L2/09-411

BidiProposal

From Internationalization

A Proposal for HTML Improvements for Bidi

Contents

Main Author

Aharon Lanin, Bidi Tech Lead, Google.

Additional Contributors (in alphabetical order)

Adil Allawi, Technical Director, Diwan Software.
Matitiahu Allouche, Bidi Architect, IBM.

Document status

This document currently has no official status within the W3C or the W3C Internationalization Activity. It is being used to sketch out an set of initial ideas by a small community of bidi experts in preparation for consideration by the W3C i18n Working Group. (The WG will then examine the proposals with a view to making recommendations to HTML and CSS Working Groups after further discussion and feedback are incorporated.)

This is a document for discussion and as such it is expected to be subject to ongoing and substantive changes. It is inappropriate to cite this document as other than work in progress.

Introduction

Authoring a web app that needs to support both right-to-left and left-to-right interfaces, or to take as input and display both left-to-right and right-to-left data, usually presents a number of challenges that make it an especially laborious and bug-prone task. Some of these are due to browser bugs, but some can be traced to a lapse in the specification of the bidirectional aspects of a given HTML feature. And some of these challenges could be greatly simplified by adding a few strategically placed new HTML features. The following document proposes fixes for some of the most repetitive pain points.

Preliminaries

- LTR: left-to-right
- RTL: right-to-left
- UBA: the Unicode Bidi Algorithm (http://unicode.org/reports/tr9/) , which determines the visual order in which bidi text is to be displayed.
- All examples in this document are in "fake bidi", i.e. use uppercase English to represent RTL characters and lowercase English for LTR characters. For example, according to the UBA, "RTL TEXT" is normally displayed as "TXET LTR".
- Much of this proposal deals with determining and declaring the overall direction of text. This is because text displayed in the wrong direction is often garbled. For example, "10 main st." is displayed in RTL as

```
.main st 10
```

and "MAKE html WORK FOR YOU" is displayed in LTR as

```
EKAM html UOY ROF KROW
```

instead of the intended

```
UOY ROF KROW html EKAM
```

and is quite unreadable.

Part 1: New HTML Features

1.1. support bidi isolation of inline element content

Background

The UBA's rendering of a piece of text depends not only on its explicitly declared direction (e.g. the dir attribute value on the parent element), but on the implicit directional properties of the characters preceding and following it. For example, in an RTL context, "john: " is displayed as "john: " when followed by "susan" (i.e. "john: susan"), but as " :john" when followed by "SUSAN" (i.e. "NASUS :john") - note the change in colon positioning.

The bidi formatting characters LRO, RLO, LRE, RLE, and PDF have particularly strong influence on what surrounds them. For example, RLO makes all text up to the next PDF behave as RTL characters, making "hello" display as "olleh".

The Problem

Most documents contain a large number of self-contained entities whose content must not influence the directional rendering of what precedes or follows them. Furthermore, the document author naively expects such an entity to be displayed visually between what precedes it and what follows it, laid out in the current direction: preceding - entity - following.

Examples of such entities are legion: the title of an article, the name of an author, a description, etc.

As long as the entire document and all the entities it contains are of uniform direction, there is no problem. Arbitrary-direction entities also don't cause a problem when they are displayed as a separate block element (which is treated as a separate "paragraph" in UBA terms). However, when an inline entity is allowed to contain text of arbitrary direction, bad things start happening, and existing HTML mark-up is powerless to stop it.

Example 1: <span dir=rtl>PURPLE PIZZA</span> - 3 reviews

The entity here is the name of a restaurant, marked here with a span and its RTL direction, to be displayed in an LTR context. Surprisingly enough, this is displayed as

```
3 - AZZIP ELPRUP reviews
```

instead of the intended

```
AZZIP ELPRUP - 3 reviews
```

and is effectively unreadable. This happens because according to the UBA, a number "sticks" to the strong-directional run preceding it, whether or not the run is wrapped in an embedding level as shown here.

Example 2: USE css (<span dir=ltr>position:relative</span>).

The entity here is a code snippet ("position:relative"), marked with a span and its LTR direction, to be displayed in an RTL context. Despite the RTL context, it is preceded by the LTR word "css" because technical terms and brand names often appear in their original Latin script in RTL text. Surprisingly enough, this is displayed as

```
                                                            .(css (position:relative ESU
```

instead of the intended

```
                                                            .(position:relative) css ESU
```

This happens because the LTR word "css" before the entity "sticks" to the LTR entity according to normal UBA rules.

Example 3: documents &gt; <span dir=rtl>MY FIRST NOVEL</span> &gt; <span dir=rtl>CHAPTER 1</span>

The entities here are folder names displayed in "breadcrumbs" in an LTR context, where two of the folder names happen to be RTL. Suprisingly enough, this is displayed as

```
documents > 1 RETPAHC < LEVON TSRIF YM
```

instead of the intended

```
documents > LEVON TSRIF YM > 1 RETPAHC
```

i.e. with the RTL folder names visually in the wrong order (and the arrow between them reversed). This happens because according to the UBA, the two RTL entities "stick" together, whether or not they are wrapped in <span>s as shown here.

Example 4: joe hacker`RLO`: overdrawn

The entity here is the name of a user, as chosen by a malicious user to include the invisible RLO character (U+202E), followed by a status string. Obviously, the user's name is "HTML-escaped (http://en.wikipedia.org/wiki/HTML#Character_and_entity_references) " when displayed, but this does not do anything to the RLO character. The not very surprising outcome is that this is displayed as

```
joe hacker: nwardrevo
```

where the entity influenced the display of what follows it, reversing its characters. This has security implications and has surfaced on blogs (http://simonwillison.net/2009/Mar/15/bidi/#comments) . On the other hand, it does not even have to be due to malicious use, only to the inadvertently bad trimming of an overly-long string.

