()`, `createHTMLDocument()`, HTML-specific overrides to some DOM Core features (like `createElement()`), some definitions, the `id` IDL attribute and ID handling moved to DOM4.
- Fragment identifiers can now survive redirects.
- The `pushState()` and `replaceState()` methods now change the history entry to GET.
- The command API now has its properties prefixed so they are now `commandLabel`, `commandIcon`, `commandHidden`, `commandDisabled` and `commandChecked`.
- The structured clone algorithm now supports sparse arrays.
- `window.postMessage` now supports transferring some objects instead of cloning them, and supports transferring `ArrayBuffer`.
- Application cache was made stricter in its MIME type checking.
- The `placeholder` attribute is now allowed on `input` elements with `type=number`.
- `MediaController` gained an `onended` event listener.
- The HTML parser changed its handling of U+0000 characters in some places.
- The `object` element gained a new attribute `typemustmatch`, to make it safer for authors to embed untrusted resources where they expect a certain content type.
- The `form` attribute was removed from `meter` and `progress`.
- The HTML parser was made more forward compatible in its handling of `ruby`.
- Some MIME types (e.g. `text/plain`) that are guaranteed to never be supported as scripting types for `script` were specified, so authors can safely use them for custom data blocks.
- `about:blank` documents created from `window.open()` now get a `load` event.

- `window.status` was specified to exist but do nothing.
- Drag and drop `DataTransferItems` was renamed to `DataTransferItemList`.
- Application cache now supports 'no-store' and HTTPS.
- The structured clone algorithm now supports getters.
- The `crossorigin` attribute has been added to `img`, `video` and `audio` to use CORS.
- The `external` IDL attribute has been added on `window` and has the members `AddSearchProvider()` and `IsSearchProviderInstalled()`.

Further changes to WHATWG HTML that do not affect HTML5:

- The 2d context now supports ellipses with the `arc()` and `arcTo()` methods and the new `ellipse()` method.
- The 2d context now supports `Path` objects. SVG path data can be added to a `Path`.
- The `http+aes:` and `https+aes:` URL schemes were added to allow sensitive resources to be held on untrusted servers.
- When the `itemprop` attribute is used on an element where microdata gets its value from an attribute (like `href` on `a` elements), that attribute is now required.
- `PeerConnection` was moved to WebRTC.
- WebVTT was moved to its own specification.
- WebSockets no longer receive messages in the `CLOSING` state.
- The Atom conversion algorithm was dropped.
- The `itemtype` attribute now allows multiple types.
- `CanvasPixelArray` was dropped in favor of `Uint8ClampedArray`.
- The microdata to RDF conversion algorithm was dropped.
- The `link` element is no longer allowed to have both `rel` and `itemprop`.
- WebSocket API disallows opening an insecure connection if the document uses a secure connection.
- The "storage mutex" is made optional.
- Web Storage no longer supports structured data.
- The `a` element got a new `download` attribute. This attribute is not included in HTML5.
- An experimental specification for the `window.find()` method was added.
- The 2d context `fillText()` and `strokeText()` methods now do not collapse whitespace.
- Microdata now handles infinite loops.
- Web Worker `location` now stringifies.
- Script errors in a Web Worker can now be detected in a parent worker or the document with the `onerror` handler.
- `EventSource` now supports CORS.
- `EventSource` was made stricter in its MIME type checking.
- Web Workers gained the `atob()` and `btoa()` methods.
- Web Workers gained the `ononline` and `onoffline` event handlers.
- WebSockets API has the `error` event again.
- WebSockets API now exposes the selected extensions.
- Various tweaks to the UDP `PeerConnection` API.
- WebSocket close code and reason are now supported in the API.
- Binary data is now supported in WebSockets.
- Redirects in WebSockets are now blocked for security reasons.

