Practical Tips for Designing
International Web Pages

Richard Ishida
W3C Internationalization Activity Lead

Outline

Text size
Navigating to localized pages
Separating content & presentation
Composite messages

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 4

Text size

W3C

Text size

slide 5

This slide shows an example of how text can expand (and shrink) when translated (here from English).

If you are using graphic text, beware! Not only is it more difficult to translate graphic text, but graphic text is often squeezed into small spaces, making it difficult to replicate the source in another language.

Here Flickr has resorted to using just icons with mouse-overs in the translated UIs to get around the expansion issues they would have otherwise had.

Here is another potential issue related to the length of text in translation. Let's assume that we want to implement a **fixed-width** box on our page. The text can expand downwards, but not sideways.

Let's continue to assume a situation where text appears in a fixed width box. We have used a table to apply form labels to the left side of the form entry field to which they apply. The table framework should allow text to expand vertically without destroying the relationship with the input controls. Our initial source text is in English.

Text size
Form field label positioning

W3C

Global settings

Interface language
English

Search language
English

Number of results 10

Save preferences

Acuan Umum

Bahasa Pengantar di Antar Muka
Ingriss

Bahasa Pengantar untuk Penelusuran
Ingriss

Jumlah Hasil Penelusuran 10

Simpan Acuan

Allgemeine Voreinstellungen

Sprache der Benutzeroberfläche
Englisch

Suchsprache
Englisch

Anzahl der Ergebnisse 10

Einstellungen speichern

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 11

You may want to consider avoiding table cells in such constrained circumstances. This slide shows how the text would look if the input fields were just in a paragraph with the label text.

All the boxes now look fine, and although there is a very slight increase in vertical height overall, we have removed the problems seen with the Malay and German text on the previous slide.

Let's note that this is down to the way the page is **designed/developed**, not the way it is localized. That's a fundamental message of this presentation. Internationalization during design and development removes significant barriers to deploying your content globally.

Let's now assume that we want a background with a nice gradient behind the title of a fixed width box, and that the background has a line across the bottom.

(This example is in Spanish, and has the title 'Interface Language', and a list of radio buttons to select a language.)

Text size
Graphics in fixed width boxes

W3C

Bahasa Pengantar
untuk Penelusuran
- Arab
- Inggris
- Finlandia
- Perancis
- Italia
- Jepang
- Rusia
- Spanyol
- Swedia

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 13

As our text expands during translation into Malay, the title occupies two lines. Unfortunately the graphic used for the gradient background is only one line deep, and things now begin to look a mess.

