Interactive form

Open this in a reader that supports forms - the fields below are fillable.

Full name

Ada Lovelace

A value in /V, a tooltip in /TU, and a box the library draws itself.

Email

Required, so a reader marks it when the form is submitted empty.

Passphrase

Password: echoed as bullets. Advisory only - not secret storage.

Postcode

Comb + MaxLength: one character per cell, evenly spaced.

Notes

Multiline: this box wraps and scrolls.

Multiline, so a reader wraps the value rather than scrolling it sideways.

Subscribe

✔

/AP /N holds one stream per state; /AS names the one on show.

Delivery

(•) Standard ( ) Express ( ) Collect

One field, three widgets under /Kids. The field holds the name and the value.

Country

United Kingdom

Combo + Edit, so the list can also be typed into. Sort orders it for display.

Interests

Typography
Colour management
Page imposition
Tagged PDF

A list box is a choice field without the Combo flag. /I carries the selected rows.

Then

Read the manual

A push button carries an action instead of a value: /A here is a URI action.

What the typed AcroForm API does

Every field on page one is a PdfTextField, PdfCheckBoxField, PdfRadioButtonField, PdfComboBoxField, PdfListBoxField or PdfPushButtonField, made with new, named, given flags, added to the form and put on the page. None of it is assembled by hand.

It used to be. Every constructor under PdfPinata.Pdf.AcroForms was internal, PdfAcroFieldCollection had no Add, PdfWidgetAnnotation was internal and there was no way to make a form at all - so the only route was to write the dictionaries of ISO 32000-1 section 12.7 yourself and hang them off the catalogue's /AcroForm.

One capability to a line, and where it lives

Make a form	`PdfDocument.GetOrCreateAcroForm()`
Create a field of any type	`new PdfTextField(document), and so on`
Add a field to a form or a field	`PdfAcroFieldCollection.Add`
Put a field on a page	`PdfAcroField.AddWidget`
Name it, describe it, flag it	`Name, ToolTip, Flags`
Give a widget an appearance	`PdfAnnotation.SetAppearance`
Offer choices	`PdfChoiceField.Options`
Read and fill a form somebody wrote	`AcroForm.Fields[name].Value`
Make every field read-only	`PdfDocument.MakeAcroFormsReadOnly()`
Sign a document	`PdfPinata.Signing - see the Signing demo`
Flatten a form into page content	`not offered`

Two entries above are still written by name, because nothing wraps them: /MK, which a viewer paints a field's box from when it builds the appearance itself, and the push button's /A action. Everything else on page one goes through a property or a method.

One rule to know. A partial field name may not contain a period, because a period is what joins nested names into the path a field is found by - so Name = "name.full" is refused at the call rather than left to produce a field nobody can look up.