PDF Complex Structures

Steven R. Bagley

Introduction

- Looked at the structure of PDF over the last few lectures
- Defined in terms of objects, stored in a file indexed by the cross-reference table
- PDF uses these simple objects to build up some complex structures which represent the document

Today we'll look at those complex structures

Graphs or Trees

- Complex structures made from dictionaries
- That form a tree
- More accurately several overlapping trees
- Technically they form graphs but trees is the term used…
- Root of these trees is the `Catalog` object

Complex Structures

- Dictionaries used to express a complex structure have a `/Type` key
- This identifies the type of the object
- Value is another name object, e.g.
  ```
  /Page
  /Catalog
  ```
- Sometimes have `/Subtype` as well

The `Catalog` object

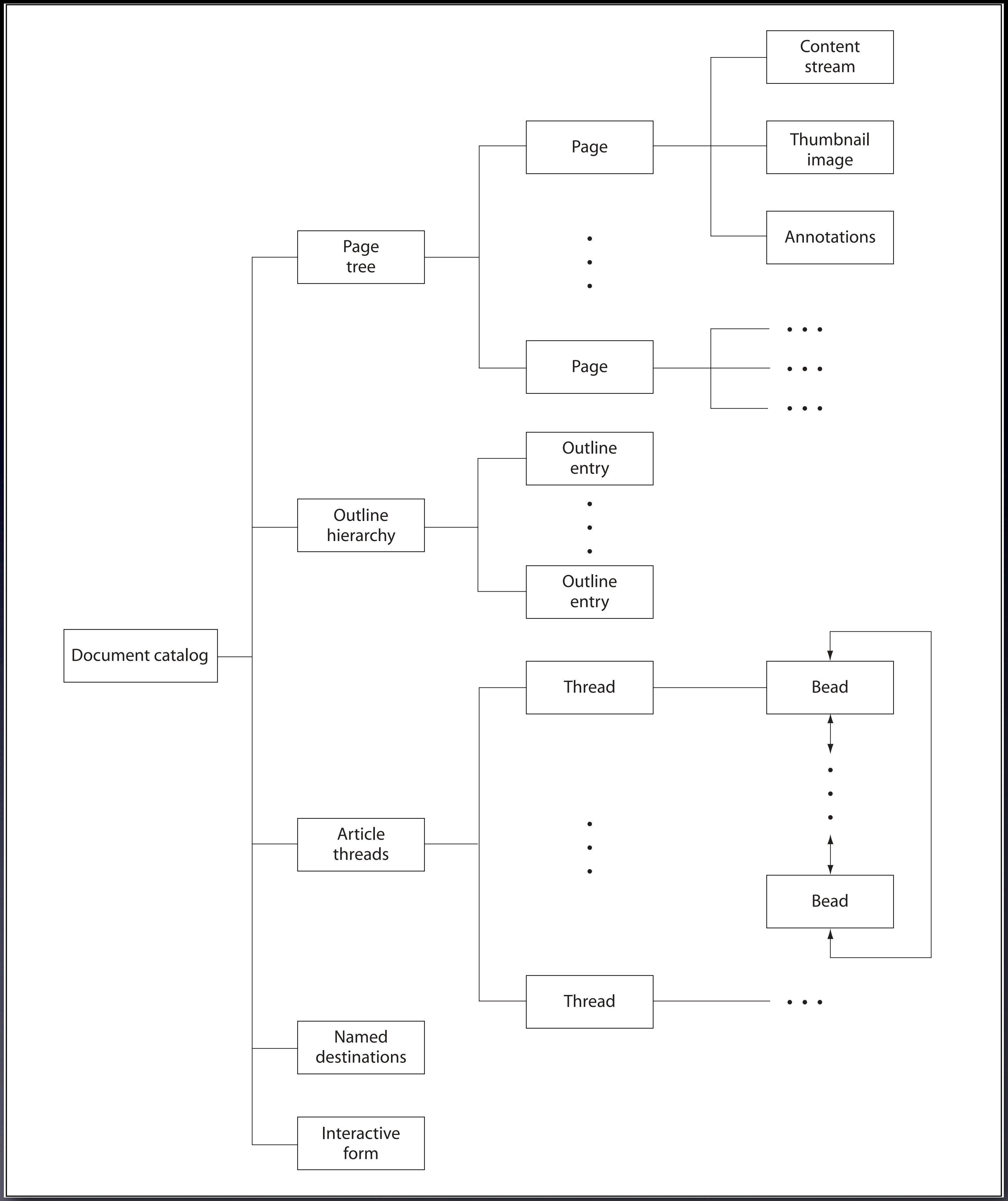
Key	Required	Description
`Type`	Yes	Always the name `/Catalog`
`Pages`	Yes, indirect	Root of the Pages tree, a reference to a Pages object
`StructTreeRoot`	Optional	Reference to the root object of the structure tree (from PDF 1.4)
`AcroForm`	Optional	The document's interactive form dictionary

