micro tutorial: what you need to know about the bidi algorithm and inline markup

nearby: **ishida home** - **ishida writings** - **script tutorial** - **w3c i18n** - **unicode**

alternate: **PDF version**

This tutorial first describes some of the basic principles underlying how the Unicode bidirectional algorithm works. Then it looks at some of the more common scenarios where the bidi algorithm requires help through the addition of markup or control codes.

Although we try to take a markup independent view here, most of the examples will use XHTML, since it is widely recognisable. For advice relative to a specific markup language see the sidebar.

definitions

Bidirectional text is commonplace right-to-left scripts such as Arabic, Hebrew, Syriac, and Thaana. Numerous different languages are written with these scripts.

Any embedded text from a left-to-right script and all numbers run *left-to-right* within the general right-to-left flow. Of course, the English text on this page also contains bidirectional text where it includes Arabic and Hebrew examples.

The following example illustrates these switches in directionality.

got bidi support?

אום, W3C العربية

הבינאום, W3C العربية

If the above two lines show more than font differences your browser won't display the Arabic and Hebrew examples correctly. Read the PDF version instead.

useful links

faqs

guidelines

unicode stuff

unicode & xml

```
Example
```

We will use the term **bidi** to mean 'bidirectional'. We will also use **RTL** for 'right-to-left' and **LTR** for 'left-to-right'.

visual vs. logical order

The bidi algorithm works on logically ordered text. If you use visual ordering there is no point reading further (except that it could make your life a whole lot easier).

Visual ordering of text was a common way of representing Hebrew in old user agents that didn't support the Unicode bidi algorithm. It still persists to a degree today, out of habit. Characters making up the text were stored in the source code in the same order you would expect to see them displayed.

For example, take some Hebrew text with mixed directionality.

פעילות הבינאום, W3C

If you looked at the characters in memory, one by one, you would see the following for visually ordered text. Hebrew text would have to be typed backwards.

```
W3C, פעילות הבינאום
```

Visual ordering isn't really appropriate for Arabic, since the Arabic letters are all joined up and this behaviour is not likely to work correctly when the characters are ordered visually.

The biggest issue with visual ordering of flowing text is that it requires the author to disable any line wrapping and to explicitly right-align text in paragraphs and table cells. It also requires the order of table columns to be manually reversed when translating to another language. Here is an example written in XHTML:

```
<p style="text-align:right">
W3C מחליפה את שירותי הארחה באירופה מ-INRIA, <br/>
הממוקמת בצרפת, ל-ERCIM. השינוי מאפשר ל-W3C <br/>
להעמיק את קשרי המחקר ברחבי אירופה, תוך שמירה <br />
על הקשר ההיסטורי החזק עם INRIA, אחד ממייסדי <br />
ERCIM. השינוי יתבצע ב 1 לינואר 2003.
</p>
```

The result is very fragile code that is difficult to maintain. For example, if you wanted to add a few words on the second line of this paragraph, you would have to move text from the end of that line to the line that followed it, and repeat that process for every remaining line in the paragraph.

Logical ordering is a much better approach. In this approach text is stored in memory in the order in which it would normally be typed (and usually pronounced). Thus for the example above the characters in memory would be stored as follows:

```
פעילות הבינאום, W3C
```

The Unicode bidirectional algorithm would then produce the correct visual display shown at the beginning of this section.

how the bidi algorithm works

Here we introduce some important basic concepts. If it seems boring, stick with it because without understanding this stuff properly you'll get lost when you need to write marked up bidi text.

directional context

The result of the bidirectional algorithm will depend on the overall directional context of the paragraph, block or page in which it is applied.

In XHTML that context would normally be expressed by adding nothing to the `html` tag, in which case the default directionality is LTR, or adding `dir="rtl"` to the `html` tag, in which case all the elements in the document will inherit a context of RTL. The context may be changed later by using the `dir` attribute on the relevant block element.

The reason this is important will become clear in a moment. First we must explore some additional concepts.

directional typing of characters

Each character in Unicode has an associated directional property. Most letters are **strongly typed** as LTR. Letters from bidirectional scripts are strongly typed as RTL.

Spaces and punctuation do not have LTR and RLT forms in Unicode, but may be used in either type of script. They are therefore classed as **neutral**.

We already know that a sequence of Latin characters is rendered (ie. displayed) one after the other from left to right (we can see that on this page). On the other hand, the bidi algorithm will render a sequence of strongly typed RTL characters one after the other from right to left.