6.3 Changes from 5 April 2011 to 25 May 2011

- Support for the `javascript:` scheme in `img`, `object`, CSS, etc, has been dropped.
- The `toBlob()` method has been added to `canvas`.
- The `drawFocusRing()` method on the `canvas` 2d context has been split into two methods, `drawSystemFocusRing()` and `drawCustomFocusRing()`.
- The `values` attribute on `PropertyNodeList` has been replaced with a `getValues()` method.
- The `select` event has been specified.
- The `selectDirection` IDL attribute has been added to `input` and `textarea`.
- The `:enabled` and `:disabled` pseudo-classes now match `fieldset`, and the `:indeterminate` pseudo-class can now match `progress`.
- The `getKind()` method has been added to `TrackList`.
- The `MediaController` API and the `mediagroup` attribute have been added to synchronize playback of media elements.
- Some ARIA defaults have changed, and it is now invalid to specify ARIA attributes that match the defaults.
- The `getName()` method on `TrackList` was renamed to `getLabel()`.
- The `border` attribute on `table` is now conforming.
- The `u` element is now conforming.
- The `summary` attribute on `table` is now non-conforming.
- The `audio` attribute on `video` was changed to a boolean `muted` attribute.
- The `Content-Language` meta pragma is now non-conforming.

6.4 Changes from 13 January 2011 to 5 April 2011

- The `pushState` and `replaceState` features have been changed based on implementation feedback in Firefox, and `history.state` has been introduced.
- The `tracks` IDL attribute on media elements has been renamed to `textTracks`.
- Event handler content attributes now support JavaScript strict mode.
- The `forminput` and `formchange` events, and the `dispatchFormInput()` and `dispatchFormChange()` methods have been dropped.
- The `rel` keywords `archives`, `up`, `last`, `index`, `first` and related synonyms have been dropped.
- Removing a media element from the DOM and inserting it again in the same script now doesn't pause the media element.
- The `video` element's letterboxing rules are now specified in terms of CSS 'object-fit'.
- Cross-origin fonts now don't leak information about the font when drawn on a `canvas`.
- The character encoding declaration is now allowed to be within the first 1024 bytes instead of the first 512 bytes.
- The `onerror` event handler on `window` is now invoked for compile-time script errors as well as runtime errors.
- Script-inserted `script` elements now have `async` default to `true`, which can be set to `false` to make the scripts execute in insertion order.
- The `atob()` and `btoa()` methods have been specified.
- The suggested file extension for application cache manifest files has been changed from `.manifest` to `.appcache`.
- The `action` and `formaction` attributes are no longer allowed to have the empty string

as value.

6.5 Changes from 19 October 2010 to 13 January 2011

- Drag and drop model was refined.
- A new global `dropzone` attribute was added.
- A new `bdi` element was added to aid with user-generated content that may have bidi implications.
- The `dir` attribute gained a new "`auto`" value.
- A `dirname` attribute was added to `input` elements. When specified the directionality as specified by the user will be submitted to the server as well.
- A new `track` element and associated TextTrack API were added for video text tracks.
- The `type` attribute on the `ol` element is now allowed.

The `getSelection()` API moved to a separate DOM Range draft. Similarly `UndoManager` has been removed from the W3C copy of HTML5 for now as it is not ready yet.

6.6 Changes from 24 June 2010 to 19 October 2010

- Numerous changes to the HTML parsing algorithm based on implementation feedback.
- The `hidden` attribute now works for table-related elements.
- The `canvas` `getContext()` method is now defined to be able to handle multiple contexts better.
- The media elements' `startTime` IDL attribute was renamed to `initialTime` and `startOffsetTime` was added.
- The `prefetch` link relationship can now be used on `a` elements.
- The `datetime` attribute of `ins` and `del` no longer requires a time to be specified.
- Using PUT and DELETE as HTTP methods for the `form` element is no longer supported.
- The `s` element is no longer deprecated.
- The `video` element has a new `audio` attribute.

Per usual, lots of other minor fixes have been made as well.

6.7 Changes from 4 March 2010 to 24 June 2010

- The `ping` attribute has been removed from the W3C version of HTML5.
- The `title` element is optional for `iframe` `srcdoc` documents and other scenarios where a title is already available. As is the case with email.
- `keywords` is now a standard metadata name for the `meta` element.
- The `allow-top-navigation` value has been added for the `sandbox` attribute on the `iframe` element. It allows the embedded content to navigate its parent when specified.
- The `wbr` element has been added.
- The `alternate` keyword for the `rel` attribute of the `link` element can now be used to point to feeds again, even if the feed is not an alternative for the document.
- The HTML to Atom mapping has been removed from the W3C version of HTML5.

In addition lots of minor changes, clarifications, and fixes have been made to the document.

