Jumpstart Tutorial

iText Presentation 26/07/2017

Jumpstart Tutorial

1. Basic Building blocks
2. PDF Syntax and Structure
3. Low level PDF creation
4. Combination of both approaches
5. Adding content to existing PDFs
6. Viewer Preferences
7. Metadata

Hello PDF World!

- Creating a PDF programmatically
  - Get the PdfWriter

```
PdfWriter writer = new PdfWriter(dest);
```

  - Get the PdfDocument

```
PdfDocument pdf = new PdfDocument(writer);
```

  - Get the Document

```
Document document = new Document(pdf);
```

  - Add Text : Hello PDF!

```
document.add(new Paragraph("Hello PDF World!"));
```

  - Close the Document

```
document.close();
```

Basic Building blocks

Building blocks

- Two fold approach to create and manipulate PDFs
  - High Level
  - Low Level
  - Mix of High and Low Level

- At High Level, intuitive building blocks of a document are used
  - Document
  - Paragraph
  - List
  - Image
  - Chunk
  - Cell
  - Table etc

Use of High Level APIs

- High level approach hides complexity of PDF syntax
- Setting a Font
  - Default font is Helvetica
  - Standard Type 1 fonts: 14 fonts recognized by PDF viewers
- Adding a Paragraph

```
// Create a PdfFont
PdfFont font =
    PdfFontFactory.createFont(FontConstants.TIMES_ROMAN);

// Add a Paragraph
Paragraph paragraph = new Paragraph("iText in Bangkok");
Paragraph.setFont(font);
document.add(paragraph);
```

Use of High Level APIs

Set Font properties

- Font Size
- Italic or bold
- Font Color

```
//Code
Paragraph para = new Paragraph("iText");
para.setStrokeColor(Color.RED).setStrokeWidth(0.5f)
para.setFontSize(24);

Text text = new Text("Hello Bangkok");
text.setItalic();
text. setBold();
text.setStrokeColor(Color.RED)
text. setStrokeWidth(0.5f)
```

Use of High Level APIs

Adding lists

```
// Create a List
List list = new List();
// List items are idented by 12 user units
list.setSymbolIndent(12);
// Creates a bulleted list
list.setListSymbol("\u2022");
list.setFont(font);
// Add ListItem objects
list.add(new ListItem(“Monday”));
list.add(new ListItem(“Tuesday”));
list.add(new ListItem(“Wednesday”)) ;
// Add the list to document
document.add(list);
document.close();
```

Use of High Level APIs

- Adding Images to PDF
  - Use of ImageDataFactory
    - detects the type of image that is passed
    - processes it so that it can be used in a PDF
  - Code for adding images:

```
Image fox = new Image(ImageDataFactory.create(path1));
Image dog = new Image(ImageDataFactory.create(path2));

Paragraph p = new Paragraph("The quick brown ")
p.add(fox)
p.add(" jumps over the lazy ")
p.add(dog);

document.add(p);
```

Use of High Level APIs

- Adding a Table
  - Code

```
Table table = new Table(new float[]{4, 1, 3, 4, 3, 3});
table.setWidthPercent(100);
```

  - Size of array = No of table columns
  - Float value = Relative width of a column
  - The third column is three times as wide as the second column.
  - Width of the table relative to the available width of the page (100%)
  - Adding a table with Header cells

Use of High Level APIs

- Some other high level methods
  - Rotating a page
  - Setting Page Margins
  - Setting Text Alignment
  - Setting hyphenation

```
Document document = new Document(pdf, PageSize.A4.rotate());
document.setMargins(20, 20, 20, 20);
document.setTextAlignment(TextAlignment.JUSTIFIED);
document.setHyphenation(new HyphenationConfig("en","uk",3,3));
```

PDF syntax and structure

PDF Syntax & Structure

- PDF is a binary format
- 8 types of objects
  - Boolean
  - Numeric
  - String (Literal & Hexadecimal string)
  - Name
  - Array Dictionary
  - Stream
  - Null
- Objects can be direct or indirect

PDF Syntax & Structure

- Comments are preceded by “%”
- %%EOF – End of File marker
- %PDF-X.Y – PDF version number

PDF Syntax & Structure

- Boolean objects
  - true
  - false
- Numeric objects
  - Integer
  - Real
- Strings
  - Preceded by a slash eg: /iText
  - Can contain any character
- Name
  - Literal strings: enclosed by parentheses ()
  - Hexadecimal strings: enclosed by angle brackets <>

PDF Syntax & Structure

- Array
  - Collection of PDF objects
  - Heterogeneous
  - Enclosed in square brackets []
- Dictionary
  - Key-Value pairs
  - Enclosed in Double angle brackets <<>>
- Streams
  - Sequence of bytes of unlimited length
  - All streams are indirect

PDF Syntax & Structure

- Indirect objects
  - Unique identifier : is a positive integer
  - Generation number
    - Positive integer, zero inclusive
    - Starts at 0
  - Enclosed by **obj** and **endobj**

  - Example:

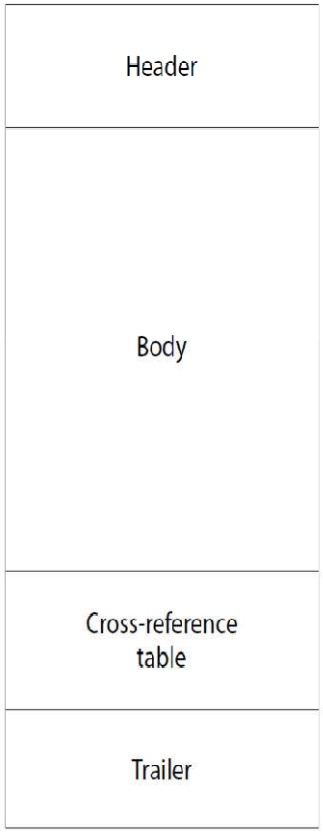
```
321 0 obj
  (An indirect object)
endobj
```

- This is an indirect string object with id “321” and generation number “0”. This object can be referenced from somewhere else in the PDF by calling the id and the generation number followed by “R”

  << /Value 321 0 R >>

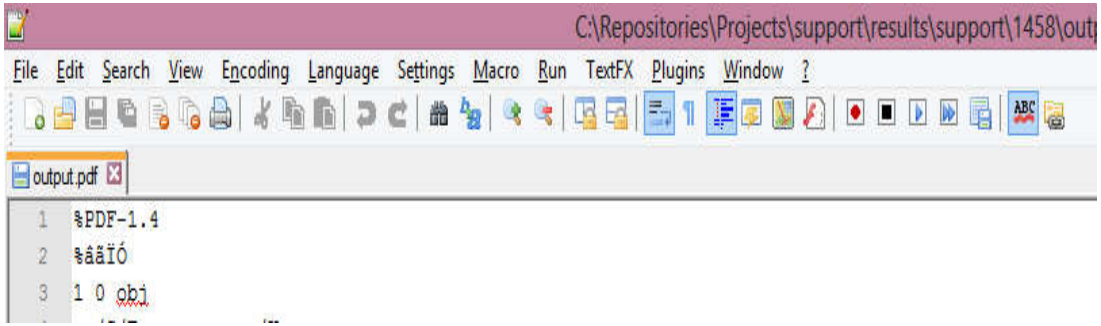
PDF Syntax & Structure

- PDF Structure has 4 sections :
  - Header
  - Body
  - XREF (Cross-reference Table)
  - Trailer



PDF Syntax & Structure

- Header
  - First line depicts the version
  - Replaced by /Version in the catalog dictionary (PDF 1.4+)
  - Second line indicating this is a binary file



PDF Syntax & Structure

- XREF table
  - Starts with keyword “**xref**”
  - One entry per indirect object

```
67  endobj
68  xref
69  0 4
70  0000000000 65535 f
71  0000002509 00000 n
72  0000000015 00000 n
73  0000000267 00000 n
74  8 10
75  0000000489 00000 n
76  0000000741 00000 n
77  0000000963 00000 n
```

