Forms

mPDF can generate a static view of HTML forms and their elements, or (from mPDF >=5.3) can produce an "Active Form".

"Active" Forms

Active forms can be generated which can either be printed, or the data submitted to a URI. The variable `$this->useActiveForms` should be set to **TRUE** either at run-time or in `config.php`

Compatibility & Limitations
Active forms have been tested with Adobe Reader 10, and Foxit 5.0 Some limitations and compatibility issues seem to be determined by the PDF reader, rather than by mPDF. This is especially true of the way the form fields are displayed, handling of non-ASCII text, and encryption.

Encrypted Active Forms do not work with Adobe Reader 10, but work fine in Foxit 5.0; I believe this to be due to the limited support of Adobe Reader for forms using the (old) PDF 1.4 specification i.e. mPDF.

Brief testing with the in-built Google Chrome PDF Reader also shows some limited functionality e.g. Submit doesn't work

Known limitations include:

- cannot save a completed form
- cannot save or export FDF data locally
- cannot "sign" PDF forms
- incompatible with rotated tables, HTML headers, and "keep-with-table" (is compatible with page-break:avoid - but not if rotated - and columns)
- cannot use SIP/SMP fonts for active form elements (ones which are subset as SIP/SMP)
- button images cannot be vector images (SVG or WMF)
- (active) radio buttons do not work inside a DIV fixed/absolute position (or with page-break-inside: avoid)

Creating a valid Active Form
A PDF document can only contain one active form e.g. submit will work on all fields in the document.

The method (GET/POST) and action (URI) are set when a <form ...> element is parsed, and remain active unless reset by another <form> element. All fields will be submitted as though from one form, whether they are enclosed within a <form> element or not.

All fields should have names.

Field names must only contain letters, numbers, colon(:), undersore(_) or hyphen(-). This is largely as per HTML spec, but cannot contain a period(.) as this is part of PDF spec. for name hierarchies.

Field names should usually be unique, except for radio buttons.

Duplicate field names can be used (e.g. to echo the text to a field elsewhere in the document), but fields with the same name must be of the the same type, and have the same default value set.

Value(s) for radio buttons and checkboxes are required, and can only contain letters, numbers, colon(:), undersore(_), hyphen(-) or period(.)

Values in all other form fields can contain any unicode character (although obviously only win-1252 codepage if you are using core fonts only for the document). HTML entities e.g. ߾ are recommended.

See HTML attibutes for details of attributes which can be set e.g. disabled, required etc.

See supported CSS for style properties which can be applied.

Exporting (submitting) data
Data from forms can be submitted to a URI in either HTML or XFDF format (cf. `config.php`). XFDF is a form of XML and is recommended, because of encoding issues. See `formsubmit.php` in the example folder for ideas on how to handle the submitted data.

NB A submitted radio button field name is doubled with an underscore i.e. "myButtonName_myButtonName"

If the export format is XFDF, the submitted data is always UTF-8 encoded.

If the export format is HTML, it is much more complicated. From a "core-fonts" only document, the submitted data uses PDFDocEncoding. (See `formsubmit.php` in the example folder for a conversion script.) But if the form contains any characters which are not in the PDFDocEncoding (similar to Windows-1252), Adobe Reader will decide which encoding to use(!?)

It is therefore recommend that you either use mPDF('c') and decode html POST from PDFDocEncoding, or

use XFDF.

The default HTML submit method is POST; GET only seems to work from a PDF document opened in a standalone Reader (not in the browser).

You can specify a mailto address as a URI e.g. `action="mailto:email@address"` but you may find that it is blocked by the user's computer if using the HTML format.

Radio buttons
Disabled: if one radio button is set as disabled, mPDF will disable the whole group. PDF readers seem to handle this situation differently i.e. Adobe Reader 10 still allows selection of the disabled button, whilst Foxit disables the whole group.

Javascript
Javascript can be set for buttons using `onClick=""` but note this uses "Acrobat" Javascript. (You can download the Acrobat Javascript reference manual from the Adobe Developer's site).

For select, text and textarea you can use `onChange=""` which is triggered after the value has been changed.

> **Note:** From mPDF >= 5.4, <textarea> and <input type="text"> will accept javascript as: `onKeystroke`, `onValidate`, `onCalculate` and `onFormat`.
> `onChange` is depracated but works as `onCalculate` (for <textarea> and <input>).
> Select still accepts `onChange`.

Unicode characters in JavaScript must be written by typing a backslash, a lowercase "u", and then the four digit hexadecimal number corresponding to the character's encoding in the utf-16 character set e.g. \u2042

Appearance of form fields
Adobe Reader 10 largely ignores any control one tries to place on the appearance of some form fields, and does it's own thing. In general, the `font-size` set for the form field will determine its size, and for text/textarea and select fields, `color` will determine the font colour used. CSS values for `border-color` or `background-color` will work for (non-image) buttons, textarea and text fields. Other things like the border style and width can be altered by configurable variables in `config.php` but the level of control is disappointing.

Radio buttons and check-boxes use Adobe Reader's own icons, but Foxit uses information provided by the PDF file. The variable `$this->formUseZapD` determines whether ZapfDingbat symbols are used, or mPDF's appearance streams designed to mimic Adobe Reader's appearance.

Some components of interactive forms may be output in RGB colorspace even if you have specified `restrictColorSpace`. Since restricted colorSpace is mainly used for PDFA/PDFX files - which cannot contain active form fields anyway - this shouldn't matter.

Printed on Wed 05 Aug 2015 12:10:10 GMT +0100 (DST)