Text and structure recognition in PDF

Feature rich open source
developer's library for
PDF generation and manipulation
in web and other applications

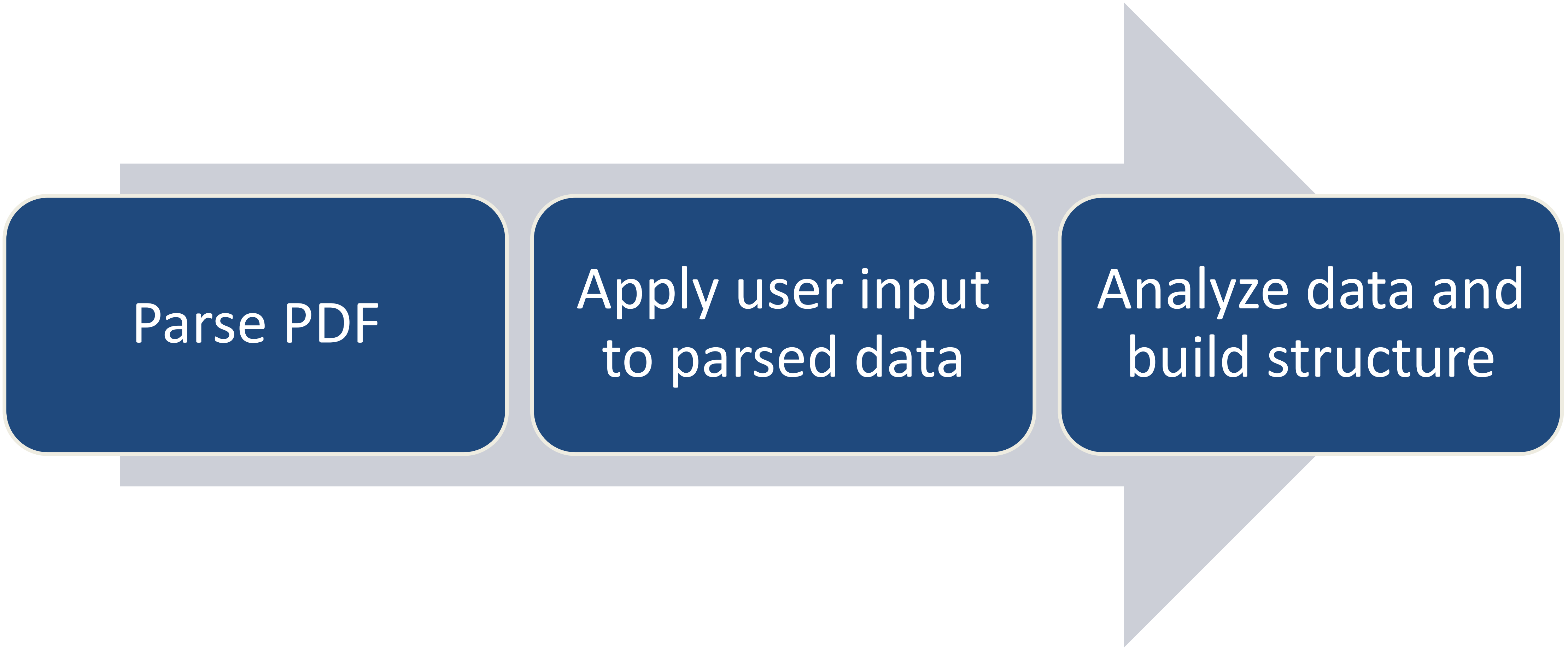
iText

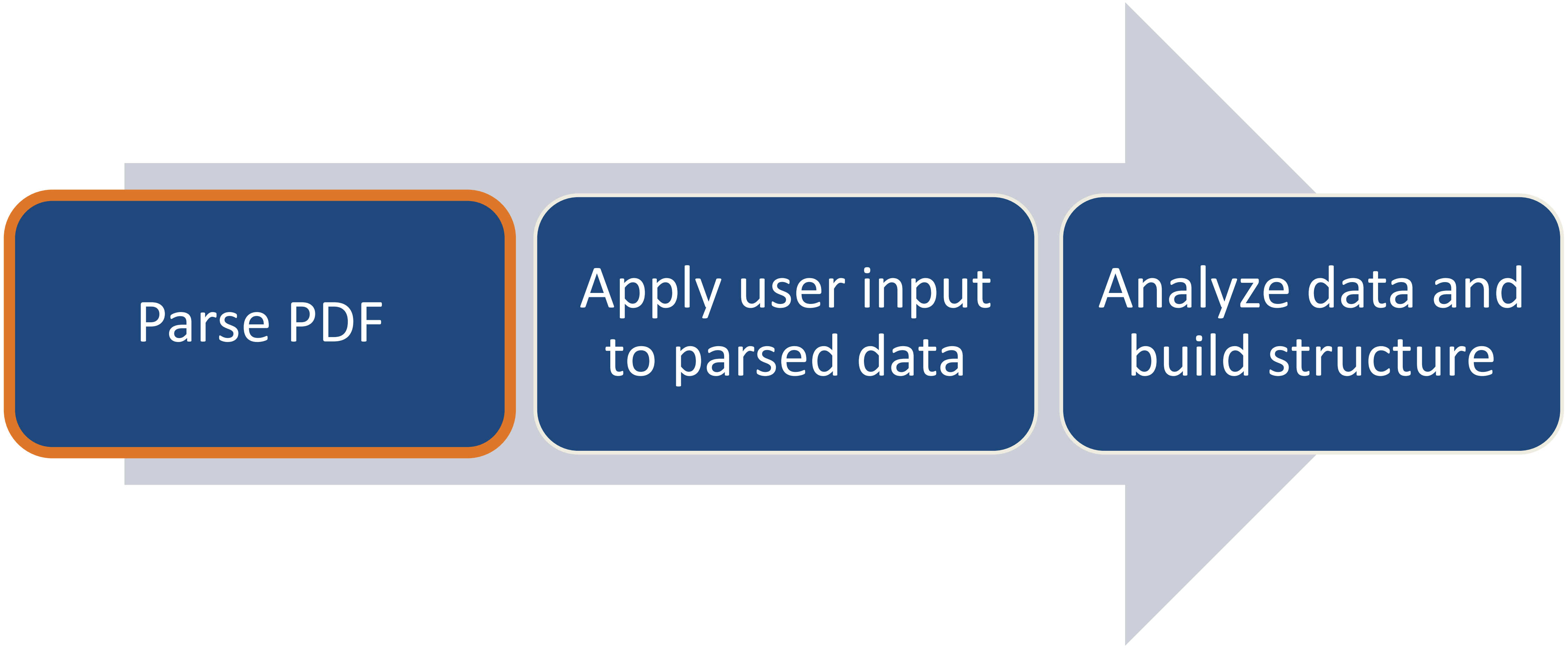
PDF structure recognition **with** iText