Currently, there is no reliable way to deal with examples 1 - 3 using mark-up, except by redundantly marking an entity's surroundings with the document's direction, which is counterintuitive and painful to implement. The usual way to deal with 1 - 3 is to surround an entity in either LRM or RLM characters - LRM in an LTR context, and RLM in an RTL context. This prevents the entity from "sticking" to what precedes or follows it.

Using LRM and RLM in the document has several disadvantages:

- The LRM or RLM is being used to address a layout issue that reflects the structure of the document, i.e. to indicate the boundary of an entity. There should be a way to express it in mark-up, not magic Unicode characters. In fact, the entity is typically already surrounded by an element that either gives it style or indicates its direction; why can't the element itself be used to indicate an entity?
- Not all search engines (e.g. the browser's own CTRL-F) are smart enough to ignore invisible Unicode characters such as LRM and RLM. This makes a document using such characters less searchable: the user searches for "A B", but does not find "A B" because there is an invisible character between them. The problem is even worse if the user copy-pastes text - accidentally including the LRM or RLM character - from the page into a search box. Avoiding such searchability problems when programmatically creating a potentially bidi document requires doing complicated logic to decide whether the LRM or RLM is really necessary instead of simply plopping them

around every entity.
- In a web application, having to add logic to choose between an LRM and an RLM is a pain, especially when the existing code layer does not happen to have easy access to the context's direction.

Furthermore, LRMs and RLMs do not help in example 4. Nor is there any mark-up to solve it. The only current way to deal with it is for the application to either remove any LRE, RLE, LRO, RLO, or PDF characters in it, or to remove any extra PDFs and then add any missing ones at the end. This is a rarely-implemented pain in the neck.

Proposed Solution

Add an element attribute to HTML used to make an inline element directionally isolated from its surroundings. A tentative name for the new attribute might be bdi, for "bidirectional isolate", as in <span dir="rtl" bdi>. The attribute would take three values:

- no, specifying no special isolation. This is the default, except in special cases indicated in the sections below.
- yes, specifying isolation.
- bdi, a synonym for yes. Allows specifying the attribute without a value for conciseness, e.g. <span dir="rtl" bdi>.

Applications generating HTML would use bdi routinely on elements that wrap an inserted data string (usually in conjunction with indicating its direction using the dir attribute).

The exact definition of the effects of bdi=yes on an element:

- The element, even when empty, is to be displayed as if it were surrounded with strong-directional characters of the last explicit embedding level within which it appears. In most cases, the last explicit embedding level is simply the effective direction of the parent element. (The exception is when the element is between LRE/RLE and PDF characters, which is discouraged by the W3C.) For example, take:

```
<span dir=rtl bdi>PURPLE PIZZA</span> - 3 reviews
```

In a dir=ltr element, it should be displayed the same as

```
&lrm;<span dir=rtl>PURPLE PIZZA</span>&lrm; - 3 reviews
```

i.e. as

```
AZZIP ELPRUP - 3 reviews
```

- The "imaginary" LRM/RLM characters must not actually appear in the output, e.g. for the purposes of a "copy to clipboard" operation.
- The effects of LRE, RLE, LRO, RLO, and PDF characters appearing in the element will never extend beyond the element: unbalanced PDFs will be ignored and missing PDFs will be assumed at the close of the element. **Open Issue**: should this rule be applied to all elements, not just those with bdi=yes?

The use of "imaginary" characters by higher level protocols such as HTML is explicitly allowed by the UBA's section 4.3, HL5 ("Provide artificial context").

1.2. support auto-direction

The Problem

Many web applications with an RTL-language interface or an RTL-language data source need to display and accept as input both LTR and RTL data. Furthermore, the application often does not know and can not control the direction of the data.

For example, an online book store that carries books in many languages needs to display the original book titles regardless of the language of the user interface. Thus, a Hebrew or Arabic book title may appear in an English interface, and vice-versa. The direction of the title may be available as a separate attribute, but more likely it isn't, and needs to be guessed. The safest guess is on the basis of the characters making up the title.

If this site also allows user comments or reviews, it is unreasonable to limit these to one language. For example, for an English book listed in an Arabic or Hebrew interface, it is perfectly reasonable to get comments both in English and in the book's language. The application does not know what the user will type until the user types it.

Unless opposite-direction data is explicitly declared as such, it is often displayed garbled as shown above. Perhaps even worse, the user experience of typing opposite-direction data is quite awkward due to the cursor and punctuation jumping around during data entry and difficulty in selecting text.

Currently, avoiding such problems requires that the application implement logic to estimate the data's direction - and use it in the many places where it is needed. Such logic is not easy to implement, since it requires using long tables of strong-RTL and/or strong-LTR characters, and becomes non-obvious when a string contains both. For an input element, where the direction must be automatically set as the user types the text, there is no choice but to implement the estimation logic in page scripts, thus requiring even more advanced programming skills. As a result, few applications wind up doing direction estimation, and a poor user experience is quite common for web pages mixing LTR data in an RTL interface or vice-versa.

Not The Problem (skip if not very interested)

The issue at hand is with text data that is basically compatible with the UBA. That is, given the correct base direction, applying the UBA will display the text intelligibly. The only problem is that we don't know the correct base direction.