Text size
Graphics in fixed width boxes

W3C

Bahasa Pengantar untuk Penelusuran

- Arab
- Inggris
- Finlandia
- Perancis
- Italia
- Jepang
- Rusia
- Spanyol
- Swedia

```
.box h3 {
    margin: 0;
    padding: 6px 8px 4px 10px;
    font-size: 130%;
    color: #333;
    background: url(img/h3-bg.gif)
                no-repeat top left;
    border-bottom: 1px solid #E0CFAB;
}
```
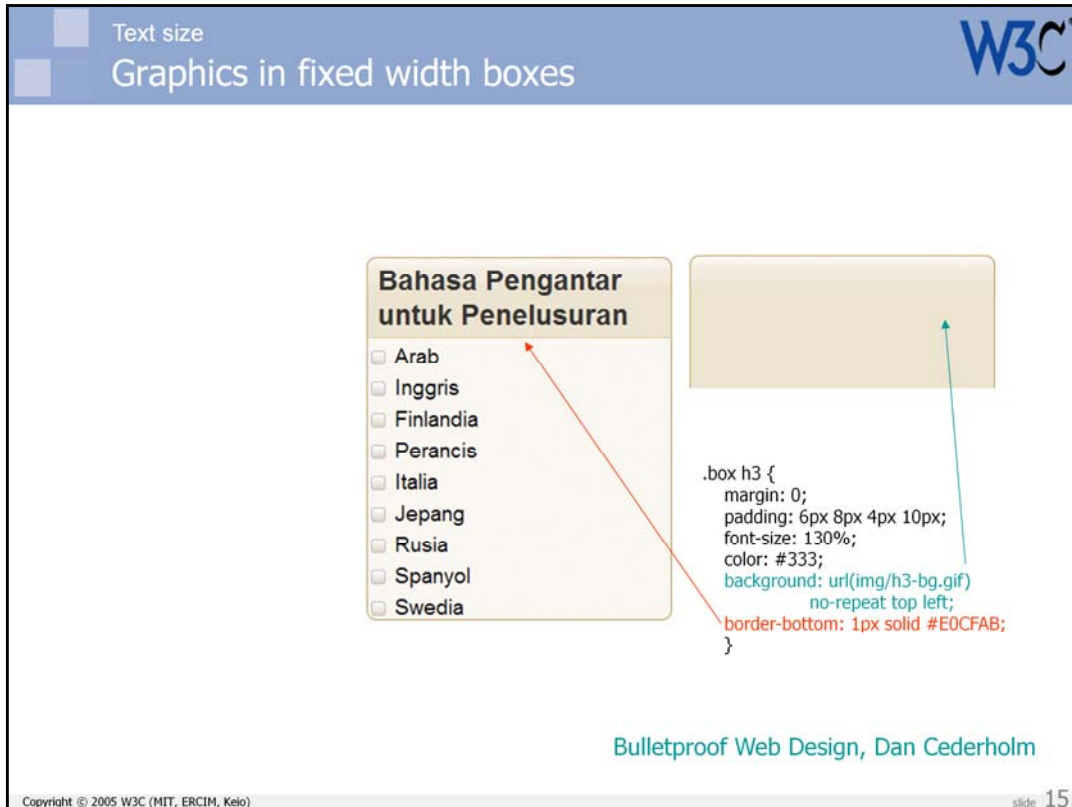
Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 14

A way to approach this issue is to use a graphic that is three or four lines deep behind the title. By attaching the graphic using the CSS background property, only the amount needed to view the title will actually be shown.

Text size
Graphics in fixed width boxes

W3C

Bahasa Pengantar untuk Penelusuran

- Arab
- Inggris
- Finlandia
- Perancis
- Italia
- Jepang
- Rusia
- Spanyol
- Swedia

```
.box h3 {
    margin: 0;
    padding: 6px 8px 4px 10px;
    font-size: 130%;
    color: #333;
    background: url(img/h3-bg.gif)
                no-repeat top left;
    border-bottom: 1px solid #E0CFAB;
}
```

Bulletproof Web Design, Dan Cederholm

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 15

To get the line to appear in the right place, we simply create it as the bottom border of the heading.

This example uses a technique (and the exact same code and graphic) described in Dan Cederholm's book, Bulletproof Web Design (although the text is borrowed from Google's language preferences). This is significant! Dan is not writing about internationalization per se – he is more concerned with people pumping up the text size for accessibility reasons. It just so happens, however, that the same approach helps with localizability.

This is an example that you don't necessarily have to learn new information to deal with internationalization issues – just following existing best practices can be the key in many cases. Note again, however, that we are still talking about the **design and development** of content – not about work that the localizers will do!

Dan's book contains several other recommendations that will benefit internationalization.

When creating tabs, don't draw the background as a single graphic.

Instead use something like Doug Bowman's 'sliding doors' technique (http://alistapart.com/articles/slidingdoors/). This allows the graphic part of the tab to expand with the text.

Note, again, that this also aids accessibility. The slide shows the result of increasing text size on the tabs.

One other thing to note, however, is that you should leave some space for expansion to the right of the tabs (or left in a right-to-left script). You may otherwise need to design the CSS so that the tabs can be wrapped and still look good.

In the W3C's i18n articles we use a box with rounded corners.