Real customer example

PDF structure recognition **in** iText

Presentation road map


Parse PDF
Apply user input to parsed data
Analyze data and build structure


Parse PDF
Apply user input to parsed data
Analyze data and build structure

- Extracts images from PDF page content
- Extracts text items from PDF page content
- Images and text items contain full graphics state
- User can specify listeners for extracted images and text items

Parse PDF. iText API

- Extracts images from PDF page content
- Extracts text items from PDF page content
- Images and text items contain full graphics state
- User can specify listeners for extracted images and text items
- **iText can do all that in only few lines of code!**

Parse PDF. iText API

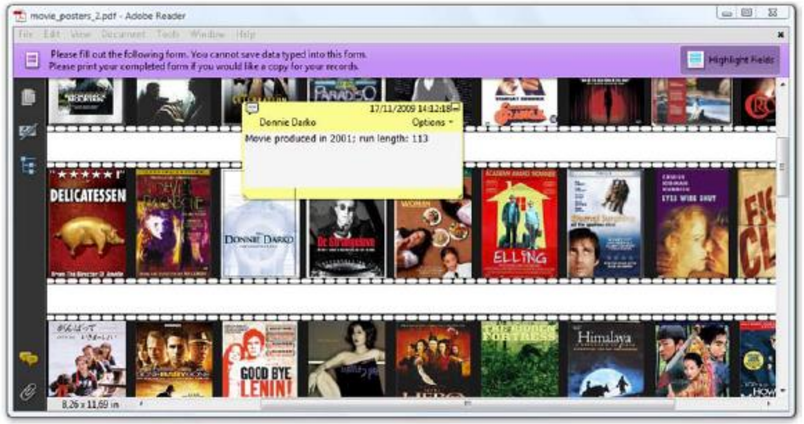
In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
Highlight Fields
Options

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

17/11/2009 14:12:18
Donnie Darko
Movie produced in 2001; run length: 113

Figure 7.15 Text annotation in a popup using a button and its events

```
PdfReader reader = new
PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
highlight(renderListener.items, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Parse PDF. Highlight all items

```
PdfReader reader = new
PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
highlight(renderListener.items, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Parse PDF. Highlight all items

```
PdfReader reader = new
PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
highlight(renderListener.items, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Parse PDF. Highlight all items

```
class MyRenderListener implements RenderListener {
    public List<Item> items = new ArrayList<Item>();
    public void beginTextBlock() {}
    public void renderText(TextRenderInfo textRenderInfo) {
        items.add(createItem(textRenderInfo));
    }
    public void endTextBlock() {}
    public void renderImage(ImageRenderInfo imgRenderInfo) {
        items.add(createItem(imgRenderInfo));
    }
}
```

Parse PDF. Highlight all items

```
class Item {
    public Rectangle rectangle;
}

class ImageItem extends Item {
}

class TextItem extends Item {
    public String fontName;
    public float fontSize;
}
```

Parse PDF. Highlight all items