PDF Syntax & Structure

- Trailer
  - Keyword trailer
  - Last line is %%EOF

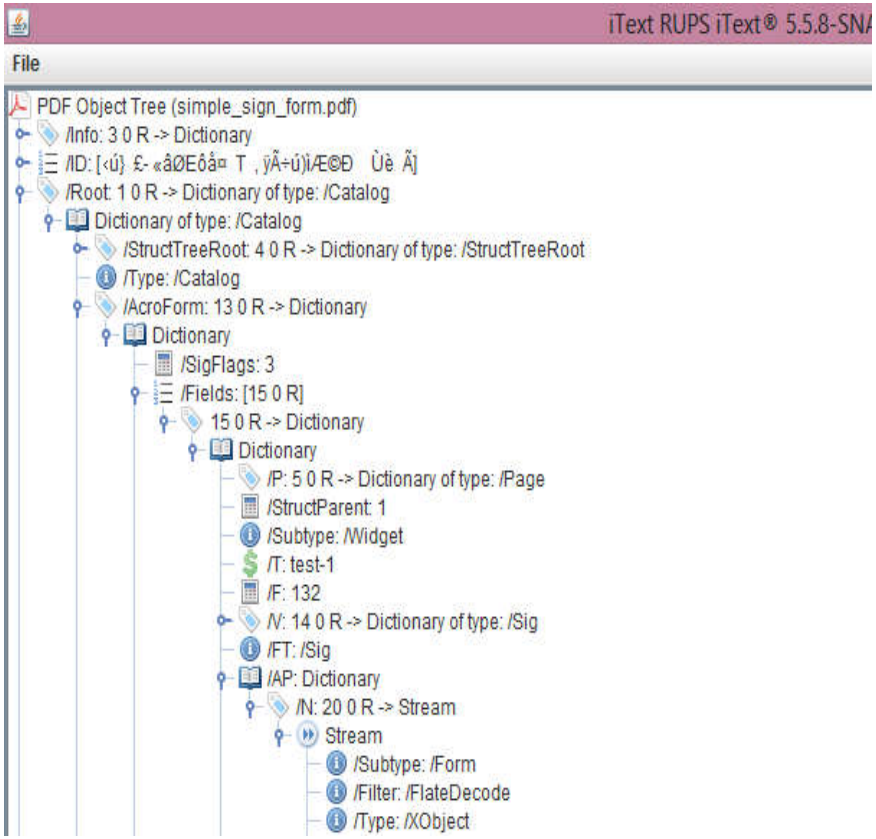
```
0000002976 00000 n
0000003029 00000 n
0000003288 00000 n
trailer
<</Size 24/Root 23 0 R/Info 9 0 R/ID [<f738801cdd3f6714585c82b317335c11><412ff00fbfcccd5346a31f86fbd34dbd>]>>
%iText-5.5.11-SNAPSHOT
startxref
3335
%%EOF

```

Normal text file | length : 3987 lines : 124 | Ln : 124 Col : 1 Sel : 0

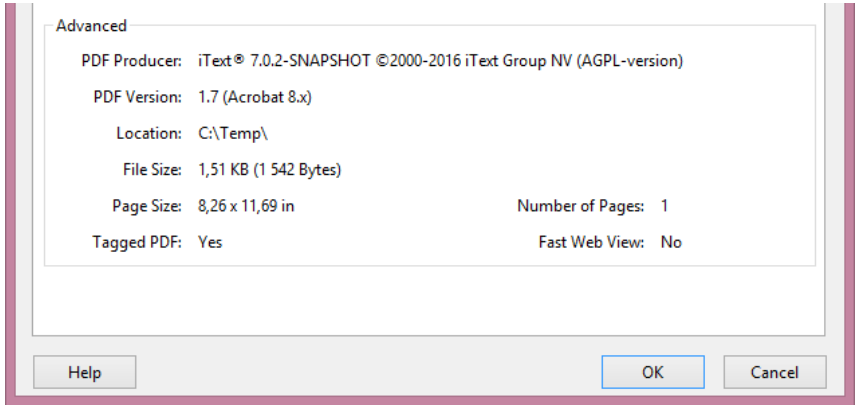
PDF Syntax & Structure

- Body
  - Sequence of indirect objects
  - Fonts & Pages



PDF Syntax & Structure

- Document structure
  - Catalog is the Root dictionary
    - Contains references to Page Tree, AcroForm, Outlines, Metadata, StructTreeRoot
  - Metadata
    - Contains information about the document
    - Author, Title, Creation Date, Modification Date, etc



PDF Syntax & Structure

- Page Tree
  - Defines the order of the pages
  - Two types of nodes : Page tree nodes & Page nodes

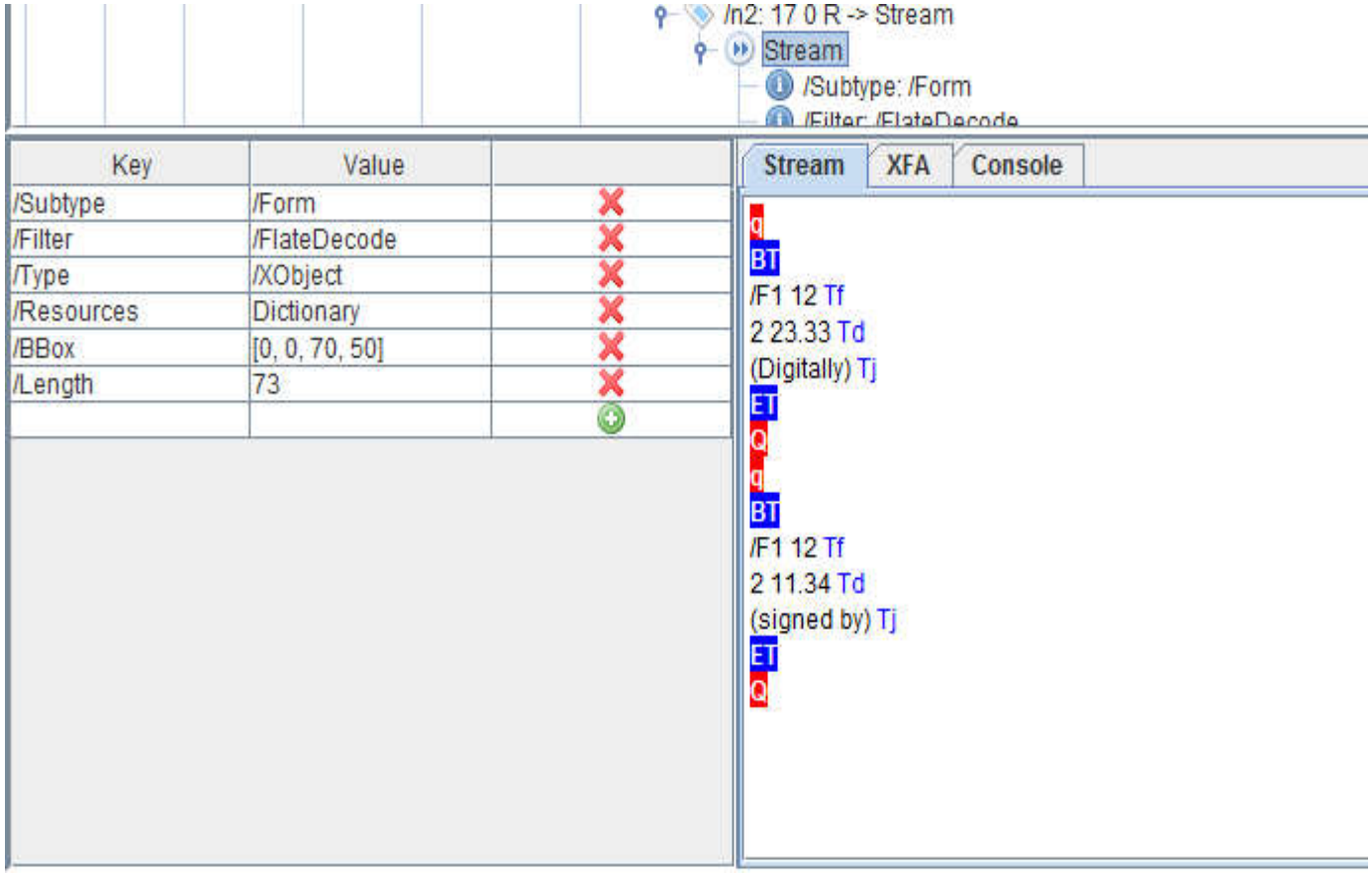
- Pages
  - Display information
    - User Units, MediaBox, Rotation, etc
  - Annotations
  - Content Stream(s)
    - Describes appearance of a page
    - Contains **operators** and **operands**

