

Translation of OpenEpi to Non English Languages

To translate OpenEpi to a non English language (for the first time):

1. Download and unzip the entire website from www.openepi.com, using the Downloads entry on the menu.
2. If the original programs are in English, copy the file EN.js to a similar file with your own language symbol. Use two-letter Internet codes such as ES for Spanish and IT for Italian. For Spanish, copy EN.js to ES.js . Each phrase is presented in English, and the translation is placed on the same line between the empty quotation marks that are supplied. Phrases for all programs and from the Exercise documentation are contained in EN.js. A program called UpdateTranslation.hta is provided for updating the file when programs are revised or added. Translation of these files will provide the core or minimal system, complete with Demo and Example files for each module.
3. Longer documents are in the Documentation directory in HTML format. If you wish to translate these, create an anchor or bookmark for the translation in the same file. The name of the anchor or bookmark should be the Internet symbol for the new language, ES or IT, for example. The first block is assumed to be the original language (for example, English or EN), but subsequent blocks should begin with anchors (bookmarks) called IT for Italian and ES for Spanish. The programs will automatically seek out the correct language when the file is requested. If they are absent, the original language (beginning of the file) will be displayed.

It should not be necessary to understand or work with the JavaScript code or HTML of the programs to do a translation.

How Translation Works

OpenEpi translation works as follows:

- Translation files for applications are named with the two-character Internet abbreviation for a country, such as “ES” for Spanish or “IT” for Italian, and are stored in the Translation directory at the same level of the directory structure as the applications and the menu. Thus the file, ES.js, is a JavaScript file containing Spanish phrases for the applications and Exercise files, and is found in the Translations directory
- A file with the name of the original language of the programs contains all the phrases to be translated, but indicates the translation only by “”. This file, currently EN.JS, should be copied to produce new translation files.
- At the top of the translation file—ES.js, for example—there is a definition of the array T. (In JavaScript, “T” is a different variable from “t”).

```
Var T = new Array() ; //Defines an array called T.
```

- The rest of lines in the file, ES.js, assign translated phrases to the T array. T is indexed by the phrases of the original language as found in the programs, rather than by numeric indices. Hence, the phrase, “Enter New Data” appears as follows in the file, EN.js

```
T[“Enter New Data”]=””;
```

- In the file, ES.js, the translator places the translation between the quotation marks:

```
T[“Enter New Data”]=”Entre Datos Nuevos”;
```

- When the user chooses Spanish-(ES) in the Options/Settings function on the OpenEpi Menu (or the language of the browser is Spanish), the programs will display the Spanish phrases, if any, that are found in the ES.js file. If a phrase is missing or is represented by “”, the display reverts to the index phrase in the original language.

Within the JavaScript and HTML

The JavaScript file, AppHelper.js, contains functions called `initTranslation()`, `t()`, and `TranslateHTML()`. The `initTranslation` function checks a cookie that is set by the `Settings.htm` program (Options/Settings on the menu) to indicate the users choice of language. It then loads the appropriate translation file(s) into memory, creating the T array corresponding to the desired language. The `t` function receives a phrase as `t(phrase)` and returns the corresponding value from the T array. Hence, if the phrase is “New Data” and the language is ES, `t(phrase)` may return “Nuevos Datos” to be used by the program. `TranslateHTML(webdocument)`, when directed to a web document, extracts all the text items from the body of the document and sends them to `t()` for translation.

Within the programs, two categories of phrases are automatically sent to the translation function, `t(phrase)`.

- All text phrases in the commands of the OpenEpi Input and Output languages, such as “title” and “newrow” are automatically sent to the `t` function.
- Components of the dynamically created main page of each application, including the Title, Authors, and Description variables in the application’s js file, are automatically sent for translation.

Some phrases in an application must be placed in calls to the `t` function, as in

```
Msg1= t(“This message pops up when a problem occurs”).
```

When `t` is called, the translation for the indicated phrase is returned from the T array indexed by the phrase. If no such item exists or the translation has not been provided, the

original phrase is returned. Phrases are filtered so that numbers are not translated. The translator has the option of ignoring words that should not be translated or of removing their entries from the file. Removing a line like `T["CDC"]=""` or leaving it alone simply means that CDC will not be translated. Neither choice will cause an error.

For translation of HTML document files such as those describing Demos and Exercises is done by calling the function `translateHTML(HTMLdocument)`. The specified HTML page is automatically scanned by `TranslateHTML` and all text phrases are sent to the `t()` function for translation. If `UpdateTranslation.hta` is running, text phrases are placed in the `EN.js` (or other specified `language.js` file during an update). In the absence of `UpdateTranslation`, text phrases are sent to the language array `T` (loaded with the correct language by `initTranslation()`), and the translated phrase is returned and displayed in the HTML page.

Using UpdateTranslation.hta to Create or Update Translation Files

The program called `UpdateTranslation.hta` runs only on Microsoft Windows® computers, where it can be used to extract all the phrases for translation from a new or revised OpenEpi program. To run `UpdateTranslation.hta`, first unzip the OpenEpi files to a directory on the hard disk of a Windows computer. Find the `UpdateTranslation.hta` file in the directory where OpenEpi resides and double-click on its icon. You may see a threatening message about security, to which the answer is “Yes”. `UpdateTranslation` will then ask for the name of the language to be updated or created. Enter your choice—ES, for example-- and `UpdateTranslation` will then run the OpenEpi menu. If the file exists, it will be updated with new entries; if it does not, it will be created.