This will be much easier to do when CSS3's rounded corners properties are available. In the meantime, people are using background graphics.

This box can use techniques similar to those described earlier to allow the box to contract or expand (up to a certain point) horizontally as well as vertically, to fit the text.

This shows the graphics used.

One potential issue here is that some languages have very long words, which may not fit within the specified max-width.

An ideal solution to this would be to get the translator to insert soft-hyphens at appropriate places in the word. The word would then wrap to fit the box.

(Although this works in the latest versions of all major browsers, the soft hyphen was only implemented in Mozilla browsers from Firefox version 3.)
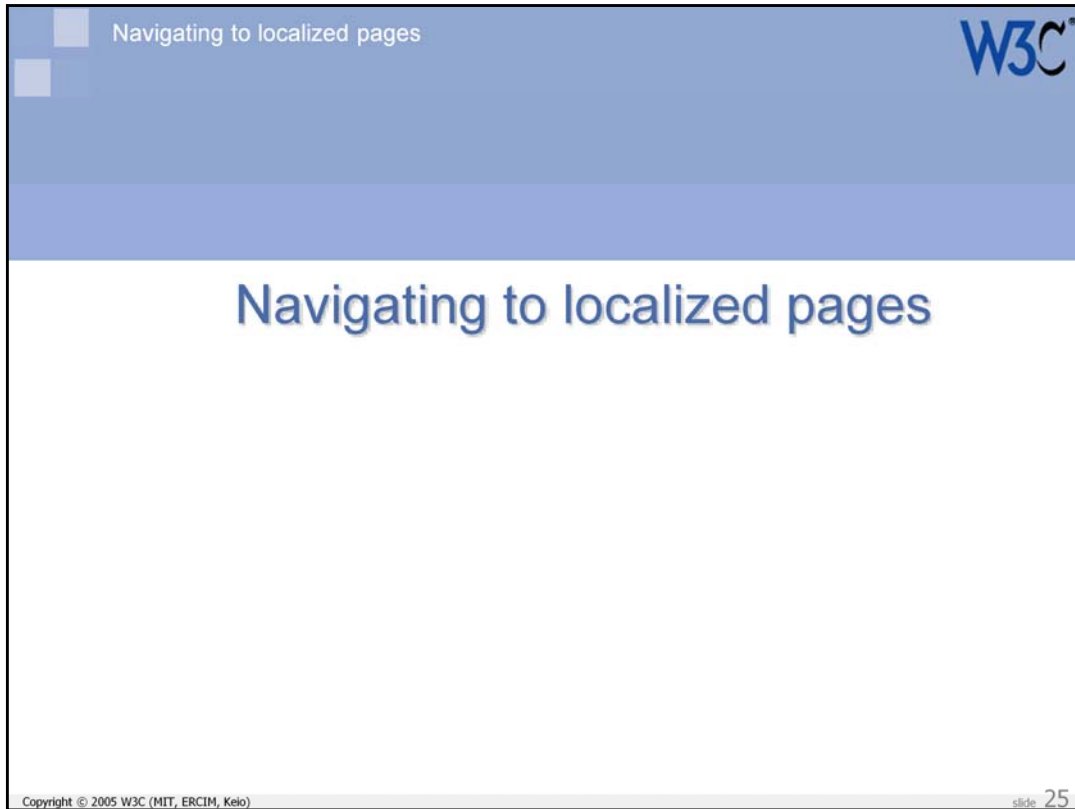
What this means to me
Some lessons from this section

W3C

- Think about how to cope with text that expands in translation

- Avoid placing labels to the side of form entry fields, icons, etc, if you can

- For text backgrounds use large images and 'sliding door' effects to allow coverage of text whatever the size

- Leave expansion space at the end of lines of tabs

- Consider how to deal with long compound nouns: don't make the column too narrow; warn translators

- Look out for and lobby the W3C CSS Working Group to publish rounded borders and multiple backgrounds parts of CSS3

- Lobby browser developers to support internationalization features such as the soft-hyphen character

http://www.w3.org/International/questions/qa-resizing-backgrounds

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 24

This slide summarizes some of the practical takeaways from this presentation.