PDF Syntax & Structure

- Operator
  - Keyword specifying an action
  - Drawing a line, moving the cursor, etc
  - Only meaningful in the content stream

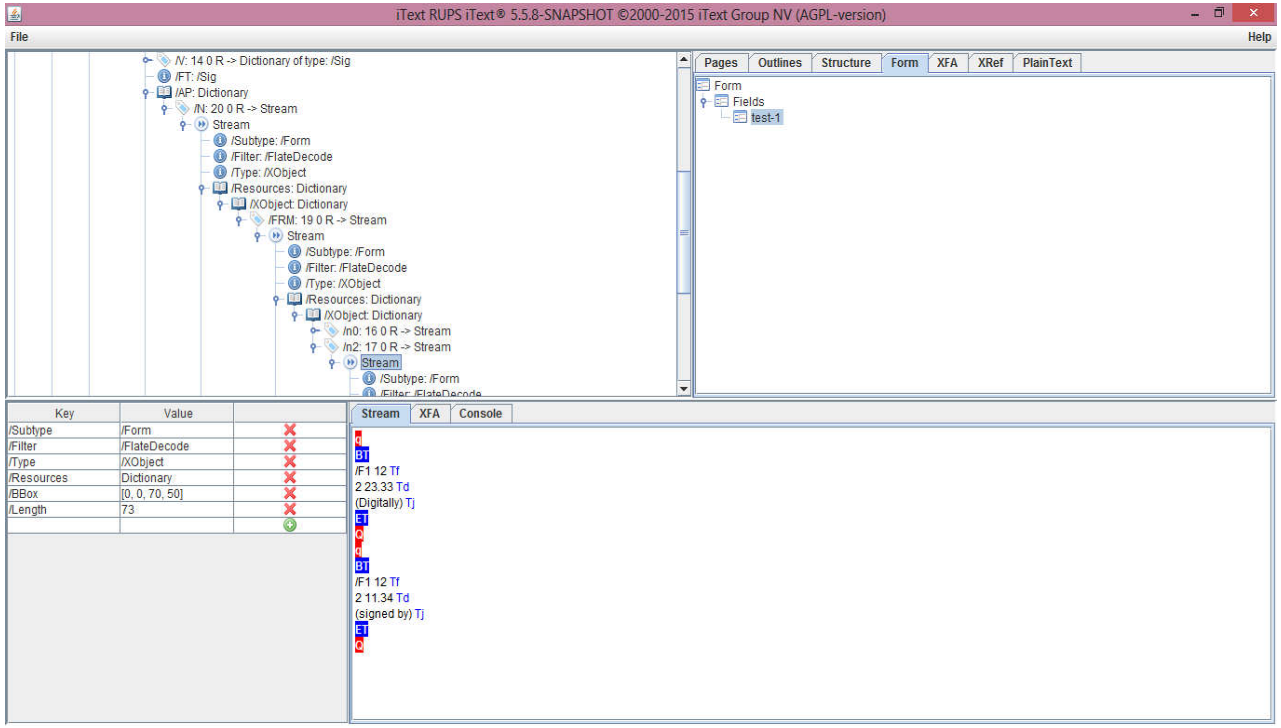
- Operand is a direct object needed by an operand
  - An operator can require more operands (e.g. Re)
  - Operands immediately precede operators

PDF Syntax & Structure



PDF Syntax & Structure

- RUPS
  - iText tool to inspect PDF files
  - Reading and Updating PDF Syntax
  - AGPL



Low Level

Use of Low Level Approach

- Writes to a PDF content stream
- PDF operators mapped to methods ()

```
m    moveTo() method
l    lineTo() method
S    stroke() method
```

- Manipulation of the layout
- Use of Canvas
  - No concept of Page
  - Use of absolute positions
  - Access to content streams
  - Adding FormXObject

Use of Low Level Approach

- Use of Tranformation matrix
  - drawing objects in a new co-ordinate system
  - concatMatrix() method
  - parameters of this method are elements of a transformation matrix

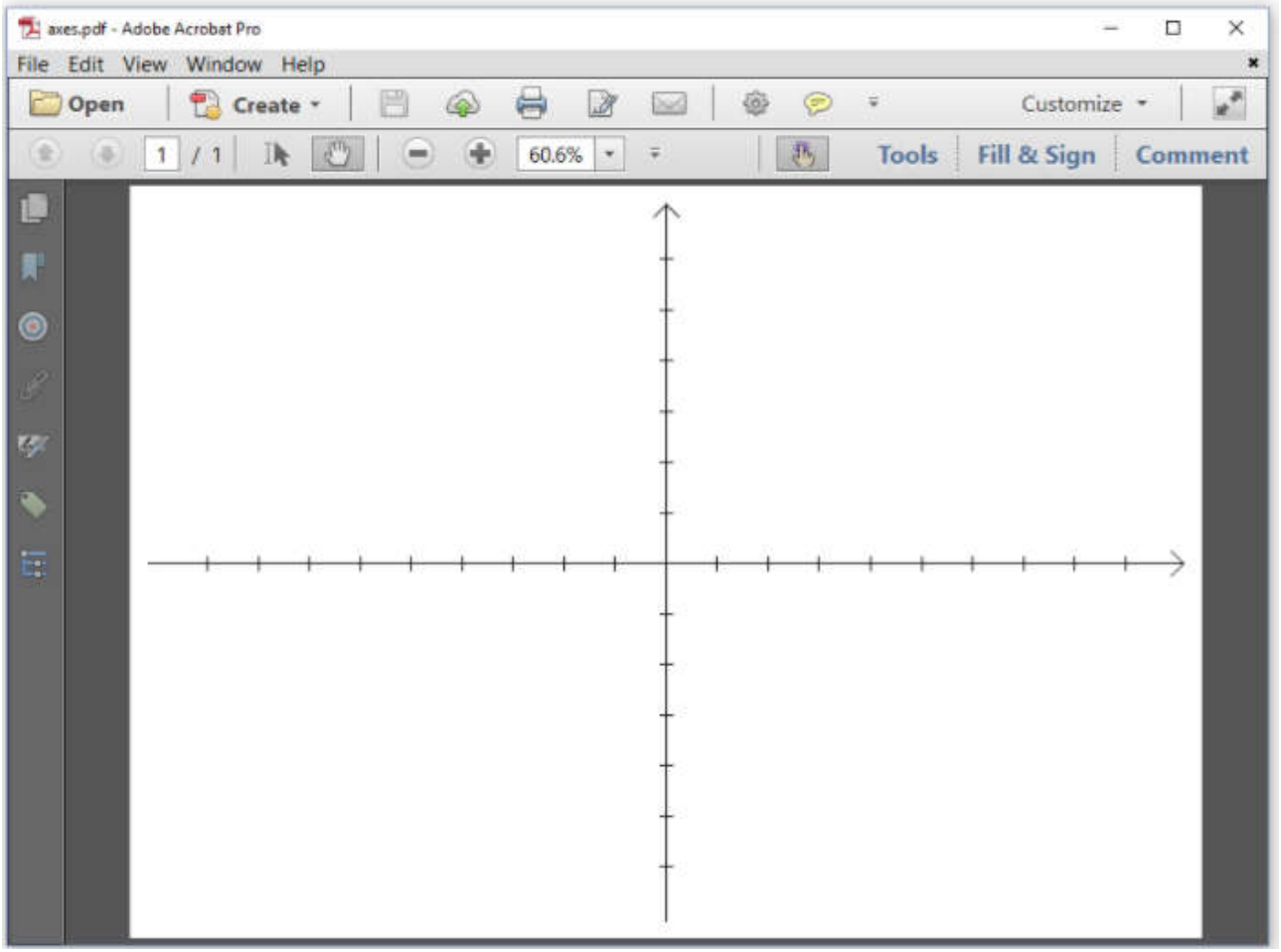
```
a   b   0   //3rd column is fixed since we work in 2D

c   d   0   //a, b, c, and d can be used to scale, rotate, and skew the
              coordinate system
e   f   1   //e and f define the translation
```