```
BaseColor getColor(Item item) {
    if (item instanceof ImageItem)
        return BaseColor.RED;
    if (item instanceof TextItem)
        return BaseColor.BLUE;
    return null;
}
```

Parse PDF. Highlight all items.

RED images, **BLUE** text

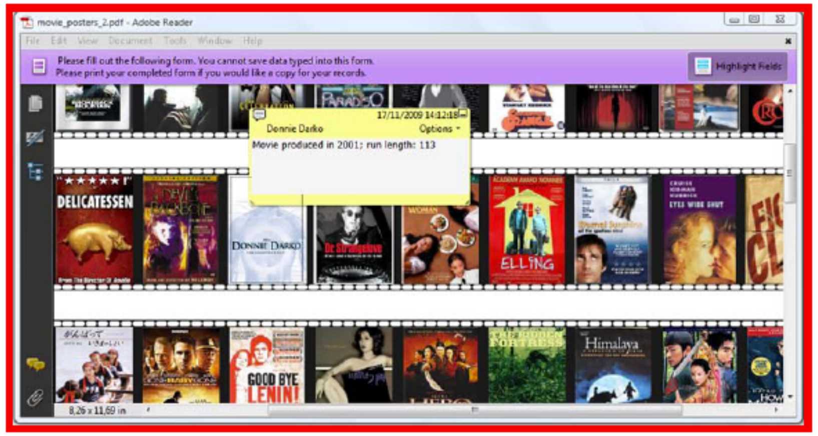
In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

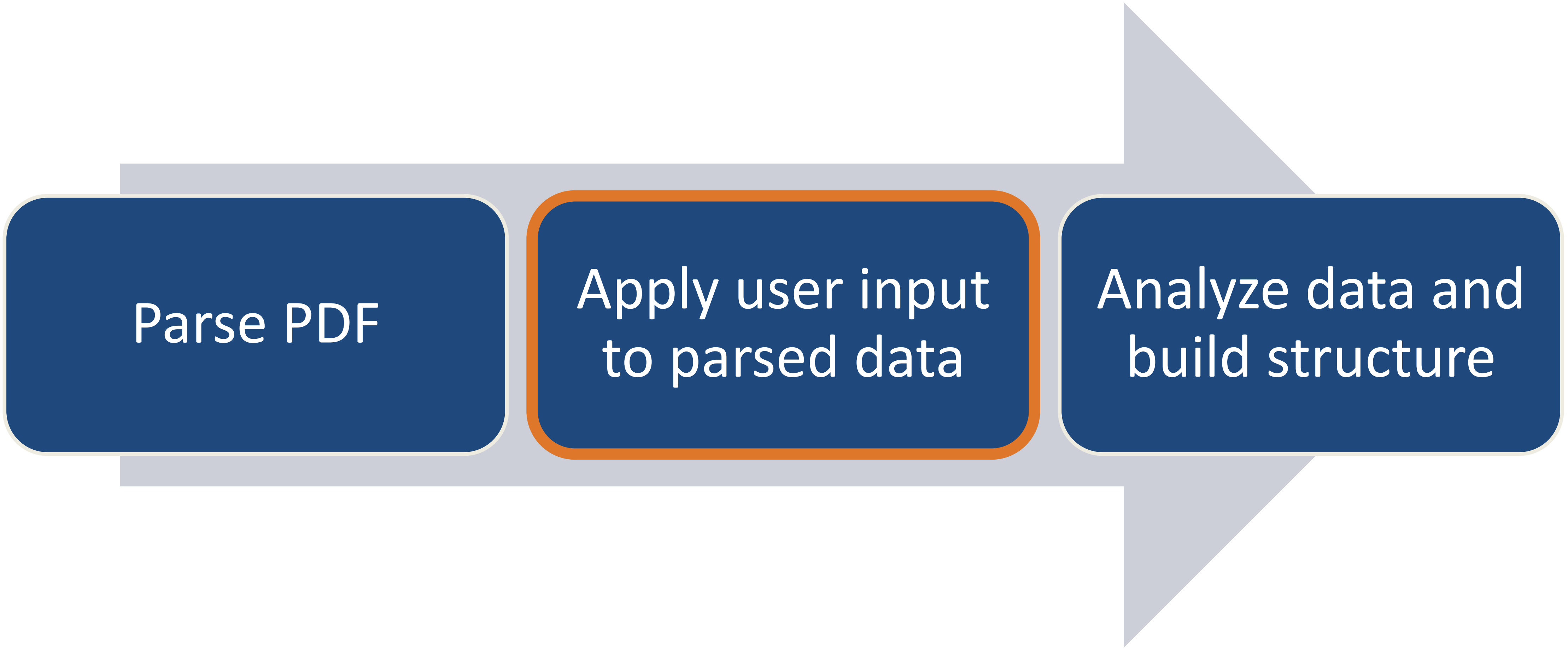
A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
Highlight Fields
8,26 x 11,69 in

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

17/11/2009 14:12:18
Donnie Darko
Options
Movie produced in 2001; run length: 113

Figure 7.15 Text annotation in a popup using a button and its events

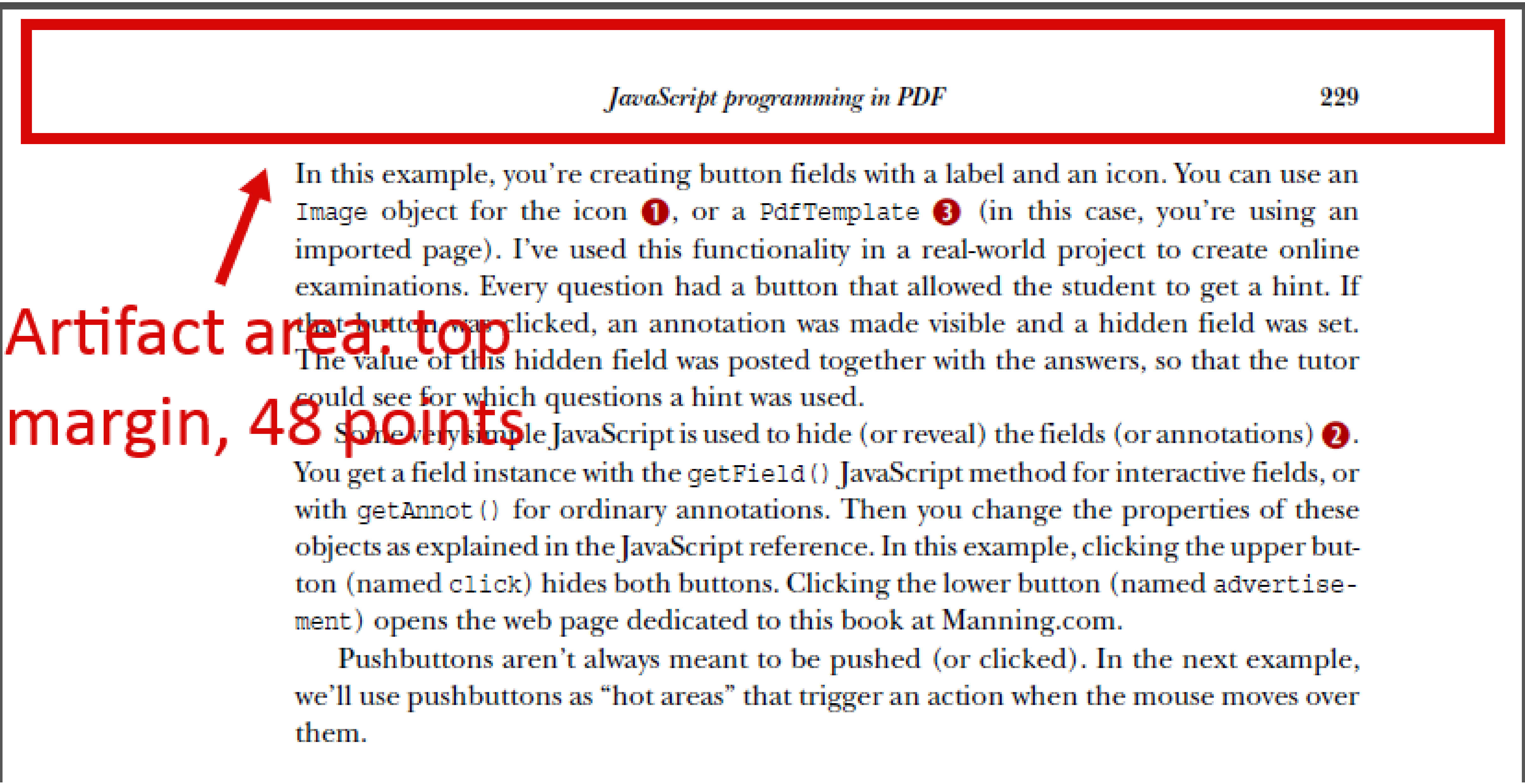

Parse PDF
Apply user input to parsed data
Analyze data and build structure

- Helps structure recognition processor to analyze parsed data
- User Input can be:
  - Text styles definitions
  - Page artifact areas definitions
  - …

Apply user input to parsed data

- Helps structure recognition processor to analyze parsed data
- User Input can be:
  - Text styles definitions
  - Page artifact areas definitions
  - …
- **Let’s add some input!**

Apply user input to parsed data



Artifact area: top margin, 48 points

In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

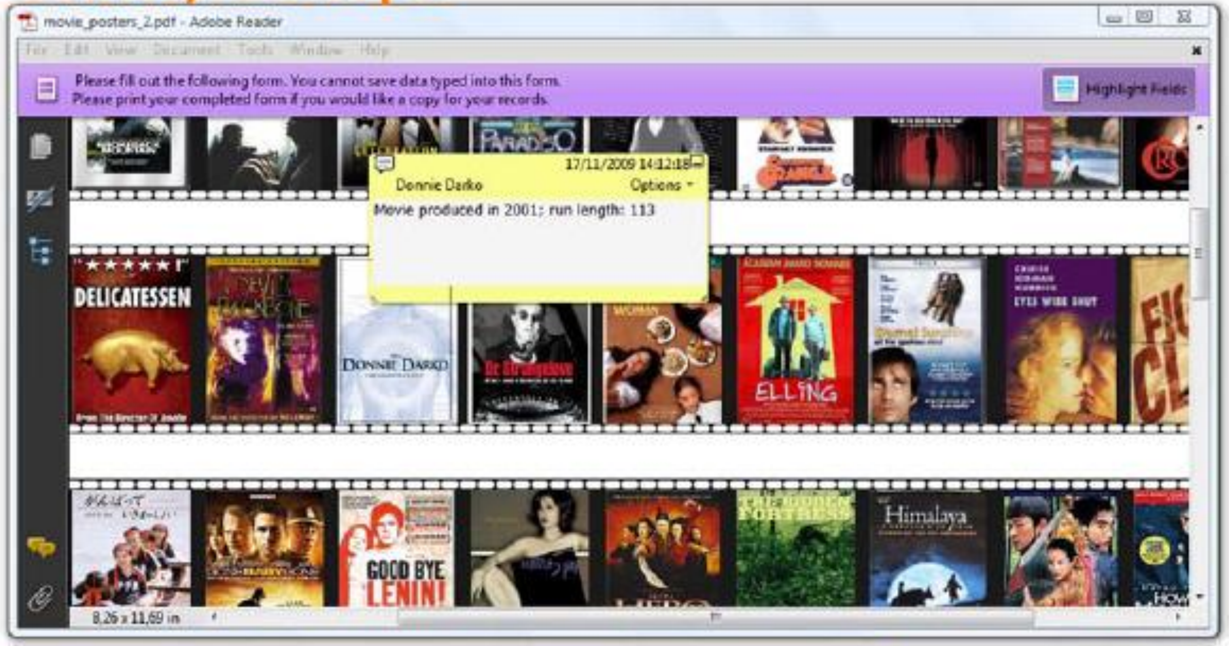
Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 *A popup triggered by a button that doesn't need to be pushed*

Header: FranklinGothic, 10.5pt

A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.



movie_posters_2.pdf - Adobe Reader

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

Highlight Fields

Donnie Darko 17/11/2009 14:12:18 Options

Movie produced in 2001; run length: 113

8.26 x 11.69 in

Figure 7.15 Text annotation in a popup using a button and its events

Caption, FranklinGothic, 8pt

Plain text:

NewBaskerville, 10pt;

Courier, 9.5pt

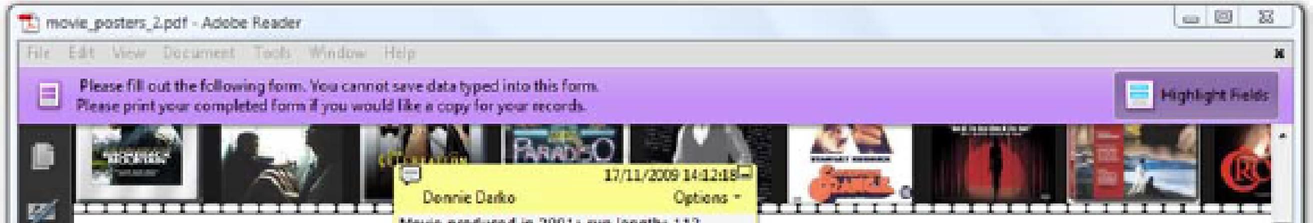
In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
Highlight Fields
Options

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

17/11/2009 14:12:18

Donnie Darko

[PSEUDO CODE]