This is distinct from a different, harder issue: text mixing LTR and RTL without using the formatting characters necessary to display it intelligibly using standard UBA rules. Whichever base direction is applied, the text will not be displayed as intended. Examples of such data are not as rare as one might think:

- Path or URL that includes consecutive RTL folder or file names
- "Tweets" that include both an RTL phrase and LTR parts like @name and a URL
- An RTL sentence that attempts to give a phone number with spaces in it
- Sentence containing an opposite-direction quotation that starts with a number or ends with punctuation
- Multi-paragraph text containing both LTR and RTL paragraphs, e.g. an RTL restaurant review followed by restaurant address in Latin script.

Such text does not include the Unicode formatting characters that could fix its display either because it must conform to a syntax that would misinterpret such characters, or simply because it was created by a human user that does not know such characters exist, much less how to enter or use them. Given the text's syntax, or at least a set of patterns for the problematic parts, the text could, in theory, be parsed into its constituent parts, and formatting characters added to make the text display correctly.

Although this is a painful real-world problem, it is unrelated to HTML per se and currently lacks a mature solution. We are not proposing one here.

Estimation Algorithms (skip if not very interested)

A data string's direction is obvious when it contains either LTR or RTL characters, but not both. The following heuristic algorithms have been used when the data does contain both LTR and RTL characters:

- First character with strong direction. This is the algorithm specifically mandated by the UBA for choosing a paragraph's base direction (unless overridden by a higher-level protocol, which is what currently always happens in HTML). This has the advantage of being easy to understand (and even surmise) for the user, and text is usually more readable when starting with a word in its overall direction. Nevertheless, it is not uncommon for an RTL phrase to start with an LTR word like a brand name or a technical term, and here this algorithm fails.
- Does the string contain any RTL characters? This fails for LTR text that includes some RTL, which is quite uncommon, but not unheard of.
- Word count: does the percentage of RTL words exceed some threshold value? Works very well, but also fails unexpectedly.

Different approaches have been preferred in different contexts: first-strong for search boxes, any-RTL for advertisements, and word-count for longer texts like e-mails. Nevertheless, it is worth pointing out that the choice of the precise algorithm is an optimization. For most real-world data strings, all these estimation algorithms will give the same correct result.

In addition to the basic algorithm choice, there are also several side issues:

- Sometimes, there may be good reason to want to bias the estimation to a particular direction unless the actual value clearly indicates otherwise. For example, the value may be one title from a feed whose contents are, generally speaking, in a particular known language.
- When the value contains no strong-directional characters, it usually seems best to display it in the same direction as its surroundings, i.e. inherit the direction.
- How should those parts of the string bracketed in LRE / RLE / LRO / RLO and PDF characters be treated? It is possible to simply ignore such bracketing characters, and this is actually specified by the UBA for its first-strong algorithm. Another possibility is to ignore them together with the substrings they bracket. The rationale for the latter approach is that the direction we estimate for the whole string will not be applied to the bracketed substrings anyway. In fact, if part of a string is explicitly declared LTR, it is usually because the string overall is RTL, and vice-versa. On the other hand, if the string contains no strong-directional characters outside the declared substrings, and all the declared substrings give the same direction, then it might be best to estimate the string overall to be of the same direction as the declared substrings.

Proposed Solution

Make simple direction estimation functionality available in the browser by allowing the dir attribute to take on a value indicating that the user agent is responsible for estimating the direction of the element's contents. Open-source implementations of direction estimation for text exist and are very fast and quite small.

The tentative name for such a value would be "auto". Specifying dir=auto would direct the user agent to examine the element's text content and estimate whether it is LTR or RTL using well-defined heuristics based on the inherent direction of the characters (as defined by the Unicode standard). The result it will return for text mixing LTR and RTL characters, although well-defined, may or may not be correct as judged by a human user.

Although dir=auto is allowed on any element, it is primarily intended for elements wrapping a "single-origin" piece of text, e.g. a text input. The more complex the element's structure, the higher the chances that it mixes LTR and RTL content, and the lower the chances that dir=auto will succeed in displaying the contents intelligibly. It is meaningless to use dir=auto on content mixed to the extent that it is unintelligible in both LTR and RTL (when displayed by standard UBA rules).

The new dir value should also be added to the CSS direction property's repertoire for completeness. However, since W3C guidelines (http://www.w3.org/TR/i18n-html-tech-bidi/#ri20030728.092130948) recommend that direction be declared using the dir attribute, not CSS, this is first and foremost an HTML issue.

Details:

- The basic algorithm to be used by dir=auto is "first strong-directional character", as defined by the UBA.
- The content to be examined is all descendant text nodes, visited in "in-fix" order, except for those under a descendant element with a unicode-bidi style other than "normal", e.g. a <bdo> element or an element with an explicit dir attribute value, including "auto".
- No attempt will be made to exclude "hidden" content, whether using display:none style or any other invisibility technique.
- When none of the examined content contains strong-directional characters, dir=auto will use the inherited effective direction.
- Elements with dir=auto will be considered to have bdi=yes by default.
- Ideally, the current effective direction of an auto-directional element should be exposed to scripts as an element property, e.g. effectiveDirection.
- For backward compatibility, dir=auto will not be the default for any other currently defined elements except as defined below.

Open Issues:

- Should dir=auto be the default for <input type=text> and <textarea> elements? Although it would be very useful in most cases, it is not backward compatible and could sometimes cause problems.
- Should dir=auto be the default for <option> elements? There is no backward compatibility issue here, since the dir attribute is currently ineffective on option elements in most browsers, but it seems silly to do this only for this one element.
- Should any additional well-defined estimation algorithms also be made available? (Answering in the affirmative requires disambiguating the algorithm details to make it sufficiently well-defined.)