Use of Low Level Approach

- Graphics state
  - Properties such as the current transformation matrix, line width, stroke color, fill color etc.
  - Default line width is 1 user unit
  - Default stroke color is black
- Text state
  - Subset of the graphics state
  - Properties related to text i.e. text matrix, font and size etc

Use of Low Level Approach

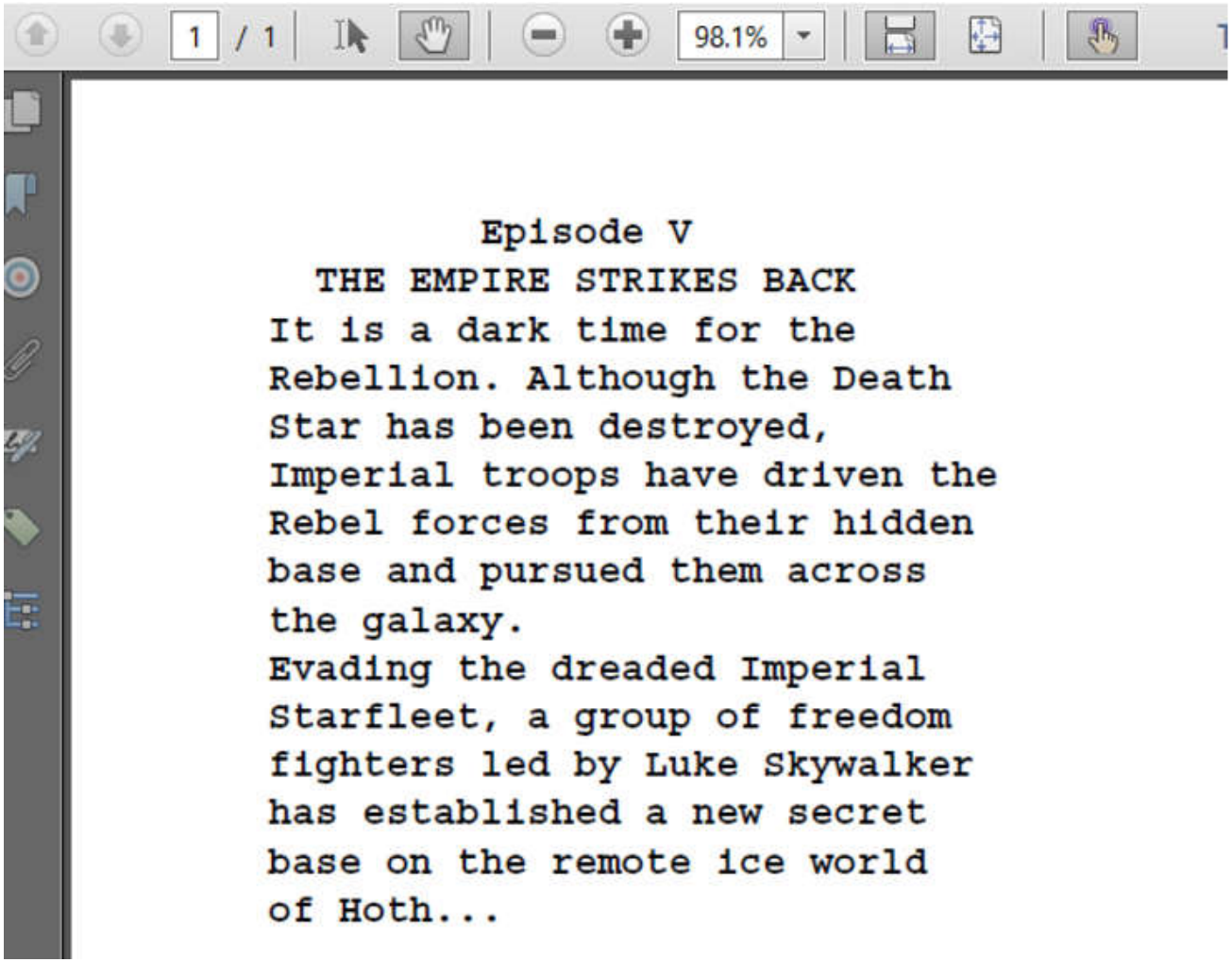


Use of Low Level Approach

- Color Spaces
  - As per PDF specification (ISO-32000)
  - Implemented under separate classes of iText

- Commonly used color spaces are:
  - DeviceGray (defined by a single intensity parameter)
  - DeviceRgb (defined by three parameters: red, green, and blue)
  - DeviceCmyk (defined by four parameters: cyan, magenta, yellow and black)

Use of Low Level Approach



Use of Low Level Approach

```
1    canvas.concatMatrix(1, 0, 0, 1, 0, ps.getHeight());
2    canvas.beginText()
3          .setFontAndSize(..)
4          .setLeading(14 * 1.2f)
5          .moveText(70, -40);
6   for (String s : text) {
7       //Add text and move to the next line
8       canvas.newlineShowText(s);
9   }
10  canvas.endText();
```

- Textbox is delimited by the beginText()/endText() methods
- Text should not be outside beginText()/endText()
- Do not nest beginText()/endText() sequences