When text with different directionality is mixed inline, the bidi algorithm renders each sequence of characters with the same directionality as a separate **directional run**. So in the following example there are three directional runs:

bahrain مصر kuwait

Directional runs are ordered according to the overall context. In the example above, that has an overall context of LTR, you would read 'bahrain', then 'مصر ' (RTL), then 'kuwait'. Note that you don't need any markup or styling to make this happen.

neutral characters

This is where things begin to get interesting. When the bidi algorithm encounters characters with neutral directional properties (such as spaces and punctuation) it works out how to handle them by looking at the surrounding characters.

A neutral character between two strongly typed RTL characters will be treated as a RTL character itself, and will have the effect of extending the directional run. This is why the three arabic words in this LTR phrase are read from right to left. (The first Arabic word you read is مفتاح then معايير then الويب.)

> The title is مفتاح معايير الويب in Arabic.

Note that you still don't need any markup or styling for this. There are still only three directional runs here.

The really interesting part comes when a space or punctuation falls between two strongly typed characters with *different* directionality. In such a case they will be treated as if they have the directionality of the *overall paragraph or context*. Even if there are several neutral characters between the two different strongly typed characters, they will all be treated in the same way.

This has the effect of creating a boundary between directional runs.

where the algorithm needs help

The bidi algorithm will handle text perfectly well in most situtations, and typically no special markup or other device is needed other than to set the overall direction for the document. You would be very lucky, however, if you got off that easily all the time.

neutrals that get misplaced

Let's introduce some punctuation to the Arabic phrase in the last example. By default we will see the following:

> The title is "مفتاح معايير الويب!" in Arabic.

The quotation marks are OK, but the exclamation mark is in the wrong position. It should appear to the left of the Arabic text.

Given our understanding of the bidi algorithm we can easily understand why this happened. Because the exclamation mark was typed in between the last RTL letter 'ب' (on the left) and the LTR letter 'i' (of the word 'in') its directionality is determined by the overall context of the paragraph (here LTR). Note that it makes no difference that there are actually two punctuation characters and a space in this position - they are all neutrals and so are all affected the same way. Because the exclamation mark is seen as LTR it joins the directional run that includes the text 'in Arabic'.

To fix this it would be useful to be able to type an invisible, strongly-typed RTL character after the exclamation mark. That way the exclamation mark would be interpreted as RTL and join the Arabic directional run.

It just so happens that there is such a character - the Unicode character U+200F, called the RIGHT-TO-LEFT MARK (RLM). There is a similar character, U+200E, called the LEFT-TO-RIGHT MARK (LRM).

Because the character is invisible it is typically better to actually type in a numeric character reference (`&#x200F;`) or, if available, a character entity (such as `&rlm;` in XHTML). In the following example `&rlm;` has been added after the exclamation mark and the result looks fine:

> The title is "مفتاح معايير الويب!" in Arabic.
>
> The title is "مفتاح معايير الويب!‏" in Arabic.

neutrals with an identity crisis

In our next example the list order is incorrect because the first two Arabic words should be reversed and the intervening comma, which is part of the English text, should appear immediately to the right of the first word.

> The names of these states in Arabic are مصر, البحرين and الكويت respectively.

The reason for the failure is that, with a strongly typed right-to-left (RTL) character on either side, the bidirectional algorithm sees the neutral comma as part of the Arabic text. In fact it is part of the English text, and should mark the boundary of two directional runs in Arabic. We actually find ourselves in the opposite situation to that described in the previous section.

The solution is to use another invisible Unicode character, this time the LEFT-TO-RIGHT MARK, next to the comma. This puts our neutral punctuation between strongly typed RTL and LTR characters and forces it to take on the directionality of the overall context, which is the English LTR flow. This breaks the Arabic words into two separate directional runs, which are ordered LTR in accordance with the prevailing direction of the paragraph.

> The names of these states in Arabic are مصر,‎ البحرين and الكويت, respectively.

Again, it might be better to use an NCR (`&#x200E;`) or a character entity (such as `&lrm;`) if available.

once again, together

The examples we have used so far have been English and LTR based. The same principles apply for RTL text in languages such as Hebrew and Arabic. Lets see one more example.

In the following Hebrew text the parentheses look a real mess.

(W3C (World Wide Web Consortium
.ERCIM - מעביר את שירותי הארחה באירופה ל

In addition, the Hebrew-speaking writer of this text really wants the word 'W3C' to be to the right, outside the parentheses as shown here:

(World Wide Web Consortium) W3C
.ERCIM - מעביר את שירותי הארחה באירופה ל

It looks like this is going to be incredibly complicated to sort out, but actually the solution is trivial. Just insert an RLM after 'W3C' and you're done. It's really that simple!