1.3. support reporting the chosen direction of <input> and <textarea> in form submissions

Background

In many applications, it is necessary to allow the user to enter text of either direction into a given <input type=text> or <textarea> element, regardless of the page's direction.

Although algorithms for estimating the direction of a string exist (and hopefully will be exposed by the browser as dir=auto), they remain heuristic for mixed-script strings.

As a result, all major browsers provide some way for the user to explicitly set the direction of an <input type="text"> or <textarea> element, e.g. via keyboard shortcuts, so the text being entered by the user is displayed correctly.

The Problem

Once the text entered by the user has been submitted to the server, the direction in which it was displayed in the page is lost, unless explicitly added to the form as an invisible input by page scripts. However, scripts are not available in all environments, e.g. e-mail forms. As a result, in such an environment, the application is forced to guess at the direction of a string submitted by the user, will sometimes get it wrong, and as a result display it incorrectly in subsequent pages.

Proposed Solution

Support a new attribute in <input> and <textarea> that would specify the name under which the effective direction value of the element would be submitted in the form. The effective direction could be explicit or inherited; auto would get collapsed to the one being used by the browser. The new attribute could be called submit-dir-as, e.g. <textarea name=foo submit-dir-as=foo_dir>.

1.4. support option for image elements to be flipped horizontally in RTL

Background

Although most images, e.g. photos, are equally applicable to LTR and RTL pages, some images are inherently and primarily "handed" or "directional", and need to appear in a mirror image in an RTL page. Common examples include various arrow and "connector" images. A less obvious example might be star rating images: the "full" half of a half-star needs to be on the left in LTR and on the right in RTL.

The Problem

Currently, the author of a page to be localized into both LTR and RTL languages is forced to create two separate versions of each "handed" image, stored in two separate files, and use one or the other depending on the page language by changing the src attribute of the <img>. This process is monotonous and error-prone.

Proposed Solution

Being able to tell the browser to do such flips automatically would make it that much easier for web applications to support both LTR and RTL interfaces. Only one image would be provided by the page, and the <img> element's attributes would not even have to differ between LTR and RTL pages. The single image file provided by the page would come with instructions in the <img> element to be flipped by the browser when its parent element has effective RTL direction. (The <img> element's own dir does not count, since it is used primarily to indicate the direction of its tooltip text as specified by alt and title attributes, which need not match the surroundings.)

One possibility for such a specification would be with a new HTML attribute: flip="no|yes|ltr|rtl".

The default "no" would mean that the image should be displayed as is, and "yes" would mean that it must be displayed horizontally mirrored. To indicate that the flip should only occur when the element's parent has effective RTL direction, the value should be "rtl", and of course "ltr" would specify the inverse.

1.5. support align=start and align=end attribute values

Background

LTR text is most readable left-aligned, and RTL text is most readable right-aligned.

HTML and CSS recognize this by making LTR block elements left-aligned and RTL block elements right-aligned by default. However, this default no longer applies when alignment is explicitly set on the element's ancestor, and inheritance prevails.

CSS3 (http://www.w3.org/TR/css3-text/#text-align)
adds two values to the text-align property's repertoire. The first, text-align:start, indicates "left" for elements with effective LTR direction, and "right" for elements with effective RTL direction. The second, text-align:end, indicates the inverse. In fact, the default text-align value (where it is not inherited) is now defined as start.

The availability of text-align:start is very useful: it makes it possible to specify the most readable alignment even when a different alignment may have been specified on an ancestor element, without having to make a distinction between the LTR and RTL cases. Furthermore, this interacts very nicely with the proposed dir=auto attribute value, where the element content's direction is unknown to the page's author.

Firefox and WebKit already support these text-align values.

However, the HTML align attribute is deprecated in favor of CSS's text-align property, and the HTML 5 specification does not include start and end values for it.

Nevertheless, the align attribute is still widely used.

WebKit supports start and end values not only for the text-align CSS property, but for the align attribute too.

The Problem

Once new text-align values are supported, and especially given that one of the new values is the new default, it is inevitable that some browsers will also support them for the align attribute, as WebKit has already done. As long as start and end are not part of the HTML specification, it is also inevitable that other browsers won't support them.

Users will continue to use the align attribute, as they have done up to now, despite its being deprecated, because it is convenient. In fact, it seems like the right thing to do in the bidi case: alignment and direction are intimately related, and according to W3C guidelines (http://www.w3.org/TR/i18n-html-tech-bidi/#ri20030728.092130948) , direction is supposed to be indicated using the dir attribute, not a CSS property. Thus, users will try to use the new start and end values for the align attribute because they are logical and useful. Browser incompatibility in the treatment of the align attribute will lead to user confusion, and thus pages working properly in only some browsers.

Proposed Solution

Add align=start and align=end to the HTML specification, with start being the default value for the align attribute. When an element has no align value, but an ancestor does, and it is start or end, the element behaves as if it itself has the start or end align value, subject to re-interpretation according to its own direction, not as the frozen left or right value of the ancestor.

Part 2: Standardizing Bidi Aspects of Existing HTML Features

2.1. <br> and embedded block elements should serve as bidi separators

Background

The UBA's rendering of a piece of text depends not only on its explicitly declared direction (e.g. the dir attribute value on the parent element), but on the implicit directional properties of the characters preceding and following it. For example, in an RTL context, "john: " is displayed as "john: " when followed by "susan" (i.e. "john: susan"), but as " :john" when followed by "SUSAN" (i.e. "NASUS :john") - note the change in colon positioning.