Combination of Both Approaches

1. Event Handlers and Renderers
2. Annotations
3. Acroforms
4. Viewer Preferences
5. Metadata

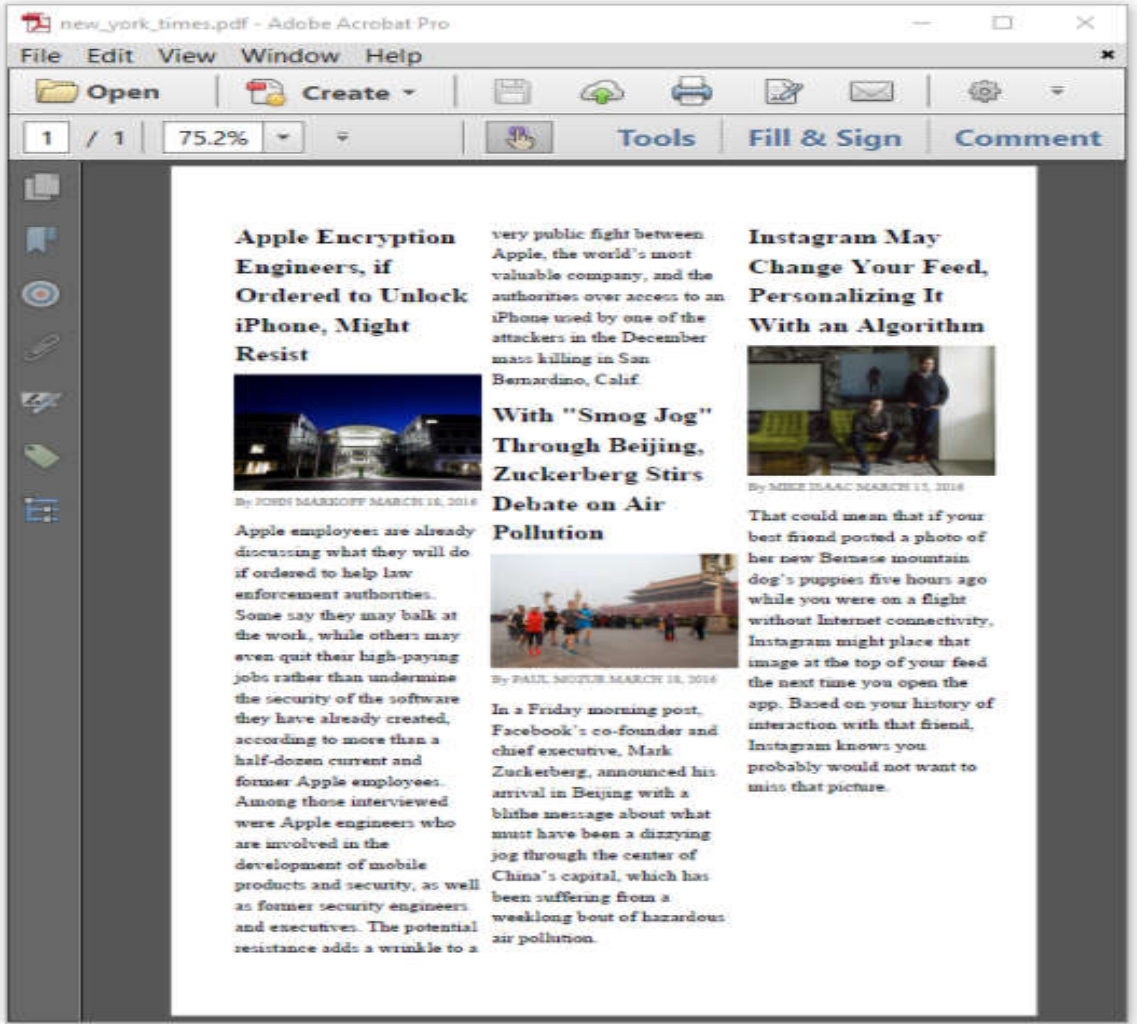
Event Handlers and Renderers

- Combining Low Level and High level approach for complex tasks

- Applying a Document renderer
  - Used on documents
  - Eg: To limit spanning of text across width of page

- Applying Block Renderer
  - Used on block level elements
  - eg: To modify properties of columns of a table individually
  - Eg: To modify individual cells of a table

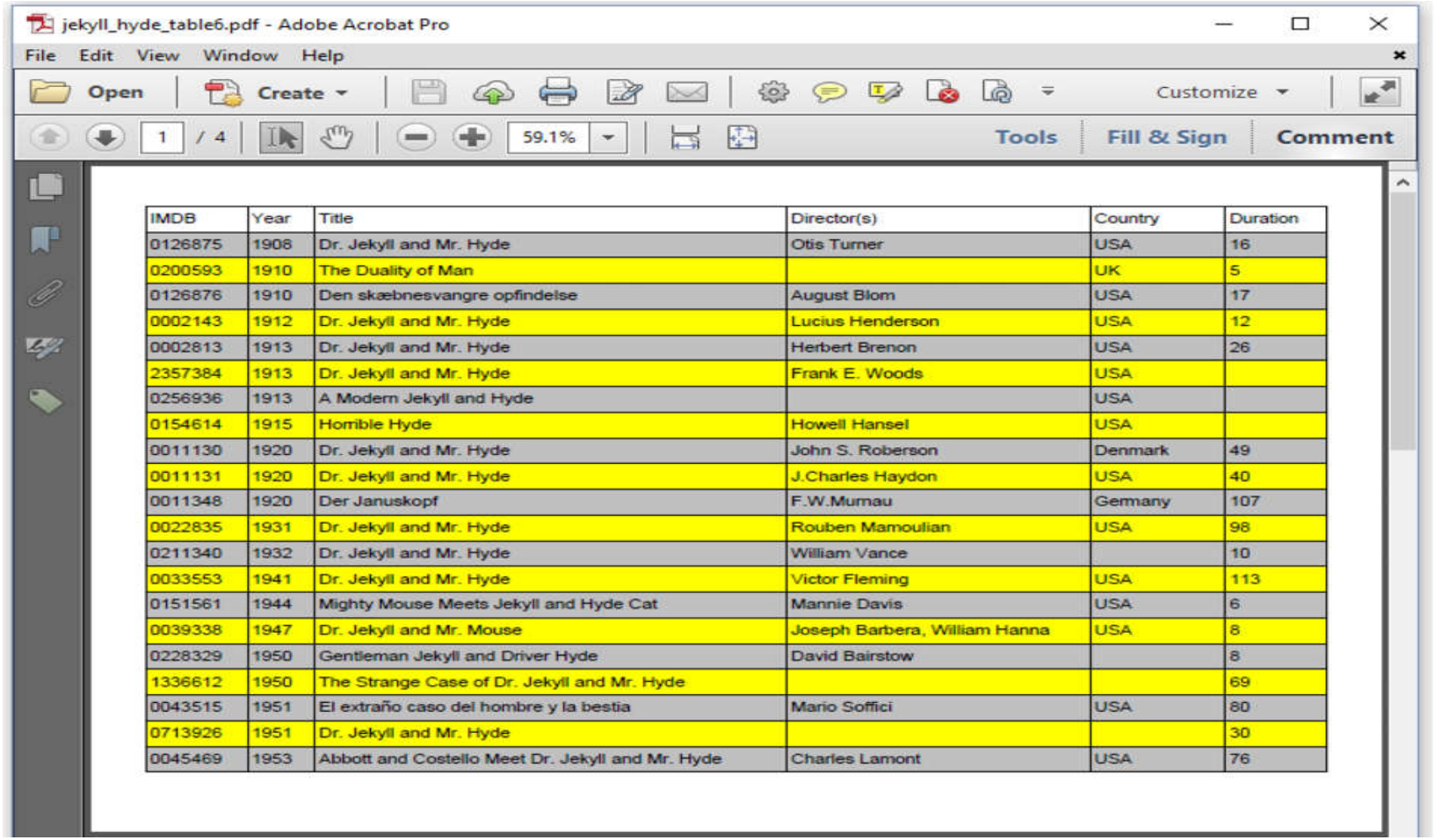
Event Handlers and Renderers



Event Handlers and Renderers

POS	CLUB	Played	Won	Drawn	Lost	Goals For	Goals against	Goal Difference	Points
1	Leicester City	30	18	9	3	53	31	22	63
2	Tottenham Hotspur	30	16	10	4	53	24	29	58
3	Arsenal	29	15	7	7	46	30	16	52
4	Manchester City	29	15	6	8	52	31	21	51
5	West Ham United	29	13	10	6	45	33	12	49
6	Manchester United	29	13	8	8	37	27	10	47
7	Southampton	30	12	8	10	38	30	8	44
8	Liverpool	28	12	8	8	43	37	6	44
9	Stoke City	30	12	7	11	32	36	-4	43
10	Chelsea	29	10	10	9	43	39	4	40
11	West Bromwich Albion	29	10	9	10	30	36	-6	39
12	Everton	28	9	11	8	51	39	12	38
13	Bournemouth	30	10	8	12	38	47	-9	38
14	Watford	29	10	7	12	29	30	-1	37
15	Crystal Palace	29	9	6	14	32	39	-7	33
16	Swansea City	30	8	9	13	30	40	-10	33
17	Sunderland	29	6	7	16	35	54	-19	25
18	Norwich City	30	6	7	17	31	54	-23	25
19	Newcastle United	29	6	6	17	28	54	-26	24
20	Aston Villa	30	3	7	20	22	57	-35	16