6.8 Changes from 25 August 2009 to 4 March 2010

- The `dialog` element has been removed. A section with advice on how to mark up

conversations has effectively replaced it.
- `document.head` has been introduced to provide convenient access to the `head` element from script.
- The link type `feed` has been removed. `alternate` with specific media types is to be used instead.
- `createHTMLDocument()` has been introduced as API to allow easy creation of HTML documents.
- Both the `meter` and `progress` elements no longer have "magic" processing of their contents because it could not be made to work internationally.
- The `meter` and `progress` elements, as well as the `output` element, can now be labeled using the `label` element.
- A new media type, `text/html-sandboxed`, was introduced to allow hosting of potentially hostile content without it causing harm.
- A `srcdoc` attribute for the `iframe` element was introduced to allow embedding of potentially hostile content inline. It is expected to be used together with the `sandbox` and `seamless` attributes.
- The `figure` element now uses a new element `figcaption` rather than `legend` because people want to use HTML5 long before it reaches W3C Recommendation.
- The `details` element now uses a new element `summary` for exactly the same reason.
- The `autobuffer` attribute on media elements was renamed to `preload`.

A whole lot of other smaller issues have also been resolved. The above list summarizes what is thought to be of primary interest to authors.

In addition to all of the above, Microdata, the 2D context API for `canvas`, and Web Messaging (`postMessage()` API) have been split into their own drafts at the W3C (the WHATWG still publishes a version of HTML5 that includes them):

- HTML Microdata
- HTML Canvas 2D Context
- HTML5 Web Messaging

Specific microdata vocabularies are gone altogether in the W3C draft of HTML5 and are not published as a separate draft. The WHATWG draft of HTML5 still includes them.

6.9 Changes from 23 April 2009 to 25 August 2009

- When the `time` element is empty user agents have to render the time in a locale-specific manner.
- The `load` event is dispatched at `Window`, but now has `Document` as its target.
- `pushState()` now affects the `Referer` (sic) header.
- `onundo` and `onredo` are now on `Window`.
- Media elements now have a `startTime` member that indicates where the current resource starts.
- `header` has been renamed to `hgroup` and a new `header` element has been introduced.
- `createImageData()` now also takes `ImageData` objects.
- `createPattern()` can now take a `video` element as argument too.
- The `footer` element is no longer allowed in `header` and `header` is not allowed in `address` or `footer`.
- A new control has been introduced: `<input type="tel">`
- The Command API now works for all elements.

- `accesskey` is now properly defined.
- `section` and `article` now take a `cite` attribute.
- A new feature called Microdata has been introduced which allows people to embed custom data structures in their HTML documents.
- Using the Microdata model three predefined vocabularies have also been included: vCard, vEvent, and a model for licensing.
- Drag and drop has been updated to work with the Microdata model.
- The last of the parsing quirks has been defined.
- `textLength` has been added as member of the `textarea` element.
- The `rp` element now takes phrasing content rather than a single character.
- `location.reload()` is now defined.
- The `hashchange` event now fires asynchronously.
- Rules for compatibility with XPath 1.0 and XSLT 1.0 have been added.
- The `spellcheck` IDL attribute now maps to a `DOMString`.
- `hasFeature()` support has been reduced to a minimum.
- The `Audio()` constructor sets the `autobuffer` attribute.
- The `td` element is no longer allowed in `thead`.
- The `input` element and `DataTransfer` object now have a `files` IDL attribute.
- The `datagrid` and `bb` have been removed due to their design not being agreed upon.
- The cue range API has been removed from the media elements.
- Support for WAI-ARIA has been integrated.

On top of this list quite a few minor clarifications, typos, issues specific to implementors, and other small problems have been resolved.

In addition, the following parts of HTML5 have been taken out and will likely be further developed at the IETF:

- Definition of URLs.
- Definition of Content-Type sniffing.

6.10 Changes from 12 February 2009 to 23 April 2009

- A new global attribute called `spellcheck` has been added.
- Defined that JavaScript `this` in the global object returns a `WindowProxy` object rather than the `Window` object.
- The `value` IDL attribute for `input` elements in the File Upload state is now defined.
- Definition of `designMode` was changed to be more in line with legacy implementations.
- The `drawImage()` method of the 2D drawing API can now take a `video` element as well.
- The way media elements load resources has been changed.
- `document.domain` is now IPv6-compatible.
- The `video` element gained an `autobuffer` boolean attribute that serves as a hint.
- You are now allowed to specify the `meta` element with a `charset` attribute in XML documents if the value of that attribute matches the encoding of the document. (Note that it does not specify the value, it is just a talisman.)
- The `bufferingRate` and `bufferingThrottled` members of media elements have been removed.
- The media element resource selection algorithm is now asynchronous.
- The `postMessage()` API now takes an array of `MessagePort` objects rather than just one.