Lots of other bits that can control the appearance etc
Outlines are bookmarks
Controlling how the PDF should be opened
Won't give every objects serialized form

The `Catalog` object

```
<<
  /Type       /Catalog
  /Pages      2 0 R
  /PageMode   /UseOutlines
  /Outlines   3 0 R
>>
```

Lots of other bits that can control the appearance etc
Outlines are bookmarks
Controlling how the PDF should be opened
Won't give every objects serialized form



Gives the overall structure of the PDF
On top of this modern PDF variants, will have structure trees, XML-based metadata

Pages Tree

- Every PDF has a Pages tree
- Describes all the Pages in it
- Simple tree of Pages objects, with Page objects as leaves
- Supposedly a balanced tree (but not enforced)

Acrobat will rebalance it though

Pages Tree

- Balanced Tree makes it more manageable to process big documents
- Acrobat’s arrays limited to 8191 items
- By using a tree you can hold an infinite number of pages

Page Tree

- Easiest way to understand the Pages tree is to look at the leaves first
- Most of the properties that can be defined on a Page can be propagated up the tree
- These then apply to all its descendants

The `Page` Object

- Represents a single page
- Has to have a MediaBox (defines the boundaries of the page)
- Has to have a reference to its parent
- A list of any resources it uses…

PDF Rectangles

- PDF requires you to specify rectangles in many places (such as MediaBox)
- Always stored as an array of four numbers
- Lower-left, and upper-right corners,
  `[` $\mathtt{ll}_x$ $\mathtt{ll}_y$ $\mathtt{ur}_x$ $\mathtt{ur}_y$ `]`

The `Page` Object

The Minimal page object
Anyone notice what’s missing?

The Page Object

```
<<
   /Type      /Page
   /Parent    12 0 R
   /MediaBox  [0 0 595 842]
   /Resources << >>
>>
```

The Minimal page object
Anyone notice what’s missing?

Page Contents

- A page doesn’t have to have any contents…
- But it usually will…
- In this case, there will be a `/Contents` key
- Can either be a reference to a stream
- Or an array of references to several streams

Page contents

- PDF allows the content description to be split up into multiple streams
- No difference in how its interpreted but can make it easier to generate the content
- Finish with one stream then move to next
- Apps not required to preserve the split

Just preserve the contents

Content Stream

- A series of operands and operators in reverse polish notation
- That describe the contents of the page
- Declarative, unlike PostScript
- Still has a concept of graphics state
- Notionally there’s an order in which the operators can occur