Event Handlers and Renderers



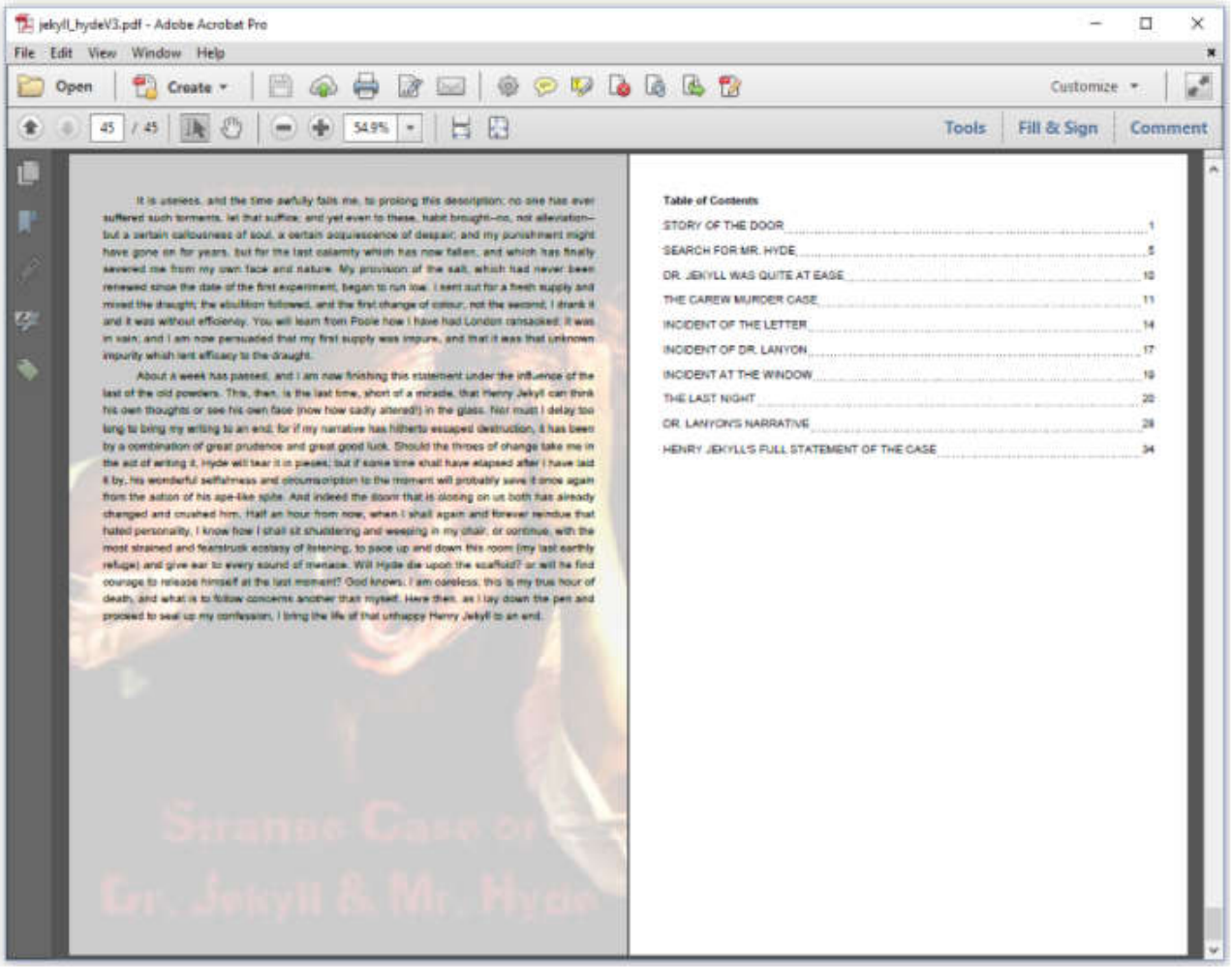
IMDB	Year	Title	Director(s)	Country	Duration
0126875	1908	Dr. Jekyll and Mr. Hyde	Otis Turner	USA	16
0200593	1910	The Duality of Man		UK	5
0126876	1910	Den skæbnesvangre opfindelse	August Blom	USA	17
0002143	1912	Dr. Jekyll and Mr. Hyde	Lucius Henderson	USA	12
0002813	1913	Dr. Jekyll and Mr. Hyde	Herbert Brenon	USA	26
2357384	1913	Dr. Jekyll and Mr. Hyde	Frank E. Woods	USA	
0256936	1913	A Modern Jekyll and Hyde		USA	
0154614	1915	Horrible Hyde	Howell Hansel	USA	
0011130	1920	Dr. Jekyll and Mr. Hyde	John S. Roberson	Denmark	49
0011131	1920	Dr. Jekyll and Mr. Hyde	J.Charles Haydon	USA	40
0011348	1920	Der Januskopf	F.W.Murnau	Germany	107
0022835	1931	Dr. Jekyll and Mr. Hyde	Rouben Mamoulian	USA	98
0211340	1932	Dr. Jekyll and Mr. Hyde	William Vance		10
0033553	1941	Dr. Jekyll and Mr. Hyde	Victor Fleming	USA	113
0151561	1944	Mighty Mouse Meets Jekyll and Hyde Cat	Mannie Davis	USA	6
0039338	1947	Dr. Jekyll and Mr. Mouse	Joseph Barbera, William Hanna	USA	8
0228329	1950	Gentleman Jekyll and Driver Hyde	David Bairstow		8
1336612	1950	The Strange Case of Dr. Jekyll and Mr. Hyde			69
0043515	1951	El extraño caso del hombre y la bestia	Mario Soffici	USA	80
0713926	1951	Dr. Jekyll and Mr. Hyde			30
0045469	1953	Abbott and Costello Meet Dr. Jekyll and Mr. Hyde	Charles Lamont	USA	76

Event Handlers and Renderers

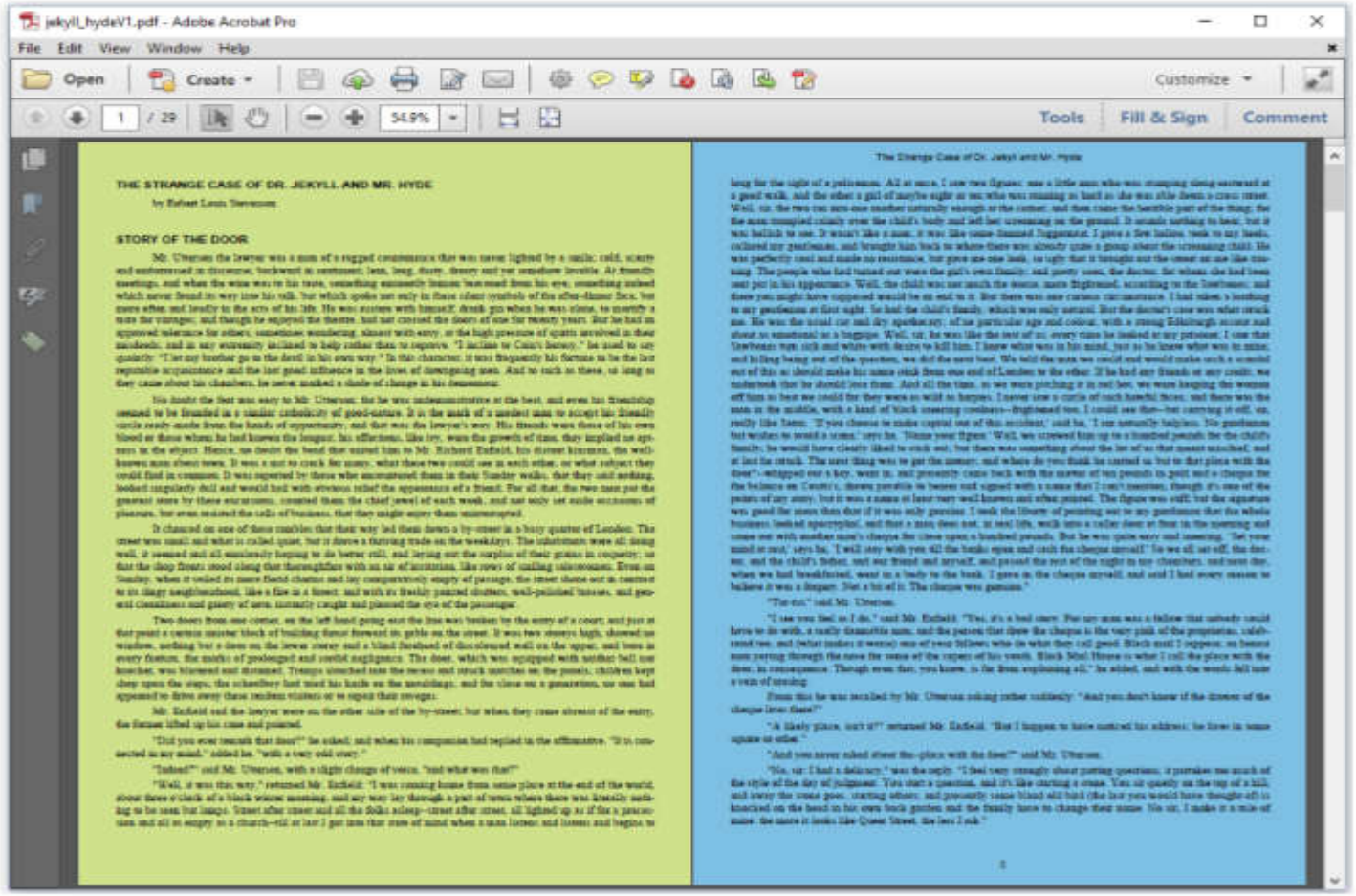
- Four types of events can be triggered:
  - START_PAGE : when a new page is started
  - END_PAGE : right before a new page is started
  - INSERT_PAGE : when a page is inserted
  - REMOVE_PAGE : when a page is removed