- The second argument of the `add()` method on the `select` element and the `options` member of the `select` element is now optional.
- The `action`, `enctype`, `method`, `novalidate`, and `target` attributes on `input` and `button` elements have been renamed to `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget`.
- A "storage mutex" concept has been added to deal with separate pages trying to change a storage object (`document.cookie` and `localStorage`) at the same time. The `Navigator` gained a `getStorageUpdates()` method to allow it to be explicitly released.
- A syntax for SVG similar to MathML is now defined so that SVG can be included in `text/html` resources.
- The `placeholder` attribute has been added to the `textarea` element.
- Added a `keygen` element for key pair generation.
- The `datagrid` element was revised to make the API more asynchronous and allow for unloaded parts of the grid.

In addition, several parts of HTML5 have been taken out and will be further developed by the Web Applications Working Group as standalone specifications:

- WebSocket API
- WebSocket protocol
- Server-Sent Events
- Web Storage (`localStorage` and `sessionStorage`)
- Web SQL Database

6.11 Changes from 10 June 2008 to 12 February 2009

- The `data` member of `ImageData` objects has been changed from an array to a `CanvasPixelArray` object.
- Shadows are now required from implementations of the `canvas` element and its API.
- Security model for `canvas` is clarified.
- Various changes to the processing model of `canvas` have been made in response to implementation and author feedback. E.g. clarifying what happens when NaN and Infinity are passed and fixing the definitions of `arc()` and `arcTo()`.
- `innerHTML` in XML was slightly changed to improve round-tripping.
- The `toDataURL()` method on the `canvas` element now supports setting a quality level when the media type argument is `image/jpeg`.
- The `poster` attribute of the `video` element now affects its intrinsic dimensions.
- The behavior of the `type` attribute of the `link` element has been clarified.
- Sniffing is now allowed for `link` when the expected type is an image.
- A section on URLs is introduced dealing with how URL values are to be interpreted and what exactly authors are required to do. Every feature of the specification that uses URLs has been reworded to take the new URL section into account.
- It is now explicit that the `href` attribute of the `base` element does not depend on `xml:base`.
- It is now defined what the behavior should be when the base URL changes.
- URL decomposition IDL attributes are now more aligned with Internet Explorer.
- The `xmlns` attribute with the value `http://www.w3.org/1999/xhtml` is now allowed on all HTML elements.

- `data-*` attributes and custom attributes on the `embed` element now have to match the XML `Name` production and cannot contain a colon.
- WebSocket API is introduced for bidirectional communication with a server.
- The default value of `volume` on media elements is now 1.0 rather than 0.5.
- `event-source` was renamed to `eventsource` because no other HTML element uses a hyphen.
- A message channel API has been introduced augmenting `postMessage()`.
- A new element named `bb` has been added. It represents a user agent command that the user can invoke.
- The `addCueRange()` method on media elements has been modified to take an identifier which is exposed in the callbacks.
- It is now defined how to mutate a DOM into an infoset.
- The `parent` attribute of the `Window` object is now defined.
- The `embed` element is defined to do extension sniffing for compatibility with servers that deliver Flash as `text/plain`. (This is marked as an issue in the specification to figure out if there is a better way to make this work.)
- The `embed` can now be used without its `src` attribute.
- `getElementsByClassName()` is defined to be ASCII case-insensitive in quirks mode for consistency with CSS.
- In HTML documents `localName` no longer returns the node name in uppercase.
- `data-*` attributes are defined to be always lowercase.
- The `opener` attribute of the `Window` object is not to be present when the page was opened from a link with `target="_blank"` and `rel="noreferrer"`.
- The `top` attribute of the `Window` object is now defined.
- The `a` element now allows nested flow content, but not nested interactive content.
- It is now defined what the `header` element means to document summaries and table of contents.
- What it means to fetch a resource is now defined.
- Patterns are now required for the `canvas` element.
- The `autosubmit` attribute has been removed from the `menu` element.
- Support for `outerHTML` and `insertAdjacentHTML()` has been added.
- `xml:lang` is now allowed in HTML when `lang` is also specified and they have the same value. In XML `lang` is allowed if `xml:lang` is also specified and they have the same value.
- The `frameElement` attribute of the `Window` object is now defined.
- An event loop and task queue is now defined detailing script execution and events. All features have been updated to be defined in terms of this mechanism.
- If the `alt` attribute is omitted a `title` attribute, an enclosing `figure` element with a `legend` element descendant, or an enclosing section with an associated heading must be present.
- The `irrelevant` attribute has been renamed to `hidden`.
- The `definitionURL` attribute of MathML is now properly supported. Previously it would have ended up being all lowercase during parsing.
- User agents must treat US-ASCII as Windows-1252 for compatibility reasons.
- An alternative syntax for the DOCTYPE is allowed for compatibility with some XML tools.
- Data templates have been removed (consisted of the `datatemplate`, `rule` and `nest` elements).