If you're not convinced, here's the explanation. The RLM after 'W3C' makes this piece of LTR text a separate directional flow. Remember that flows will be ordered right-to-left because this is the overall paragraph direction. Since the 'W3C' text was typed in first, that now appears farthest to the right. The paren is now between strongly typed RTL and LTR characters, and so also takes on the directionality of the overall paragraph, RTL. So that comes next. Then comes the unbroken LTR directional flow which is the stuff inside the parens.

(The changed direction of the right-most parenthesis comes automatically. The glyph used for these 'mirrored characters' changes according to its directionality. It's still the same character.)

nesting directional runs

The unicode bidi algorithm and the directional markers work quite well when there is only a single level of mixed text. If you have a situation where there are two or more nested levels of directional text you will need a different solution. Here is an example of incorrectly ordered text:

> The title says "פעילות הבינאום, W3C" in Hebrew.

The order of the two Hebrew words is correct, but the text 'W3C' should appear on the left hand side of the quotation and the comma should appear between the Hebrew text and 'W3C'. In other words, the desired result is:

> The title says "W3C ,פעילות הבינאום" in Hebrew.

The problem arises because the directional flows are being ordered according to the LTR context of the paragraph. Inside the Hebrew quotation, however, the correct default ordering should be RTL.

To resolve this problem we need to open a new **embedding level**. In XHTML this would be done by enclosing the quotation in markup and assigning it a directionality of RTL using the `dir` attribute.

```
The title says "<span
dir="rtl">W3C ,פעילות הבינאום</span>"
in Hebrew.
```

In markup languages other than XHTML/HTML you may find a similar attribute to which you can apply styling to achieve the correct effect. If you don't have such an attribute you may have to resort to individually styling the appropriate inline markup, but it would probably be better to lobby your markup developer to provide you with one.

There are Unicode control characters you could use to achieve the same result, but because they create states with invisible boundaries this is not recommended.

overriding the algorithm

There may be occasions where you don't want the bidi algorithm to do its reordering work at all. In these cases you need some additional markup to surround the text you want left unordered.

In XHMTL 1.0 this is achieved using the inline `bdo` element. In XHTML 2 it will be implemented as a value of `rlo` or `lro` on the direction attribute, enabling it to be applied to any element. Again, there are Unicode control characters you could use to achieve the same result, but because they create states with invisible boundaries this is not recommended.

The examples above that show the characters as ordered in memory use the bdo tag to achieve that effect. For example, to show the underlying sequence of characters for:

> פעילות הבינאום, W3C

We would use the following markup in XHTML 1.0:

```
<p><bdo dir="ltr">פעילות הבינאום,
W3C</bdo></p>
```

The result would be:

> םואניבה תוליעפ, W3C

numbers are a little special

Numbers in RTL scripts run left-to-right within the right-to-left flow, but they are handled a little differently than words by the bidi algorithm. They are said to have **weak directionality**. The following two examples illustrate this difference. Comparison of the two cases shows the words on either side of the fourth item in the sequence have been reversed. The only difference between the two sentences in memory is the use of '1234' versus 'four'.

one two ثلاثة four خمسة

one two ثلاثة 1234 خمسة

In the first example the letters in the word 'four' are strongly typed and therefore break the two Arabic words into separate directional runs, ordered left to right as per the paragraph context.

In the second example, the weakly typed number '1234' is seen as part of the Arabic text, so the two Arabic words are treated as part of the same directional run - even though the sequence of digits runs LTR, as in memory.

This only happens in RTL text.

Sounds complicated? Don't worry, usually the bidi algorithm will just take care of things for you. I really only included this section for those who notice the difference and wonder what's going on.

Note also that alongside a number certain otherwise neutral characters, such as currency symbols, will be treated as part of the number rather than a neutral .

Last modified 2003/07/23 by Richard Ishida.

```
<p style="text-align:right">
,INRIA-מ באירופה הארחה שירותי את מחליפה W3C <br/>hkkjhkjhkjhkhkj  khkjh kjh k
W3C-ל מאפשר השינוי .ERCIM-ל ,בצרפת הממוקמת< br/>
שמירה תוך ,אירופה ברחבי המחקר קשרי את להעמיק<br />
ממייסדי אחד ,INRIA עם החזק ההיסטורי הקשר על<br />
.2003 לינואר 1 ב יתבצע השינוי .ERCIM
</p>
```

```
<p style="text-align:right"> ,INRIA-מ באירופה הארחה שירותי את מחליפה W3
            khkjh kjh k W3C-ל מאפשר השינוי .ERCIM-ל ,בצרפת הממוקמת<
            להעמיק<br /> ממייסדי אחד ,INRIA עם החזק ההיסטורי הקשר ול
            .ERCIM </p>
```