PostScript allows interpretive execution (e.g. for loops) in PDF you can’t

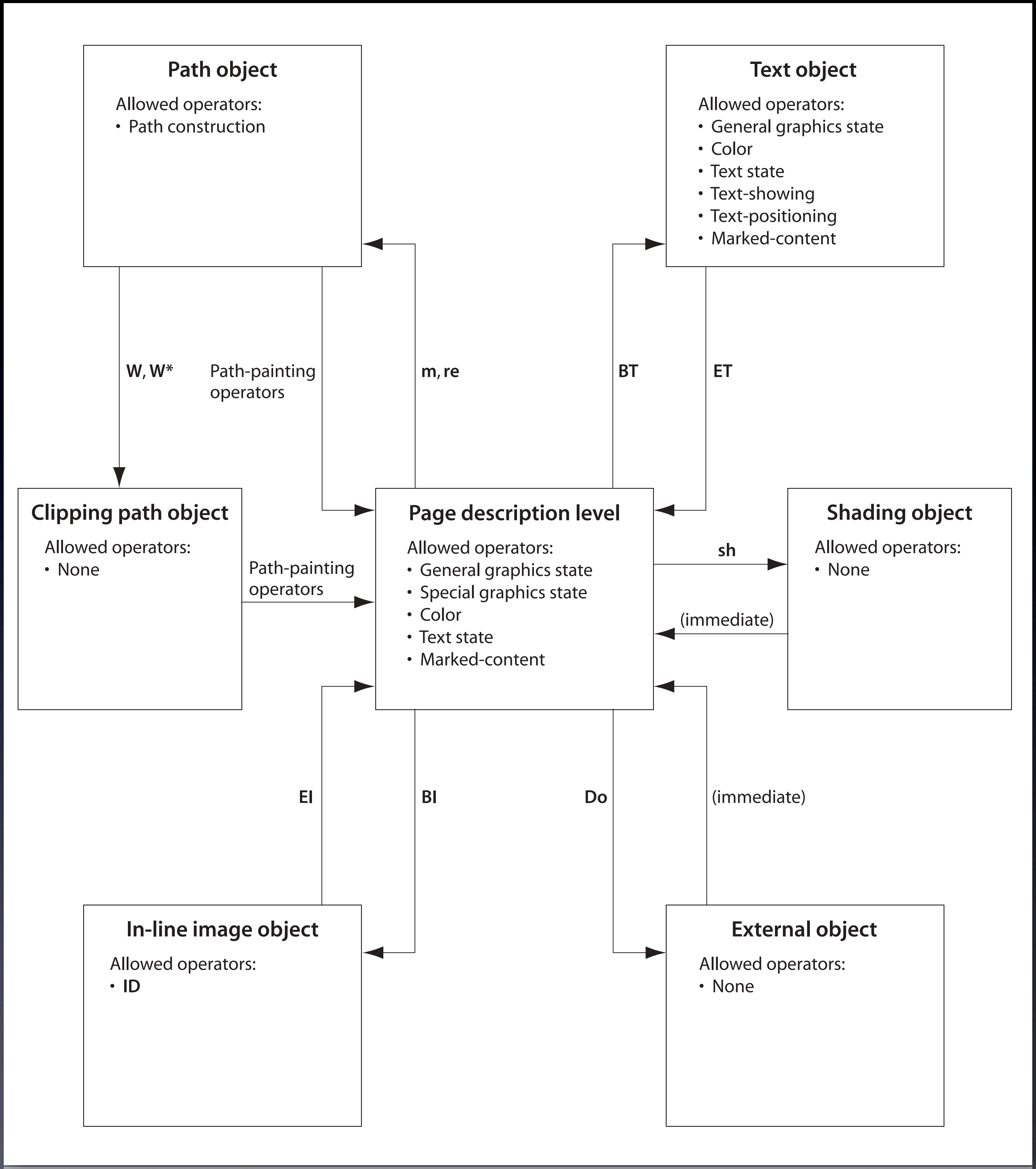
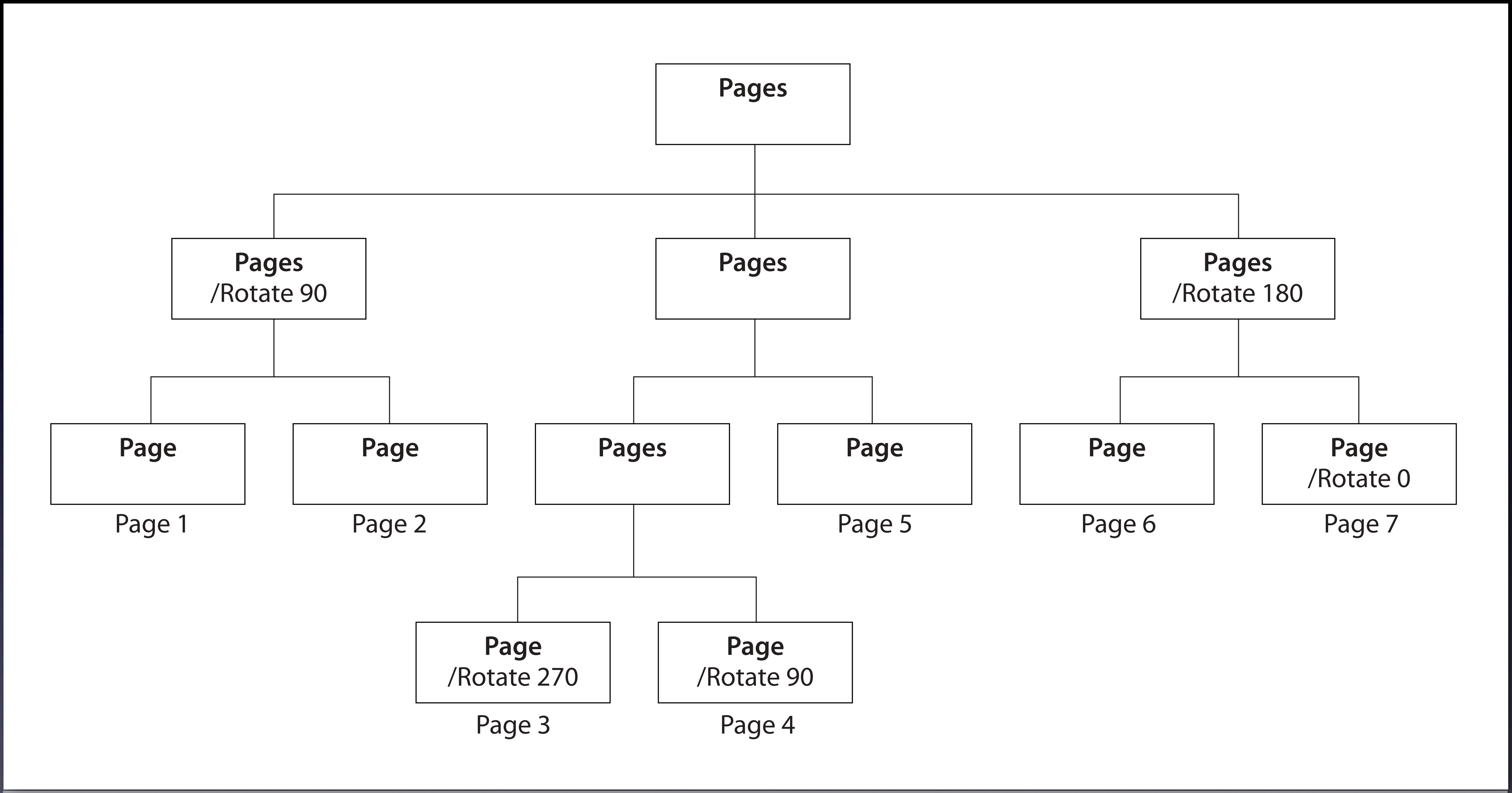