Navigating to localized pages

Note, in passing, an issue related to the Google text I used in an earlier example. The dialogue allowed you to select a different language for the user interface from a pull-down list, presumably assuming that your reason for changing was that you couldn't read the current language.

The issue for me is that the names of all the languages are in the language of the current page. Let's assume, for example, that a curious person wanted to see what the interface looked like in Persian, so they selected that language from the list and clicked on the 'Save Preferences' button.

Assuming that they would be able to find their way back to the appropriate dialogue box to get back to English (which would require them to remember which link to hit on the thankfully uncluttered Persian Google home page), that they can remember which is the required select list, and that they can do so in spite of the mirror-imaging of the page when using Arabic script, they would then be faced with what you see on the next slide.

Note that the names of languages are all in Persian, and are sorted by Persian rules.

Which selection would get you back to English ??

(Hint: if you want to explore like this, use a different tab or window for your explorations, and leave the original dialogue available in another for when you want to reset to your current language.)

Of course, the point is really that a Persian person taken to the English site may have as much trouble finding their way to the appropriate user interface language as the curious explorer does in getting back. In my opinion it would help a great deal to write each language name in its own script and language. You can read more about that in the W3C Internationalization article at http://www.w3.org/International/questions/qa-navigation-select .

Ideally you would display individual links to translated pages on the page itself.

These should be in a prominent place so that they can be easily found.

I recommend that you always provide these links, regardless of whether you try to automatically guide the user to the right page using content negotiation, ip lookup, etc.

If you have a large number of options, one way of allowing users to access an alternative language or country site is to use pull-down lists (with the select element in HTML).

Here is an example. My personal preference would be to use the native translation of the country name first and put the English version in parentheses after. It can be useful, however, to have both as shown here (though inconsistently). My preference would also be to put the non-English first, and add the English in parentheses after (unlike this example).

A potential issue with this approach is that the user may not have fonts to cover all the characters, resulting in something like the boxes shown above.

You may feel that this is acceptable, since the English text alongside gives sufficient cues to the reader about what those boxes are about.

On the other hand, some people may feel that this is not acceptable.

This example shows an approach that could be used (although hasn't here) to provide what looks like a pull-down list using javascript and HTML. This produces a hybrid approach that would allow you to use graphics in the pull-down, so that all native language text could be seen as intended.

An alternative approach is to route the user to a selection page like this. On such a page, which is in HTML, you can also resolve the missing font issue by presenting the text in graphic form.

I would not normally recommend using text in graphics, but there shouldn't be a big issue here, given that this text doesn't need translating and is perhaps less likely to be searched for.

One thing to bear in mind, however, is that you should route a user to an alternative language and (often) country site that doesn't remove them from their current context, ie. don't force them up to the home page if they entered the site from a link. If they go to a selection page like this, you should return them to the same page (if such exists) in the localized site.

In this example, the link to the selection page (though it could also be the location of a pull-down list) is not easy to spot from the text. The designer has, however, used a graphic – ie. a non-verbal cue – to help users locate the appropriate place.

If you do take users to a selection page, don't be too chatty on that page.
If they are going to it because they don't speak the current language, it
won't be a lot of help.

One issue that is difficult to resolve is how to order the list of languages or countries. You could use the Unicode Collation Algorithm, sort by population size or GNP, sort by BCP 47 language tags, etc. to be neutral, although many users would possibly not understand the rationale for such ordering.

For **country-specific** links it may help to use maps of regions to narrow the search to a list that may be easier to scan.

Don't use flags for language-specific selections.

This slide summarizes some of the practical takeaways from this presentation.