The bidi formatting characters LRO, RLO, LRE, RLE, and PDF have particularly strong influence on what surrounds them. For example, RLO makes all text up to the next PDF behave as RTL characters, making "hello" display as "olleh".

In the UBA, whitespace provides almost no separation against either kind of bidi influence.

However, the UBA's sections 3.3.1 and 3.3.2 require that the bidi state be completely reset at a "paragraph break". This means that strong-directional text (e.g. letters) and explicit bidi formatting characters (e.g. RLE and RLO) in one paragraph have no effect on the formatting of the text in the next paragraph and vice-versa. This is a very high level of bidi separation.

In plain text, "newline" characters like the line feed (U+000A) and carriage return (U+000D) are commonly used both to end paragraphs and simply to wrap logical lines. The former usage needs a UBA paragraph break, while the latter usage wants no more bidi separation than other kinds of whitespace. The UBA resolves this ambiguity in favor of the paragraph break because of its importance. All common UBA implementations for plain text treat newline characters as a UBA paragraph break, in accordance with the UBA specification.

The UBA leaves the definition of a "paragraph" in higher-level protocols like HTML up to the protocol.

The Problem

It is well-accepted that HTML block elements like <div> and <p> form UBA paragraphs, and this is implemented by all major browsers. Thus, whatever happens inside a block element has no effect on the bidirectional rendering of the text before it or after it.

However, there is no standard definition of whether a block element serves as a UBA break between the text preceding

and following it, i.e. whether the text preceding a <div></div> or an <hr> (defined (http://www.w3.org/TR/html4/sgml/dtd.html#block)
to be a block element) should behave as if it were in the same UBA paragraph as the text following it. For short, we will call block elements with text on both sides "embedded".

For <br>, HTML 4 does explicitly specify (http://www.w3.org/TR/html4/struct/text.html#edef-BR) that it is to be treated for bidi purposes as whitespace, and not as a paragraph break. The arguments for this decision seem to be that:

- <br> is defined as an inline element.
- The preferred way to demarcate a paragraph in HTML is as a <p> or some other block element.

Firefox and Opera follow this specification and treat <br> as whitespace for UBA purposes. Perhaps influenced by this, they also treat embedded block elements as UBA whitespace.

In actual usage, however, <br> is a very popular element and is used to form paragraphs at least as often as <p>, just like newlines in plain text. In fact, unlike newlines in plain text, it is almost always used for that purpose, as opposed to just wrapping a line to fit in a limited amount of space, simply because HTML normally takes care of line wrapping by itself.

As a result, Firefox's implementation of <br> as UBA whitespace, despite being in accordance with the current HTML specification, is regularly reported as a bug. It results in innocent-looking HTML like

```
1. his name is JOHN.<br>
2. SUSAN is a friend of his.
```

being rendered as

```
1. his name is .NHOJ
NASUS .2 is a friend of his.
```

Because the "JOHN.<br>2. SUSAN" forms a single RTL run despite the <br>, the "2" goes to the right of SUSAN. (Please note that wrapping the "JOHN" and "SUSAN" in separate dir=rtl spans, i.e. "<span dir=rtl>JOHN</span>.<br>2. <span dir=rtl>SUSAN</span>", does not make any difference.)

Although this LTR example is somewhat contrived, the RTL equivalent is quite realistic because it is common for LTR brand names, acronyms, etc. to be used in RTL text:

```
1. IT IS IMPORTANT TO LEARN html.<br>
2. css IS IMPORTANT TOO.
```

which is rendered in Firefox and Opera as

```
                                                      html. NRAEL OT TNATROPMI SI TI .1
                                                            .OOT TNATROPMI SI 2. css
```

As a result, IE and WebKit treat <br> and embedded block elements as UBA paragraph breaks. Although this is not in conformance with the HTML 4 spec for <br>, the bidi separation it provides does seem to follow most users' expectations.

If IE and WebKit were to change their <br> behavior to conform to the current standard, many existing RTL HTML documents would be broken, especially given that they tend to be authored mostly with IE in mind.

While the bidi separation provided by treating <br> as a UBA paragraph separator is useful, the very strong nature of this separation (closing all open embedding levels) also creates problems. Being an inline element, <br> can be nested within an arbitrary number of other inline elements. If these inline ancestors have explicit dir attribute values of their own, should the <br> terminate their effects as UBA's definition of a paragraph separator says it should? That is what a newline in plain text does when it comes between an LRE or RLE and its matching PDF. So, should the second line in <div dir=rtl><span dir=ltr>1. hello!<br>2. goodbye!</span></div> be displayed as RTL? That would conform to the definition of a UBA paragraph break, but would go against the spirit of HTML. This is, in fact, what WebKit currently does (although it is now being treated as a bug).

To avoid this problem, IE apparently re-opens the directional embedding levels specified on ancestor elements via mark-up (dir attribute, <bdo> element) or CSS up to the closest ancestor block element after closing them at a <br> paragraph break. On the other hand, it does not reopen the directional embedding levels stemming from surrounding LRE/RLE/LRO/RLO and PDF characters.

Should the HTML specification of the bidi behavior of <br> be changed to this rather complicated definition in the hope that all browsers will be able to standardize around it?

And what about those rare uses of <br> when it is simply being used to wrap a line?