Diagram showing where certain operators are allowed
Acrobat will cope if they are wrong though (annoyingly!)

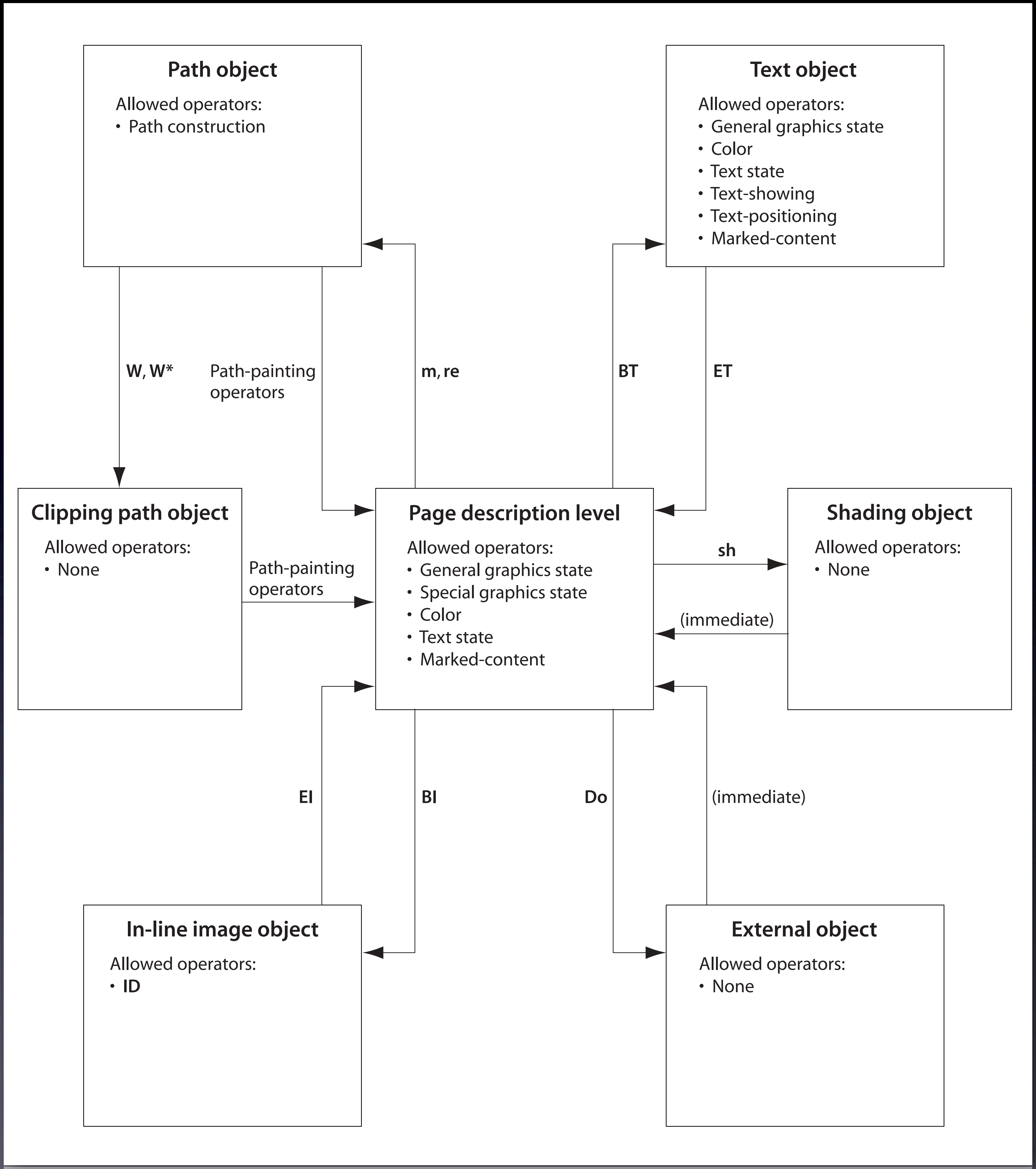
Category	Operators
General graphics state	w, J, j, M, d, ri, i, gs
Special graphics state	q, Q, cm
Path Construction	m, l, c, v, y, h, re
Path Painting	S, s, f, F, f*, B, B*, b, b*, n
Clipping paths	W, W*
Text objects	BT, ET
Text state	Tc, Tw, Tz, TL, Tf, Tr, Ts
Text positioning	Td, TD, Tm, T*
Text showing	Tj, TJ, ', "
Color	CS, cs, SC, SCN, sc, scn, G, g, RG, rg, K, k
Shading Patterns	sh
Inline images	BI, ID, EI
XObjects	Do