```
BaseColor getColor(Item item) {
    if (item instanceof ImageItem)
        return BaseColor.RED;
    if (item instanceof TextItem) {
        if (item.baseline.y > pageSize.y - 48)
            return BaseColor.RED;
        else
            return textStyles.get(new TextStyle(item.fontName,
item.fontSize));
    }
    return null;
}
```

Apply user input to parsed data.

RED artifacts, BLUE text, ORANGE headers, GREEN captions

[PSEUDO CODE]

```
Map<TextStyle, BaseColor> textStyles = {
      (new TextStyle("FranklinGothic", 10.5), 
BaseColor.ORANGE);
      (new TextStyle("FranklinGothic", 8), 
BaseColor.GREEN);
      (new TextStyle("NewBaskerville", 10), 
BaseColor.BLUE);
      (new TextStyle("Courier", 9.5), 
BaseColor.BLUE);
}
```

Apply user input to parsed data. Text styles

[PSEUDO CODE]

```
class TextStyle {
    String fontName;
    Float fontSize;

    TextStyle(String fontName, float fontSize) { … }

    boolean equals(TextStyle ts) {
        return fontName.equals(ts.fontName) &&
            fontSize.equals(ts.fontSize);
    }

    int hashCode() { … }
}
```

Apply user input to parsed data. Text styles

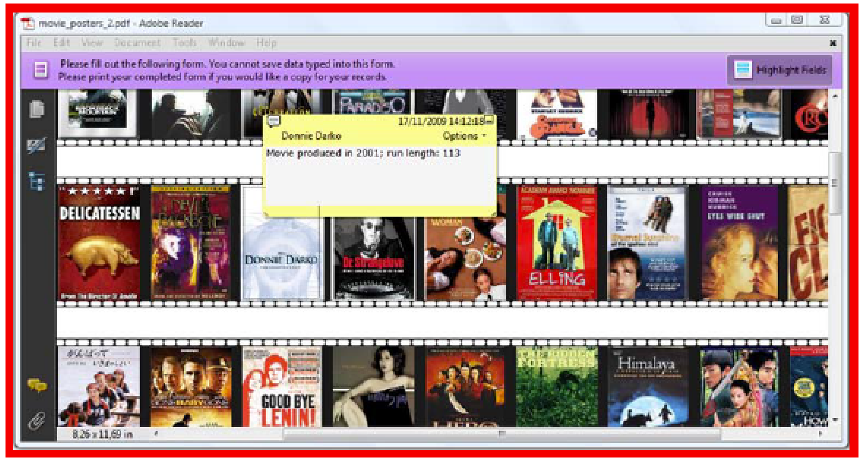
In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

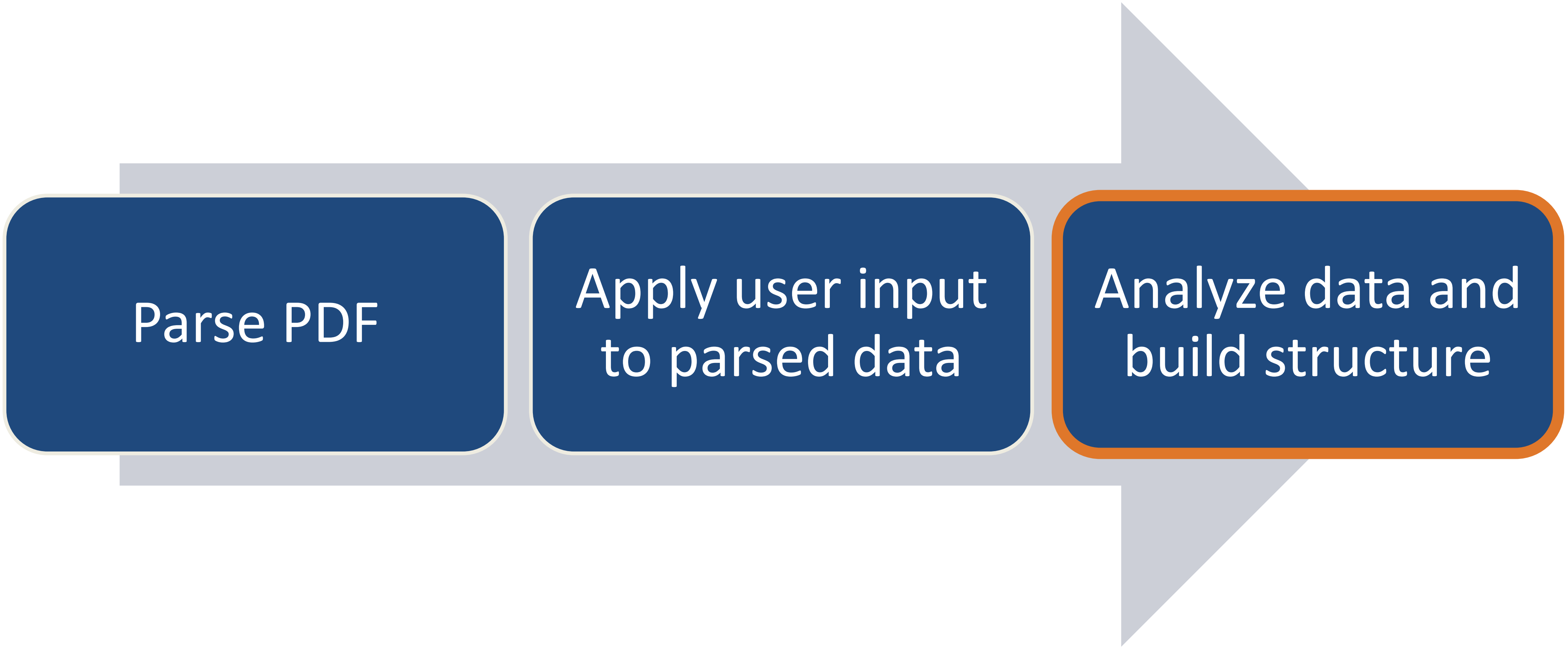
A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
Highlight Fields
8.26 x 11.69 in

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

Donnie Darko 17/11/2009 14:12:18 Options
Movie produced in 2001; run length: 113

Figure 7.15 Text annotation in a popup using a button and its events


Parse PDF
Apply user input to parsed data
Analyze data and build structure

- Sort all items in natural reading order
  - Items appear in PDF page content chaotically
- Combine items into lines
- Combine lines into common structures
  - Paragraphs
  - Headers
  - Captions

Analyze data and build structure

```
PdfReader reader = new PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
List<Item> items = sort(renderListener.items);
List<List<Item>> lines = getLines(items);
highlight(lines, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Analyze data and build structure.

Sort all items and combine into lines

```
PdfReader reader = new PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
List<Item> items = sort(renderListener.items);
List<List<Item>> lines = getLines(items);
highlight(lines, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Analyze data and build structure.

Sort all items and combine into lines

```
PdfReader reader = new PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new
MyRenderListener());
List<Item> items = sort(renderListener.items);
List<List<Item>> lines = getLines(items);
highlight(lines, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Analyze data and build structure.