While the menu is run under `UpdateTranslation` as the top level window, the `t` function in the programs behaves differently than under normal conditions. If `t("my phrase")` finds a corresponding entry in the array, `T["my phrase"]`, it behaves as during ordinary display, returning either the value of `T["my phrase"]` (perhaps “mi frase”), or “my phrase” if `T["my phrase"]=""`.

If the program searches for a phrase without a corresponding item in `T`, for example, “New Phrase”, and there is no `T["New Phrase"]`, then this item is appended to the end of the translation file. Hence exercising all the functions in all the programs once while `UpdateTranslation.hta` is running will place all the phrases in the programs that have been automatically or manually marked for translation into the translation file. The exception would be a phrase that occurs in an alert box that appears only under special conditions. Such a phrase can be inserted into the translation file manually, but the following will ensure that it is always translated. Suppose that the alert box signifies a rare event, as follows:

```
if (rareEvent) {Alert("Very rare event occurred.") }
```

Normally this would be handled by inserting a call to `t()`

```
if (rareEvent) {Alert(t("Very rare event occurred.")) }
```

but it is unlikely that `t` will actually be called while `UpdateTranslation` is running. Hence, the solution is:

```
var msg1=t("Very rare event occurred");
if (rareEvent) {Alert(msg1); }
```

In the latter case, `t` will be called for "Very rare event occurred" every time the program is run, and the phrase will not be concealed from the translation mechanism.

What happens if a phrase is edited by the programmer? Suppose that "Very rare event occurred" is changed to "A rare event occurred". The next time that the program runs under `UpdateTranslation`, the new phrase is appended to the file, leaving the old phrase and its translation in place. Although this may leave some "dead wood" in the file, it has the great advantage of requiring new translation only for the new phrases. There is no need to start translating from the beginning when a new version comes out. Of course, as with any computer operation, it is a good idea to back up your working files before running `UpdateTranslation.hta`.

Translation of Longer Document Files

Document files are in HTML format, although they may have originated in Microsoft Word or PowerPoint. Shorter documents, like the Demos and Examples files, whether dynamically generated or saved as HTML files, can be translated by calling `translateHTML()` as described above. Those that are, are now contained in the directory `BriefDoc`.

For longer documents--those in the Documentation directory-- the best method for translation seems to be to put the translation in the same HTML file, with an anchor or bookmark indicating the language at the top of the translated text. When the Documentation button is clicked in the application, the language block indicator (e.g, "#ES" if language is set to Spanish) is added to the document reference. If the Spanish translation is found in the HTML document at the anchor named ES, the translation is displayed; if not, the display defaults to the beginning of the document—the one in the original language.

For example, when the user interface is written dynamically in `Proportion.htm`, the Documentation button links to:

`ProportionDoc.htm#ES`

When the ES anchor is present, the browser loads ProportionDoc.htm and jumps to the ES anchor at the top of the Spanish translation. If the ES anchor is not found, however, the browser displays the English translation at the top of the document. There are no errors or interruptions when the ES anchor is absent.

In translating longer document files, then, no new files are created (other than images perhaps), and translations in several languages can be found in the the same HTML file.

Including the Language Choice in the Options/Settings Page

To allow a user to choose a new language translation, the options in the file called Settings.htm may need editing to include the new language choice. To see if the language choice is already available, examine the Language choice in the Options/Settings page to see if the language is already present. If not, edit Settings.htm to add the new language choice.

If you produce a new translation, please notify Andy Dean and/or Kevin Sullivan at the addresses given on the OpenEpi website, www.openepi.com, so that that the translation can be added to the website version of OpenEpi.

Tips and Problem Solving

In inserting translated phrases, embedded quotation marks will cause problems and even lock up the system. Since the translations are enclosed in double quotes, single quotes can safely be used inside the phrases, but NOT DOUBLE QUOTES. If it is necessary to use double quotes, they can be preceded by a slash (\'') to tell JavaScript to ignore the meaning of the quote. The same can be done with other characters that may have special meaning to JavaScript. Strings (text phrases) must be on a single line, leading to some very long and awkward lines. A string can be broken into lines as follows, however:

```
T["This is a very long phrase that extends beyond the right margin of the word
processor"]="Translation of very long phrase "+
"can be broken using a \+ and \' sign";
```

It is also all right to break the line on either side of the equal sign. Note that a space between words must be included after the word "phrase" above, or the word 'phrasecan' will appear in the translation.

If a translated JavaScript file seems to cause trouble, try running the application in Netscape 7. After or while it is running, choose Tools|Web Development|JavaScript Console from the Netscape menu. This brings up a debugger that will often pinpoint syntax errors such as unpaired quotation marks. If this does not solve the problem, try commenting out suspect lines, such as those with special symbols, using // at the

beginning of the line. Look for unpaired quotes, particularly with long lines, where it is easy to omit the closing quote. If none of this works, please feel free to ask the authors for help. We would like to know about translations, and to have them work correctly.