Proposed Solution

The <br> situation can be resolved by the simple expedient of defining <br> as having bdi=yes by default. (The bdi attribute is proposed in 1.1 above.) Thus, it would not form a UBA paragraph break, but would be treated, by default, as if it had either LRM or RLM characters around it, providing the required bidi separation to fix the examples above. Not being a paragraph break, it would not close any explicit embedding levels surrounding it. And when the author wants to use it just to wrap a line without adding bidi separation, <br bdi=no> will do the trick.

On the other hand, block elements like <div> and <hr> should be specified as introducing a UBA paragraph break between the text preceding and following them. After all, such text is said to be in "anonymous blocks (http://www.w3.org/TR/CSS21/visuren.html#anonymous-block-level) ".

2.2. newline characters should serve as bidi separators inside <pre>, <textarea>, and script dialog text.

Background

As in 2.1 above.

The Problem

IE and WebKit treat newline characters as a UBA paragraph break in <pre>, <textarea>, and the text displayed in dialogs by the page's scripts using functions such as Javascript's alert() and confirm(). Given that in these contexts newlines are expected to behave as they do in plain text, this would seem to be in accordance with the UBA. Firefox, however, treats newlines in all these contexts as UBA whitespace, while Opera treats them as UBA paragraph separators in <textarea> and dialog text, but as whitespace in <pre>. See 2.1 for examples where this makes a difference.

Proposed Solution

The HTML specification should state that the newline characters should be treated as UBA paragraph breaks in <textarea> and script dialog text. In <pre>, however, where the newlines can be wrapped in mark-up and thus have the same issues with being treated as UBA paragraphs as <br>, a newline should be treated as if it were a <br bdi> (see 2.1 above). Where lines are wrapped automatically in any of these contexts, the wrapping should be treated as UBA whitespace.

2.3. script dialog text should be displayed in the page's direction

Background

The W3C recommends (http://www.w3.org/TR/i18n-html-tech-bidi/#ri20030112.214820604) that in HTML, the direction of text be declared using the dir attribute, avoiding the use of Unicode formatting characters LRE, RLE, and PDF except where the dir attribute is inapplicable.

The Problem

One would expect that the page's direction set using <html dir=...> would apply to the text displayed in dialogs by the page's scripts using functions such as Javascript's alert() and confirm(). This used to be the case in IE6 and IE7, but no longer appears to be the case in IE8. It was never the case in any other major browser. The directional context IE8 and the major non-IE browsers use for dialog text is either the OS or the browser UI's default direction, which neither the server nor page scripts can even determine, let alone control.

Since a value displayed in the wrong direction can come out garbled, pages wind up having to wrap their RTL dialog text in RLE + PDF characters for correct display on LTR systems. On the other hand, pages dare not wrap their LTR dialog text in LRE + PDF characters for correct display on RTL systems, since most computers in the world are running an LTR OS without RTL script support turned on, and thus display LRE and PDF as rectangles. (This is not a concern in the case of RTL dialog text, since a system that does not have RTL script support will not display RTL text correctly anyway.) Furthermore, these formatting characters are little-known, lack named entities, and are generally undesirable in HTML documents.

Proposed Solution

The HTML specification should state that dialog text will be displayed in the <html> element's direction. Backward compatibility is not an issue here because this aspect of bidi behavior was never defined by the HTML standard, and browser behavior has not been consistent.

It is easy enough for a browser to implement this, since it knows the default directional context in which the text will be displayed by the underlying platform. If and only if this differs from the page's direction, the browser needs to wrap (each paragraph of) the dialog text in RLE + PDF in an RTL page and LRE + PDF in an LTR page.

2.4. <title> should support the dir attribute

Background

As in 2.3. above.

The Problem

One would expect that the page's direction set using <html dir=...> would apply to the page's <title>. Unfortunately, however, this is not the case in any major browser. The directional context all major browsers use for <title> is either the OS or the browser chrome's default direction, which neither the server nor page scripts can even determine, let alone control.

Nor does setting the dir attribute directly on the <title> element have any effect in any major browser.

Since a value displayed in the wrong direction can come out garbled, pages wind up having to wrap their RTL <title> in RLE + PDF characters. This has the same problems as with script dialog text, see 2.3. above.

Proposed Solution

The HTML specification should state that when a direction has been explicitly assigned to the <title> element, e.g. using the dir attribute on it, the <title>'s text will be displayed in that direction.

It is easy enough for a browser to implement this, since it knows the default directional context in which the text will be displayed. If and only if this differs from the desired direction, the browser needs to wrap the title text in RLE + PDF when RTL is desired and LRE + PDF when LTR is desired.

Open Issue: Should the HTML specification require the same for the direction a <title> inherits from an ancestor element? On the one hand, this is obviously useful - RTL documents will not have to declare their direction twice. On the other hand, this could break existing RTL documents that count on their title being displayed in LTR. Then again, they can't really count on it at present anyway, since it will be presented in RTL on an RTL OS / browser.

2.5. title and alt attribute text should be displayed in the element's direction

Background

As in 2.3 above.

The Problem

Currently all major browsers (IE, FF, Chrome, Safari, Opera) display the tooltips specified by a title or alt element attribute in the direction of the element to which it belongs, but this does not appear to be formally specified anywhere. Furthermore, this consensus seems fragile because in principle, the direction of an element and the text of its tooltip do not have to coincide. Here is a reasonable counterexample: an RTL web page displays an LTR address (e.g. for a location in Europe), with a tooltip on the address element saying "ADDRESS" in the page's language. The tooltip thus needs to be RTL while the element needs to be LTR.