Separating content & presentation

39

W3C

Separating content & presentation

slide 39

You should try to remove all presentational constructs from your content. For example, use of <i> tags shows that you are assuming that the text will be italicized. Because ideographic text doesn't support italicizations well in small font sizes, you could be causing problems for localization.

The basic recommendation is to avoid presentationally-oriented markup, such as <font> or <i> tags, and inline style attributes. A key reason for this is that when you want to change the styling, it is too expensive to have to trawl through the HTML changing markup in every page.

A better way is to use semantic markup (ie. meaning related markup) and relegate all styling considerations to a separate style sheet.

In the HTML to the right of this slide there is no presentational information in the HTML – which is as it should be. Browsers have default styling built in, which would produce something like the layout on the left in Firefox.

To the right is some CSS code in a style sheet that can be used to apply styling to the HTML.

The CSS style sheet is a single file that can be used by many markup files. Any changes to styling are therefore made in a single location.

Simple changes to the style sheet can be made to produce all the of layouts shown here, without any need to touch the HTML code at all.

Remember, also, that the Mobile Web is becoming increasingly important these days – and may be especially so in developing countries in the future. This means that content needs to be adapted to fit on handheld devices with smaller screens.

Pages tailored for mobile devices tend to use different styling for the same content.

You should not make assumptions, when creating content, that you know what it will look like when finally displayed. These days, it may well be displayed in a number of different formats.

International Web Sites

When it comes to aggregated data, different clients may use different styling for the same content.

Localization may also involve a need to restyle parts of the content.

Here are some ways in which typographic differences may appear between language versions of the same content, based on differences between English and Japanese. It is much easier to apply each of these typographic differences if you can do so via a CSS style sheet, rather than searching through the HTML or script code.

Separating content & presentation
Style issues for Japanese

W3C

- emphasis techniques may be different because of problems to support bold and italics in small characters on-screen

- different fonts, and possibly font-size, need to be used for font-family assignments

- line heights, paragraph spacing may need to be adapted, since characters may be more complicated

Leading the Web to its full potential...

การนำไปสู่ Webที่เต็มไปด้วยศักยภาพ

Copyright © 2005 W3C (MIT, ERCIM, Keio)

slide 47

International Web Sites

Richard Ishida   48   Version: 16 June 2009

Separating content & presentation
Style issues for Japanese

W3C

- emphasis techniques may be different because of problems to support bold and italics in small characters on-screen

- different fonts, and possibly font-size, need to be used for font-family assignments

- line heights, paragraph spacing may need to be adapted, since characters may be more complicated

- wrapping rules may need adjustment

- conventions differ: no upper- vs. lower-case distinction, no distinction between proportional and mono-spaced fonts

Press the START button.

Copyright © 2005 W3C (MIT, ERCIM, Keio)　　　　　　　　　　　　　　　　slide 49

You should also consider separation of content and presentation when adding scripting. Let's suppose that we wanted to load some JavaScript after this basic test page has loaded which would automatically add a list of tests on the page to the top right corner. (We may actually want to add links to these tests, but I have resisted that temptation so that the following slides will contain the code examples.)

Separating content & presentation
Don't forget your scripting

W3C

```
function makeList () {
    var headings = document.getElementsByTagName('h2');
    var div = document.createElement('div');
        div.style.cssFloat = 'right';
        div.style.styleFloat = 'right';
        div.style.width = '20%';
        div.style.padding = '1em';
        div.style.border = '1px solid teal';
        div.style.backgroundColor = '#FFCC99';
        div.style.fontSize = '90%';
        div.style.margin = '1em';
        div.style.color = '#996633;';
    var title = div.appendChild(document.createElement('div'));
        title.style.fontWeight = 'bold';
    title.appendChild(document.createTextNode('Tests on this page:'));
    for (i=0;i<headings.length;i++) {
        var test = div.appendChild(document.createElement('p'));
        test.appendChild(document.createTextNode(headings[i].childNodes[0].data));
        }
    body = document.getElementsByTagName('body')[0];
    body.insertBefore( div, body.firstChild );
    }
```

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 51

Here is a simple function that could be used to add the required text. It creates a div, gets a list of level two headings, and adds the text of the headings to the list.