Sort all items and combine into lines

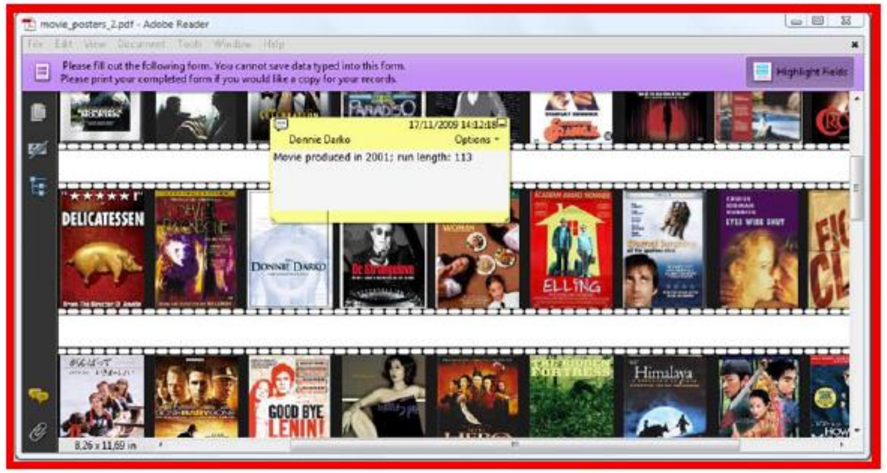
In this example, you're creating button fields with a label and an icon. You can use an Image object for the icon ❶, or a PdfTemplate ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the getField() JavaScript method for interactive fields, or with getAnnot() for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named click) hides both buttons. Clicking the lower button (named advertisement) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader
File Edit View Document Tools Window Help
Highlight Fields
8.26 x 11.69 in

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

17/11/2009 14:12:18
Donnie Darko Options
Movie produced in 2001; run length: 113

Figure 7.15 Text annotation in a popup using a button and its events

- Sort all items in natural reading order
  - Items appear in PDF page content chaotically
- Combine text items into lines
- Combine lines into common structures
  - Paragraphs
  - Headers
  - Captions

Analyze data and build structure

```
PdfReader reader = new PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new MyRenderListener());
List<Item> items = sort(renderListener.items);
List<List<Item>> lines = getLines(items);
List<List<List<Items>>> structures =
getStructures(lines);
highlight(structures, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Analyze data and build structure.

Combine lines into common structures