- IEventHandler interface
  - handleEvent() method

Event Handlers and Renderers



Event Handlers and Renderers



Annotations

- Allow user interaction
- Not a part of content stream
- Added on top of existing content
- Several types:
  - Text Annotation
  - Link Annotation
  - Line Annoation
  - Mark up Annotation

Annotations

- ISO-32000-2 defines 28 different annotation types
  - Two are deprecated in PDF 2.0
  - iText can add all 26
  - Example of adding Text annotation

```
PdfAnnotation ann = new PdfTextAnnotation(new Rectangle(..))
                            .setColor(Color.GREEN)
                            .setTitle(new PdfString("iText"))
                            .setContents("…");
                            .setOpen(true);
pdf.getFirstPage().addAnnotation(ann);
```

Acroforms

- Allows adding form fields
  - Text fields, List fields, Checkboxes, Radio buttons, Push buttons, Signature fields
- Comparison with HTML form
  - HTML form fields can be resized
  - List contents can be updated on fly
- Fields have fixed place and size
  - More comparable to paper forms
- Not very wieldy for online data collection use-cases

Acroforms

- Typical used when :
  - Form is the equivalent of digital paper
  - Form used as a template

- Form Creation
  - Mostly created manually (Adobe software, LibreOffice, or any other GUI tool)
  - Can be done programmatically

Acroforms

- Creating an acroform

```
PdfAcroForm form = PdfAcroForm.getAcroForm(PdfDocument d,Boolean b);
```

- Fields are added as widget annotations
  - Not a part of content stream

- Filling an existing form

```
Map<String, PdfFormField> fields = form.getFormFields();
fields.get("name").setValue("James Bond");
fields.get("language").setValue("English");
```

Acroforms

- Flattening of forms
  - End user is not allowed to change information in the PDF
  - Removes the user interactivity
  - Widget annotations are replaced with their content

```
form.flattenFields();//flattens all the fields
```

Adding content to Existing PDFs

1. Watermarks
2. Header & Footer
3. Merging PDFs

Watermarks

- Text or Images
- Angle of watermark
- Opacity of watermark
- showTextAligned() method does the heavy loading

```
//define the text to add as a watermark
Paragraph p = new Paragraph("CONFIDENTIAL").setFontSize(60);
canvas.saveState();

//set opacity of watermark on the graphics state
PdfExtGState gs1 = new PdfExtGState().setFillOpacity(0.2f);
canvas.setExtGState(gs1);

//set the location and tilt of watermark on the page
document.showTextAligned(p,width,
height,pageNumber,TextAlign,VerticalAlign,45);
canvas.restoreState();
```

Header and Footer

- Read an existing PDF

```
PdfDocument pdf = new PdfDocument(PdfReader obj,PdfWriter obj);
Document document = new Document(pdf);
```

- Get number of pages & loop over each page

```
int n = pdfDoc.getNumberOfPages();

for (int i = 1; i <= n; i++) {
     PdfPage page = pdfDoc.getPage(i);
     canvas = new PdfCanvas(page);
     // add header and footer
     }
```

- Add Header

```
canvas.beginText()
     .setFontAndSize(..) //set the Font and its size
     .moveText(width,height)
     .showText("I want to believe")
     .endText();
```

Header & Footer

Add Footer

```
//Draw footer line
     canvas.setStrokeColor(Color.BLACK)
          .setLineWidth(.2f)
          .moveTo(width, 20)
          .lineTo(width, 20)
          .stroke();
```

Add Page number

```
//Draw page number in format : X of Y
     canvas.beginText()
          .setFontAndSize(..)
          .moveText(width, 10)
          .showText(String.valueOf(i))// i is the current page number
          .showText(" of ")
          .showText(String.valueOf(n))// n is the total no of pages
          .endText();
```

Merging PDFs

- Merging content of 2 PDF documents
- Use of **PdfMerger** Utility Class
- Partial Merging
  - Selective pages from 2 documents are to be merged

    PdfMerger merge(PdfDocument from, List<Integer> pages)

    PdfMerger merge(PdfDocument from, int fromPage, int toPage)

Merging PDFs

```
//1. Define new merger
PdfMerger merger = new PdfMerger(newPdf);

//2. Add pages from the first document
merger.merge(firstPdf, 1, firstPdf.getNumberOfPages());

//3. Add pages from the second pdf document
merger.merge(secondPdf, 1, secondPdf.getNumberOfPages());

//4. merge and close
firstPdf.close();
secondPdf.close();
newPdf.close();
```

Merging PDFs

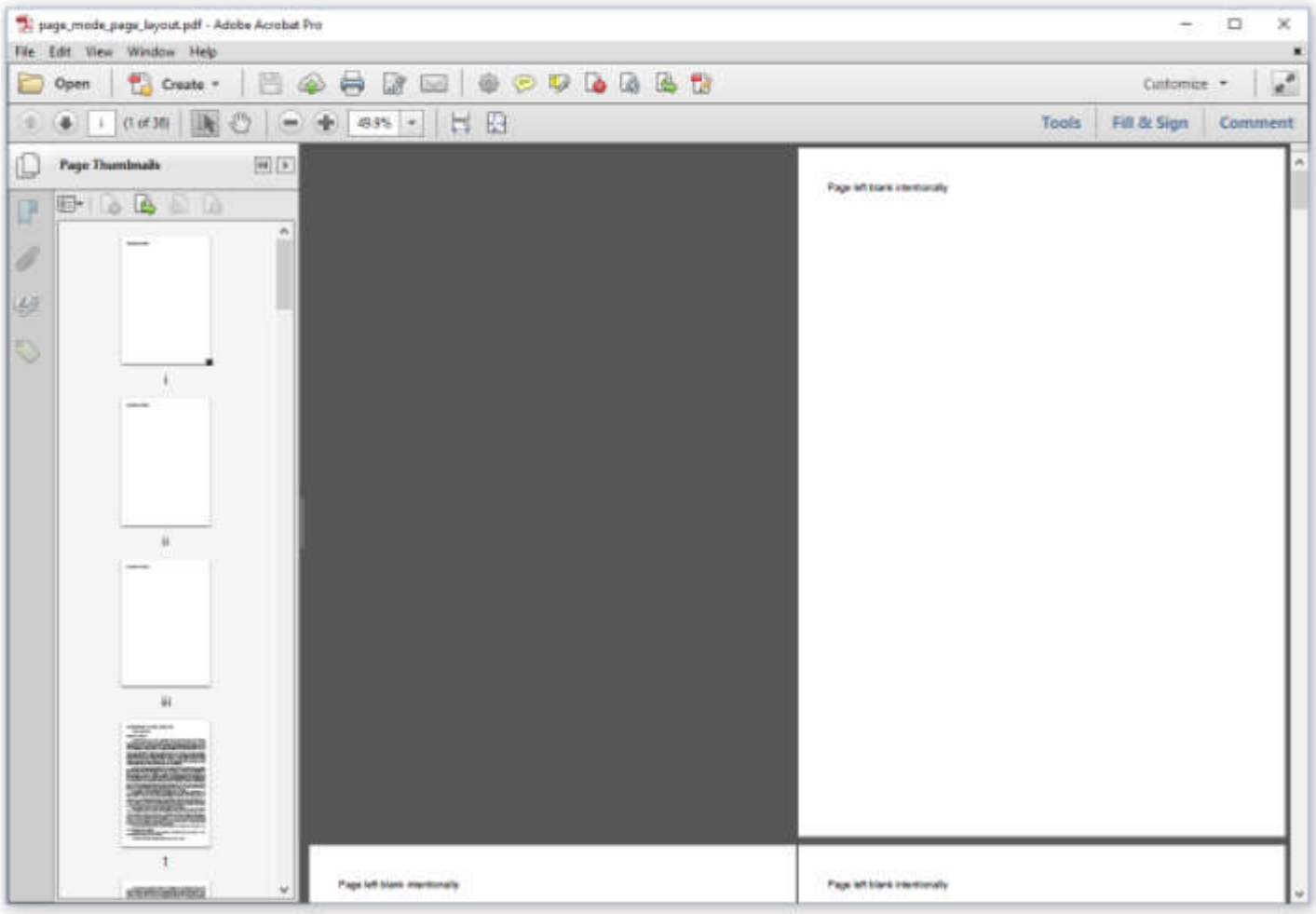
- Merging Forms
  - PDF can contain only one form
  - Merging is done using PdfPageFormCopier Class
  - Duplicate form fields need to be renamed
- Merging “Flattened” Forms
  - Use of smart mode
  - Use of normal mode