- The media elements now support just a single `loop` attribute.
- The `load()` method on media elements has been redefined as asynchronous. It also tries out files in turn now rather than just looking at the `type` attribute of the `source` element.
- A new member called `canPlayType()` has been added to the media elements.
- The `totalBytes` and `bufferedBytes` attributes have been removed from the media elements.
- The `Location` object gained a `resolveURL()` method.
- The `q` element has changed again. Punctuation is to be provided by the user agent again.
- Various changes were made to the HTML parser algorithm to be more in line with the behavior Web sites require.
- The `unload` and `beforeunload` events are now defined.
- The IDL blocks in the specification have been revamped to be in line with the upcoming Web IDL specification.
- Table headers can now have headers. User agents are required to support a `headers` attribute pointing to a `td` or `th` element, but authors are required to only let them point to `th` elements.
- Interested parties can now register new `http-equiv` values.
- When the `meta` element has a `charset` attribute it must occur within the first 512 bytes.
- The `StorageEvent` object now has a `storageArea` attribute.
- It is now defined how HTML is to be used within the SVG `foreignObject` element.
- The notification API has been dropped.
- How [[Get]] works for the `HTMLDocument` and `Window` objects is now defined.
- The `Window` object gained the `locationbar`, `menubar`, `personalbar`, `scrollbars`, `statusbar` and `toolbar` attributes giving information about the user interface.
- The application cache section has been significantly revised and updated.
- `document.domain` now relies on the Public Suffix List. [PSL]
- A non-normative rendering section has been added that describes user agent rendering rules for both obsolete and conforming elements.
- A normative section has been added that defines when certain selectors as defined in the Selectors and the CSS3 Basic User Interface Module match HTML elements. [SELECTORS] [CSSUI]

Web Forms 2.0, previously a standalone specification, has been fully integrated into HTML5 since last publication. The following changes were made to the forms chapter:

- Support for XML submission has been removed.
- Support for form filling has been removed.
- Support for filling of the `select` and `datalist` elements through the `data` attribute has been removed.
- Support for associating a field with multiple forms has been removed. A field can still be associated with a form it is not nested in through the `form` attribute.
- The `dispatchChangeInput()` and `dispatchFormChange()` methods have been removed from the `select`, `input`, `textarea`, and `button` elements.
- Repetition templates have been removed.
- The `inputmode` attribute has been removed.
- The `input` element in the File Upload state no longer supports the `min` and `max` attributes.
- The `allow` attribute on `input` elements in the File Upload state is no longer authoritative.
- The `pattern` and `accept` attributes for `textarea` have been removed.

- RFC 3106 is no longer explicitly supported.
- The `submit()` method now just submits, it no longer ensures the form controls are valid.
- The `input` element in the Range state now defaults to the middle, rather than the minimum value.
- The `size` attribute on the `input` element is now conforming (rather than deprecated).
- `object` elements now partake in form submission.
- The `type` attribute of the `input` element gained the values `color` and `search`.
- The `input` element gained a `multiple` attribute which allows for either multiple e-mails or multiple files to be uploaded depending on the value of the `type` attribute.
- The `input`, `button` and `form` elements now have a `novalidate` attribute to indicate that the form fields should not be required to have valid values upon submission.
- When the `label` element contains an `input` it may still have a `for` attribute as long as it points to the `input` element it contains.
- The `input` element now has an `indeterminate` IDL attribute.
- The `input` element gained a `placeholder` attribute.