Until recently, Chrome displayed tooltips in the OS / browser's default direction. When fixing this bug, the initial inclination was to apply only the page's direction, not the element's, due to the "in principle" consideration above.

Nevertheless, although counterexamples as given above can be found, tooltip text most usually does have the same direction as the element's text even where the element does have text, which is not very often. For such counterexamples, there is a simple workaround in the form of putting the tooltip on an extra element wrapping the original one.

Apparently not trusting browser behavior, the W3C suggests (http://www.w3.org/TR/i18n-html-tech-bidi/#tech-tooltips-etc) that tooltip direction may have to be set using LRE | RLE + PDF. This is actually quite difficult to do properly, since wrapping an LTR tooltip in LRE + PDF just in case the browser winds up displaying it in an RTL context will result in the LRE and PDF displaying as rectangles on LTR OS's without RTL support enabled, i.e. the vast majority of computers.

Proposed Solution

The HTML specification should explicitly state that title and alt attribute text will be displayed in the element's effective direction.

2.6. <option> should support the dir attribute and be displayed accordingly both in the dropdown and after being chosen

Background

As in 2.3 above.

The Problem

In a single <select>, the values of different options may have different directions. Currently, however, out of all major browsers, only FF supports the dir attribute on <option>, and does so poorly: once the value is chosen, it is displayed in the <select>'s direction.

IE and Opera display all options in the <select>'s direction.

Safari automatically estimates the direction of each option and displays it as such both in the dropdown and after it has been chosen regardless of the <select>'s direction (which is only used to place the down-arrow button and to align the values). This is all very nice, but direction estimation algorithms do make mistakes, so it would be good to be able to specify the actual dir value for a given <option> - and Safari does not support that.

Chrome does not support the dir attribute on <option> and is on its way to doing what Safari does.

As a result, the only practical way to specify <option> value direction is using LRE | RLE + PDF, which is cumbersome.

Proposed Solution

The HTML specification should state that setting an explicit direction on <option> should determine the way it is displayed in both the dropdown and after being chosen. Using auto-estimated direction is allowed when the <option> element does not have an explicitly specified direction.

2.7. <input type="text"> and <textarea> should support compatible "set direction" functionality

Background

Garbling by incorrect direction also applies to text being entered by the user in an input control. In fact, entering text of direction opposite to the input's declared direction is an unpleasant experience even if the full text does not wind up being garbled, due to the cursor and punctuation jumping around during data entry and difficulty in selecting text. All major browsers thus provide some way for the user to set the direction of each <input type="text"> and <textarea> element.

The Problem

Unfortunately, the way "set direction" functionality interacts with page scripts varies significantly between browsers, which makes it difficult to write scripts that are informed of the user's choice.

IE: Direction is set using keyboard shortcuts - CTRL + LEFT SHIFT for LTR and CTRL + RIGHT SHIFT for RTL. (These key combinations are also adopted for this purpose by most Microsoft products, e.g. Windows dialogs, notepad and Word.) They set the value of the element's dir attribute, which is then available to scripts. They trigger the onpropertychange event, at which time the dir value is already changed. They also trigger onkeyup, but before the dir value has been changed, so setTimeout(0) has to be used to get the updated dir value. They do not trigger onkeypress.

FF: Direction is set using the CTRL + SHIFT + X keyboard shortcut, which cycles through LTR and RTL. It does not set the value of the element's dir attribute, and is thus invisible to scripts.

Opera: same keyboard shortcuts as IE. They do not set the value of the element's dir attribute, and are thus invisible to scripts.

Chrome: same keyboard shortcuts as IE. They set the value of the element's dir attribute, which is then available to scripts. They trigger the onkeyup event, at which time the dir value is already changed. They do not trigger onkeypress or oninput. They also do not trigger onpropertychange, since this event exists only in IE.

Safari: Right-click on the <input> or <textarea> provides a "Set paragraph direction" submenu. Using "Set paragraph direction" sets the value of the element's dir attribute, which is then available to scripts. However, it does not trigger onkeyup, onkeypress, or oninput. It also doesn't trigger onpropertychange, since this event exists only in IE.

Proposed Solution

The HTML specification should state that some way to set the direction of <input type=text> and <textarea> elements should be exposed to the user, and using it will:

- Set the element's dir attribute value accordingly.
- Trigger oninput after the dir attribute has been set; even though no actual input took place, the user did change the recommended interpretation of the input already collected.
- Trigger onkeyup after the dir attribute has been set.
- Trigger onkeypress? **Open Issue**. But one way or the other, it should be specified.

Furthermore, it should be recommended that on an OS that has a widespread convention for setting direction (such as CTRL + LEFT SHIFT for LTR and CTRL + RIGHT SHIFT for RTL on Windows), the user agent will support that convention (although it may provide other methods too).

2.8. when an input value is remembered, its direction should be remembered too

Background

Some browsers implement auto-completion, a feature whereby values previously entered into an element like <input type=text> are remembered and under certain conditions presented to the user in a dropdown. When the user selects one of the items in the dropdown, this value is assigned to the element. At different times, the user may enter values of different direction for the same input. The direction of a value is set either directly by the user through a "set direction" command exposed by the browser (e.g. via keyboard shortcuts, see 2.7 above) or letting page scripts automatically set the input's dir attribute after estimating the direction of the value on the fly.