Viewer Preferences

Viewer Preferences

- Set default behaviour for document
  - Setting Page Mode : for panel visibility
    - PdfName.UseNone
    - PdfName.UseOutlines
    - PdfName.UseThumbs
    - PdfName.FullScreen
    - PdfName.UseOC
    - PdfName.UseAttachments
  - Setting Page Layout : display of pages of PDF
    - PdfName.SinglePage
    - PdfName.OneColumn
    - PdfName.TwoColumnLeft
    - PdfName.TwoColumnRight
    - PdfName.TwoPageLeft
    - PdfName.TwoPageRight

Viewer Preferences



Viewer Preferences

- Use of **PdfViewerPreferences**
  - Viewer related preferences
    - setFitWindow(), setHideMenuBar(), setHideToolbar(), setCenterWindow(), setDisplayDocTitle(), setNonFullScreenPageMode(), setDirection(), setViewArea() etc
  - Printer related preferences
    - setPrintArea(), setPrintScaling(),setNumCopies() etc

```
pdf.getCatalog().setPageLayout(PdfName.TwoColumnRight);
pdf.getCatalog().setPageMode(PdfName.UseThumbs);
pdf.getCatalog().setViewerPreferences(preferences);
```

Metadata

Metadata

- Stored in the info-dictionary of a PDF
- Contains key-value pairs
- **PdfDocumentInfo** Class corresponds to info-dictionary
- set Title, set Author, set Producer, add Creation Date etc

```
PdfDocumentInfo info = pdf.getDocumentInfo();
info.setTitle("A Strange Case");
```

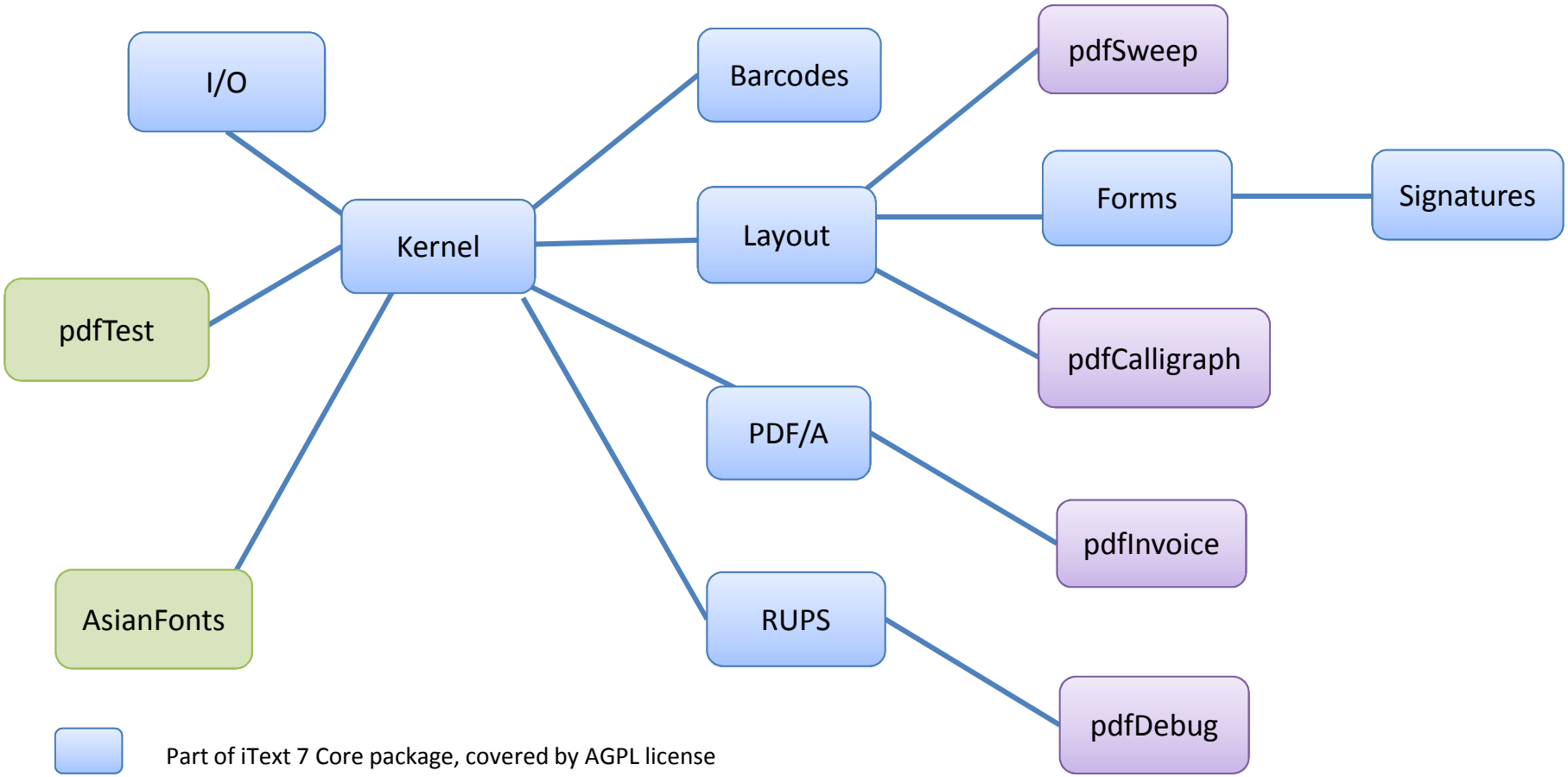
Metadata

- XMP MetaData
  - Metadata store as XML inside a PDF
  - More Flexibility

- Use of **WriterProperties** Class to add XML
  - set Full compression mode or compression levels (0 to 9)
  - set Passwords on the document
  - set PDF version

```
WriterProperties w = new WriterProperties();
w.addXmpMetadata();
w.setFullCompressionMode(true);
w.setStandardEncryption(user, owner, permissions, encryptionAlgo);
```

iText 7 modules



Thank You!

- Log on to http://itextpdf.com/
- Log on to http://developers.itextpdf.com/