6.12 Changes from 22 January 2008 to 10 June 2008

- Implementation and authoring details around the `ping` attribute have changed.
- `<meta http-equiv=content-type>` is now a conforming way to set the character encoding.
- API for the `canvas` element has been cleaned up. Text support has been added.
- `globalStorage` is now restricted to the same-origin policy and renamed to `localStorage`. Related event dispatching has been clarified.
- `postMessage()` API changed. Only the origin of the message is exposed, no longer the URL. It also requires a second argument that indicates the origin of the target document.
- Drag and drop API has got clarification. The `dataTransfer` object now has a `types` attribute indicating the type of data being transferred.
- The `m` element is now called `mark`.
- Server-sent events has changed and gotten clarification. It uses a new format so that older implementations are not broken.
- The `figure` element no longer requires a caption.
- The `ol` element has a new `reversed` attribute.
- Character encoding detection has changed in response to feedback.
- Various changes have been made to the HTML parser section in response to implementation feedback.
- Various changes to the editing section have been made, including adding `queryCommandEnabled()` and related methods.
- The `headers` attribute has been added for `td` elements.
- The `table` element has a new `createTBody()` method.
- MathML support has been added to the HTML parser section. (SVG support is still awaiting input from the SVG WG.)
- Author-defined attributes have been added. Authors can add attributes to elements in the form of `data-`*`name`* and can access these through the DOM using `dataset[`*`name`*`]` on the element in question.
- The `q` element has changed to require punctuation inside rather than having the browser render it.
- The `target` attribute can now have the value `_blank`.

- The `showModalDialog` API has been added.
- The `document.domain` API has been defined.
- The `source` element now has a new `pixelratio` attribute useful for videos that have some kind encoding error.
- `bufferedBytes`, `totalBytes` and `bufferingThrottled` IDL attributes have been added to the `video` element.
- Media `begin` event has been renamed to `loadstart` for consistency with the Progress Events specification.
- `charset` attribute has been added to `script`.
- The `iframe` element has gained the `sandbox` and `seamless` attributes which provide sandboxing functionality.
- The `ruby`, `rt` and `rp` elements have been added to support ruby annotation.
- A `showNotification()` method has been added to show notification messages to the user.
- Support for `beforeprint` and `afterprint` events has been added.

Acknowledgments

The editors would like to thank Ben Millard, Bruce Lawson, Cameron McCormack, Charles McCathieNevile, Dan Connolly, David Håsäther, Dennis German, Frank Ellermann, Frank Palinkas, Futomi Hatano, Gordon P. Hemsley, Henri Sivonen, James Graham, Jens Meiert, Jeremy Keith, Jürgen Jeka, Krijn Hoetmer, Leif Halvard Silli, Maciej Stachowiak, Mallory van Achterberg, Marcos Caceres, Mark Pilgrim, Martijn Wargers, Martin Leese, Martyn Haigh, Masataka Yakura, Michael Smith, Mike Taylor, *Ms2ger*, Olivier Gendrin, Øistein E. Andersen, Philip Jägenstedt, Philip Taylor, Randy Peterman, Toby Inkster, and Yngve Spjeld Landro for their contributions to this document as well as to all the people who have contributed to HTML over the years for improving the Web!

References

[CSSOM]
CSSOM, Glenn Adams, Shane Stephens and Anne van Kesteren. W3C.

[CSSUI]
CSS Basic User Interface Module, Tantek Çelik. W3C.

[DOCTYPE]
Activating Browser Modes with Doctype, Henri Sivonen.

[DOM2HTML]
Document Object Model (DOM) Level 2 HTML Specification, Johnny Stenback, Philippe Le Hégaret and Arnaud Le Hors. W3C.

[HTML]
HTML, Ian Hickson. WHATWG.

[HTML4]
HTML 4.01 Specification, Dave Raggett, Arnaud Le Hors and Ian Jacobs. W3C.

[HTML5]
HTML5, Ian Hickson. W3C.

[PSL]
Public Suffix List. Mozilla Foundation.

[SELECTORS]
Selectors Level 4, Elika J. Etemad. W3C.

[XHTML1]

XHTML™ 1.1 - Module-based XHTML - Second Edition, Murray Altheim and Shane McCarron.

[XML]

Extensible Markup Language, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen et al.. W3C.