```
PdfReader reader = new PdfReader("iText.Book.pdf");
PdfReaderContentParser parser = new
PdfReaderContentParser(reader);
RenderListener renderListener =
parser.processContent(pageNum, new MyRenderListener());
List<Item> items = sort(renderListener.items);
List<List<Item>> lines = getLines(items);
List<List<List<Items>>> structures =
getStructures(lines);
highlight(structures, reader, pageNum,
"./target/iText.Book.pdf");
reader.close();
```

Analyze data and build structure.

Combine lines into common structures

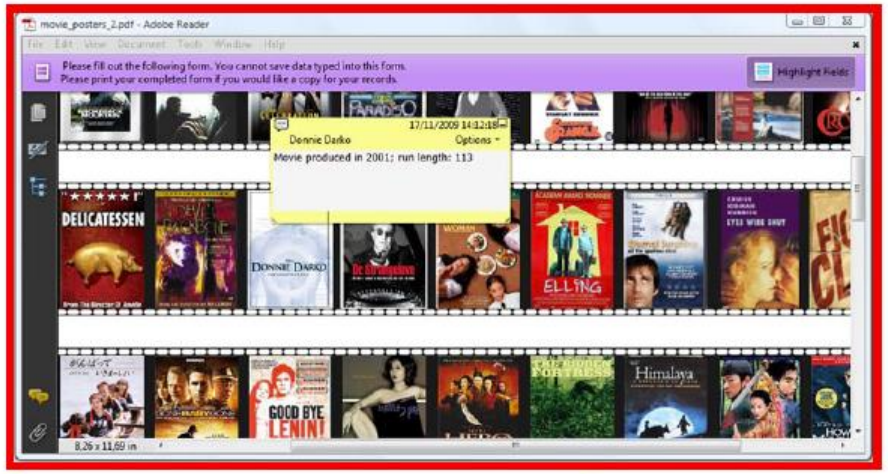
In this example, you're creating button fields with a label and an icon. You can use an `Image` object for the icon ❶, or a `PdfTemplate` ❸ (in this case, you're using an imported page). I've used this functionality in a real-world project to create online examinations. Every question had a button that allowed the student to get a hint. If that button was clicked, an annotation was made visible and a hidden field was set. The value of this hidden field was posted together with the answers, so that the tutor could see for which questions a hint was used.

Some very simple JavaScript is used to hide (or reveal) the fields (or annotations) ❷. You get a field instance with the `getField()` JavaScript method for interactive fields, or with `getAnnot()` for ordinary annotations. Then you change the properties of these objects as explained in the JavaScript reference. In this example, clicking the upper button (named `click`) hides both buttons. Clicking the lower button (named `advertisement`) opens the web page dedicated to this book at Manning.com.

Pushbuttons aren't always meant to be pushed (or clicked). In the next example, we'll use pushbuttons as "hot areas" that trigger an action when the mouse moves over them.

7.4.3 A popup triggered by a button that doesn't need to be pushed