Separating content & presentation
Don't forget your scripting
W3C

```
function makeList () {
    var headings = document.getElementsByTagName('h2');
    var div = document.createElement('div');
        div.style.cssFloat = 'right';
        div.style.styleFloat = 'right';
        div.style.width = '20%';
        div.style.padding = '1em';
        div.style.border = '1px solid teal';
        div.style.backgroundColor = '#FFCC99';
        div.style.fontSize = '90%';
        div.style.margin = '1em';
        div.style.color = '#996633;';
    var title = div.appendChild(document.createElement('div'));
        title.style.fontWeight = 'bold';
    title.appendChild(document.createTextNode('Tests on this page:'));
    for (i=0;i<headings.length;i++) {
        var test = div.appendChild(document.createElement('p'));
        test.appendChild(document.createTextNode(headings[i].childNodes[0].data));
        }
    body = document.getElementsByTagName('body')[0];
    body.insertBefore( div, body.firstChild );
    }
```

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 52

Note how we are adding style information directly to the DOM while running this script. This is really obvious in this example, since there is such a lot of it. It is particularly tempting to do this sort of thing if you just want to add a single style effect, such as bolding, to text.

This version of the same function shows a much better approach. We assign an id attribute to the box, then move all the styling information to a CSS file, referencing the markup via the id. This makes the code much cleaner and makes it easier to manage the styling.

Again, this technique is recommended as a standard best practice in Jeremy Keith's book Dom Scripting (which contains many other useful ideas along similar lines). It is another good example of how good web design benefits localization.

This slide summarizes some of the practical takeaways from this section.

This shows four different ways of writing one idea. In each case the order of 'words' and the number of 'words' is different.

Composite messages
Agreement & Concept mappings

W3C

On

Conectado	-os -a -as
Encendido	-os -a -as
Activado	-os -a -as

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 57

This slide shows how the English word 'On' can map to three different words in Spanish. And then there are the masculine, feminine and plural forms of agreement that change the shape of the word according to its context.

Composite messages
Agreement of Russian plurals

Number of pages	Russian word for 'page'
1	страни**ца**
2-4	страни**цы**
5-10	страниц
11-20 (irregular)	страниц
21	страни**ца**
22-24	страни**цы**
25-30	страниц
> 30	Repeat pattern of endings for 1 to 10.

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 58

In Russian there is a complex plural system. Apart from the irregular teens, the word endings are applied in a rotating way.

For this slide we imagine that the W3C Validator is altered slightly so that it tells you how many validation errors are in your file. It will do this using a 'composite message' whose parts are assembled using PHP code as the page is served.

Although we use PHP for these examples, the concepts can be applied to other scripting or coding environments.

In the German translation, the order of the two variables may need to be changed.

Typically translators have no access to the actual code, to avoid them introducing bugs into the page. Either the text is extracted or a translation tool masks the code.

Although we are fortunate that we were able to add words after the second variable, due to the English string containing a period, this still didn't produce the right result. The German reads "File 268 contains myFirst.html validation errors."

The reason is that the translation process didn't switch the order of the variables.

So next we try using a printf statement. This has the benefit that text and variable locators all sit within a single string, and the translator can access the items they want to reorder.

Unfortunately, this doesn't help, since PHP still replaces the variables in the string in the order of the variables cited in the following parameters to printf. This causes the 268 to be shown instead of the filename, by converting the integer value to a string. It is unable to find an integer value in the file name, and so presents us with the zero for the number of errors.

By embedding the variable names directly in the printf string, as shown in this slide, we finally achieve the desired result in German.