The Problem

Browsers do not remember the direction of previously-entered values. Some display them in the dropdown in the OS or browser default direction. Some display them in the input's current direction. Finally, some display each value in its own estimated direction. Each of these will result in some values being displayed incorrectly; even the last approach will sometimes fail because estimation algorithms do make mistakes, and this may not have been the direction originally set by the user or page scripts.

After the user chooses a value from the dropdown, the value is usually displayed in the input's current direction, which may or may not be correct for it.

Proposed Solution

The HTML specification should state that whenever a user agent stores a user-provided <input type=text> or <textarea> value for later use (such as auto-completion), it should also store the nominal direction value the element had when displaying this value. This may be the original direction of the element, or may have been set by the user for that value via keyboard shortcuts, or may have been set for that value by page scripts. If the user agent later displays the value in an auto-completion dropdown, it should be displayed in its stored direction. If the value is assigned to an element, the element's dir value should be set to its stored direction.

2.9. the rendering of numbering or bullets in a list should be independent of the direction of

individual <li> elements

Background

The HTML specifications gives no indication of how the bullet or number of an <li> element should be displayed when its effective direction is the opposite of its parent's (e.g. <ol> or <ul>).

The Problem

In practice, different browsers do different things, and the effects vary depending on the list's alignment, and whether it is ordered or unordered:

Example 1: <ul dir=ltr><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

IE	Firefox, Opera and WebKit
* item a.	* item a.
* longer item b.	* longer item b.
.C METI LTR *	.C METI LTR

Example 2: <ol dir=ltr><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

IE	Firefox, Opera and WebKit
1. item a.	1. item a.
2. longer item b.	2. longer item b.
.C METI LTR .	.C METI LTR

Example 3: <ul dir=ltr style="text-align:left"><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

IE	Firefox and Opera	WebKit
* item a.	* item a.	* item a.
* longer item b.	* longer item b.	* longer item b.
.C METI LTR *	.C METI LTR *	.C METI LTR

Example 4: <ul dir=ltr style="text-align:right"><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

IE	Firefox and Opera	WebKit
* item a.	* item a.	* item a.
* longer item b.	* longer item b.	* longer item b.
.C METI LTR *	.C METI LTR	.C METI LTR

In our opinion, not only is browser behavior unacceptably incompatible and inconsistent, but none of the above provides a usable display of opposite-direction list items.

Proposed Solution

In our opinion, the bullets/numbers of all items, regardless of their direction, should always:

- Be present.
- Line up.

This can be easily achieved by specifying that the rendering of list item bullets should depend on the direction of the list element (<ul>, <ol>, etc.), and not depend on the direction of the individual list items or the alignment of either the list or the list item. The outcome should look like this:

Example 1: <ul dir=ltr><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

```
 * item a.
 * longer item b.
 *                          .C METI LTR
```

Example 2: <ol dir=ltr><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

```
 1. item a.
 2. longer item b.
 3.                         .C METI LTR
```

Example 3: <ul dir=ltr style="text-align:left"><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

```
 * item a.
 * longer item b.
 * .C METI LTR
```

Example 4: <ul dir=ltr style="text-align:right"><li>item a.</li><li> longer item b.</li><li dir=rtl>RTL ITEM C.</li></ul>

```
 *                             item a.
 *                      longer item b.
 *                         .C METI LTR
```

Appendix A: Document History

A summary of the significant changes that have been made since the preceding version of this document (http://docs.google.com/Doc?docid=0AWd2A0jMoVOMZGQ2ZjU4NnRfMTlkZzRwa3FxYw) :

A1. All Examples are now given in "Pseudo Bidi" English (uppercase English for RTL characters, lowercase for LTR characters) instead of using an actual RTL script.

A2. "Part 1: Standardizing Bidi Aspects of Existing HTML Features" is now Part 2.

A3. "New HTML Features" has been added as Part 1.

A4. Section "1.1. <br>, <hr>, and embedded block elements should "reset" bidi state" (now section 2.1) has been rewritten, describing and taking into account various difficulties with defining <br> as a UBA paragraph break. Instead, we now propose that the new bdi attribute (new section "2.1. support bidi isolation of inline element content") should have the value "yes" by default for the <br> element, giving a weaker but sufficient amount of bidi separation. To have <br> treated as whitespace as in the current standard, one would use <br bdi=no>. We still propose that embedded block elements (including <hr>) should be treated as UBA paragraph breaks.

A5. Section "1.2. newline and other line-breaking characters should "reset" bidi state in <textarea>, <pre> and script dialog text" (now section 2.2): although we still propose that newline characters be treated as UBA paragraph breaks in <textarea> and script dialog text, we now propose that in <pre> they be treated the same as <br bdi=yes>. This avoids difficulties when the newline appears inside an inline element with an explicit direction. Such difficulties do not exist for <textarea> and script dialog script, when can not contain markup.

A6. Section "1.3. <title> and script dialogs should use the page's directionality" has been split into sections 2.3 and 2.4 covering script dialogs and <title> separately. The proposal that <title> text be displayed in the direction inherited from its ancestors, e.g. the <html> element, is now an open issue because of backwards compatibility concerns. We still require that when a direction has been specified to the <title> itself, e.g. <title dir=rtl>, it must be displayed as such.

A7. Section "1.7. auto-completion should remember and use the directionality of each value" (now section 2.8) has been somewhat broadened to cover any time that the browser stores an input value for later use, not specifically auto-completion.

A8. Section "2.9. the rendering of numbering or bullets in a list should be independent of the direction of individual <li> elements" has been added.

Retrieved from "http://www.w3.org/International/wiki/BidiProposal"

- This page was last modified 15:44, 29 January 2010.