A popup annotation has no appearance stream or associated actions of its own. It's always associated with a parent annotation. Figure 7.14 shows a text annotation as a popup. If you take a close look at the image, you'll also see a widget annotation on top of the *Donnie Darko* poster. If you move the mouse inside the borders of this widget annotation, the popup with the text annotation will appear; if you move the mouse pointer outside the widget annotation, the popup will disappear.


movie_posters_2.pdf - Adobe Reader

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

17/11/2009 14:12:18
Donnie Darko
Movie produced in 2001; run length: 113

Figure 7.15 Text annotation in a popup using a button and its events

PDF structure recognition **<u>with</u>** iText.

Done!

- Comments to example
  - Not all methods are exposed
  - Though structure is recognized we do not create tagged PDF here. We only highlight structure elements by color
- Limitations
  - More difficult layouts (multi column text, tables etc.) are not handled in this example
  - If plain text, headers, captions use same styles output can be incorrect

An iText project for GlobalSubmit, a health care service provider

Examining the content of PDF documents with the goal to add interactive features such as links bookmarks and annotations

Version 2.0.0.0
Copyright GlobalSubmit, Inc. 2014
All Rights Reserved. This program is protected by US and international copyrights laws.
License expiration: 7/22/2014

Real customer example. Live demo

- Build structure recognition API on a top of PDF parsing API
- Minimize user input for structure recognition
- Support more element types:
  - Lists
  - Tables
  - TOCs
- Support for complex layouts
- Possibility to export recognized structure to different output formats

For Europe, Middle-East,
Asia and Africa

Kerkstraat 108
9050 Gentbrugge
BELGIUM

T: +32 92 98 02 31
F: +32 92 70 33 75
E: sales.isb@itextpdf.com

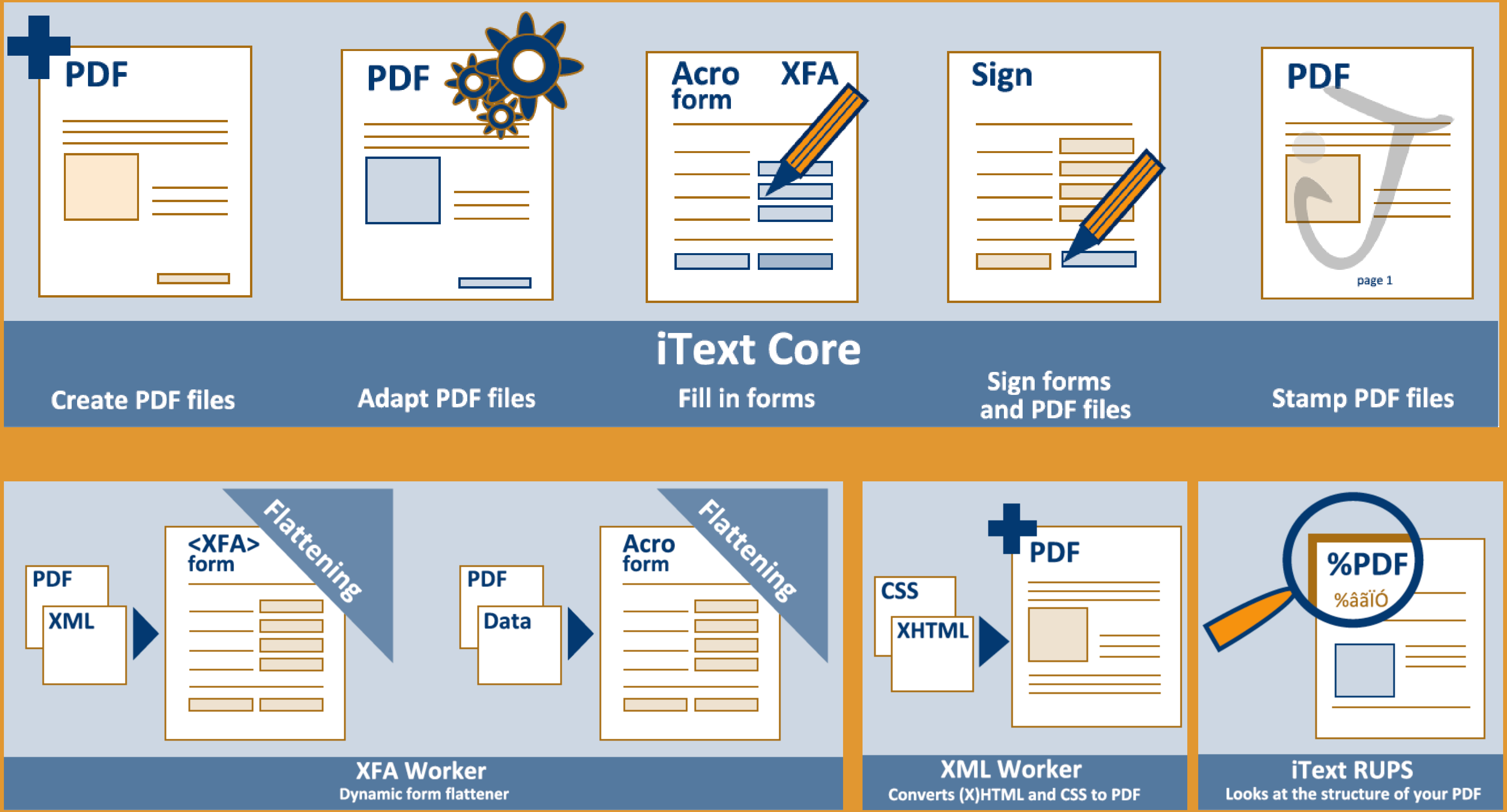
For the US, Canada,
Latin-America and Oceania

1 Broadway, 14th floor
Cambridge, MA 02142
USA

T: +1 617 982 2646
F: +1 617 982 2647
E: sales.isc@itextpdf.com

www.itextpdf.com

Contact us!



iText technology

www.itextpdf.com/learn

Download our free ebooks!