Nota bene: Successful, or at the very least, cost effective localization in this case is down to the **designer/developer** understanding the potential pitfalls of various approaches to coding. It is not the job of the localization vendor to get this right. It needs to be done as the initial content is created!

You should also be very careful of the assumption that 'This doesn't affect me, since we don't translate the content I develop.' I have seen many, many cases where the thing being developed was later so successful that people wanted to take it to other regions, only to find that they ran into major difficulties because of issues with the translatability of the code or content. It's best to just do it right from the start.

By the way, there is a way to produce the right effect while using the %d and %s variable markers in a PHP string, but it involves a slightly more complex syntax. This is shown in the above slide. The numeric markers refer to the relevant variable in the parameters that follow the string, even after reordering.

So now we know how to code this type of text in PHP... or do we?

Let's think back to our example of how plurality works in Russian, and we realize that we still have a problem for that language. We only have a single string and it can only be translated one way – yet the Russian requires three variants of the word ошибка, depending on the number that precedes it.

To deal with this, the Russian translator would probably resort to a completely different structure for the text, essentially equivalent to "File: X. Validation errors: Y". This approach requires only one form of ошибка in the invariable string. This is an example of what I call a 'topic-comment' composite message.

So we are beginning to see here that there are two distinct types of composite message. The first is based on a sentence-like approach, and the invariant string can be difficult to translate in some circumstances because of the need for agreement or different word mappings.

In the example above, 'The' should be translated 'el', 'la', or 'las' in Spanish, depending on what word follows it. Also the word 'on' should be translated using three different Spanish words (with different endings).

The other approach to designing composite messages is what I like to call the 'topic-comment' approach: you state a topic, then you say something about it.

This approach works much better for the previous slide, since you each comment you associate with a topic can use a different word with the appropriate word endings.

There is a little more to this theory of composite messages than we have mentioned so far, but you can get more information from the W3C Internationalization site at the following URI: http://www.w3.org/International/articles/composite-messages/ .

I should, however, mention just one other point. Many designers/developers looking at the English topic-comment arrangement on the previous slide might think to themselves that they could save a little bandwidth by reducing all those instances of the word 'On' to a single string that is used for all comments, ie. they want to re-use strings.

Tempting as this idea may appear, it will unfortunately introduce insurmountable problems for translation, since the comment is likely to require different agreement forms at the least, and possibly different words altogether, depending on the context.

This slide shows an example of how such a problem may come about by returning the same text from a function for each comment.

Note that I do not want to rule out string re-use altogether – there are situations where it is a sensible approach. But re-use must not occur across different contexts. For more information about this, see the W3C Internationalization article at http://www.w3.org/International/articles/text-reuse/ .

What this means to me
Some lessons from this section

- Use topic-comment arrangements for composite messages, if you can, rather than sentential arrangements

- Use coding approaches that allow syntactic flexibility when creating composite messages

- Be careful about reuse of text strings – ensure that the context of reuse is always the same

Copyright © 2005 W3C (MIT, ERCIM, Keio)   slide 72

This slide summarizes some of the practical takeaways from this section.

The presentation is not designed to give you a thorough overview of potential internationalization and localization issues. It aims to provide you with a few practical takeaways, but more importantly it aims to get you thinking about what internationalization is all about – to take designers out of their comfort zone, and help them realize that if you want your content to wow people outside your own culture and language, you need to build in certain flexibilities and adopt certain approaches during the design and development – not as an afterthought. Otherwise you are likely to be creating substantial barriers for worldwide use.

The presentation also aims to show that, although using Unicode is an extremely good start to making your stuff world-ready, using a Unicode encoding such as UTF-8 throughout your content, scripts and databases is only a start. You need to worry about whether translators will be able to adapt your stuff linguistically, but you also need to also consider whether graphics and design are going to be culturally appropriate or can be adapted, and whether your approaches and methodologies fit with those of your target users.

The W3C is trying to provide useful advice at
http://www.w3.org/International/

We could always do with help and support for this.