Page Properties

- Pages can have various properties
- Seen the MediaBox, also the CropBox, BleedBox, TrimBox, ArtBox
- Rotation (in 90 degree steps)
- Inheritable down the pages tree
- Apply to all children of the `Pages` object



Talk through inheritance



Lets think about how we display some text

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`

```
BT
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state (in this case, font and size)

```
BT
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state

```
BT
/F1 12 Tf
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position (note separate from paths)

```
BT
/F1 12 Tf
```

Unlike PostScript

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position

```
BT
/F1 12 Tf
100 100 Td
```

Unlike PostScript

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position
- Show the Text

```
BT
/F1 12 Tf
100 100 Td
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position
- Show the Text

```
BT
/F1 12 Tf
100 100 Td
(Hello world) Tj
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position
- Show the Text
- End the text object

```
BT
/F1 12 Tf
100 100 Td
(Hello world) Tj
```

Text in PDF

- Text is relatively straightforward
- Start text object with `BT`
- Set the text state
- Set the text position
- Show the Text
- End the text object

```
BT
/F1 12 Tf
100 100 Td
(Hello world) Tj
ET
```

Text in PDF

- However, I've rarely seen text imaged in this way
- More often than the font is selected at size 1pt
- A text matrix set up that scales it to the relevant point size

```
BT
/F1 12 Tf
100 100 Td
(Hello world) Tj
ET
```

```
BT
/F1 1 Tf
12 0 0 12 100 100 Tm
(Hello world) Tj
ET
```

Goto voyeur and give an example
Probably related to the way that PDF keeps track of a line matrix that captures the text matrix at the beginning of a line

Resources

- Note the font was not referred to by a specific font name
- In PostScript, resources are just known about by the VM
- PDF resources must be explicitly linked to a page that uses them
- Name used in the content stream must match an object in the Page's resources

resources are anything external to the content stream referenced from within it
Enables the PDF viewer to ensure it has them all loaded before displaying the pageS
So /F1 must occur in the PAge's resources or its an error

Page Resources

- Also inheritable — resources are first looked for in the local resources dictionary
- Then in the Page’s parent and upwards…
- Error if not found

Page Resources

- Resources dictionary is broken down into different types of resources
- Some are obvious — Fonts, Patterns, ColourSpaces etc.
- Others less so — XObjects, ExtGState
- Also ProcSet, specifies what type of operators used in the PDF

XObjects

- External Objects
- Graphics objects to be imaged on the page
  - Raster images
  - FormXObject
  - PostScript XObject

FormXObject

- Self-contained description of some graphics objects
- Think of it like a PDF sub-routine that can be called to image some graphics
- Appearance can be cached…

External Graphics State

- ExtGState objects allow you to specify some graphics state properties externally
- Dictionary mapping keys to values
- Some state parameters can be set via operators
- Others limited to the graphics state dictionary

Page Resources

- Since the resources map keys to indirect objects
- There’s no need to have lots of instances of the same resource for each page
- You can just point to the same font description

Although it is possible to do that if they use different

Extensibility

- Easy to add features to PDF
- Just add extra keys and objects to the file
- PDF viewers will ignore what they don’t understand
- Although they may also remove/corrupt it if asked to save the document
- Also allows for forwards compatibility…

I had to, erm, circumvent some of Acrobat’s machinations for my COG work

Annotations

- One example are annotations
- Places things on top of the page
- PDF viewer can complete ignore them yet still display the page contents
- Found by following the `/Annots` key in the Page object
- Array of `/Annot` objects

Annotations were there from PDF1.0 but have been extended with new versions ever since

Annotations

- Many different types of annotations
- Text annotations — a la post-it note
- Link annotations — a la hypertext
- File attachments…
- Video and audio
- Sometimes don’t even look like annotations

Annot Object

- All `/Annot` objects will have
  - A `/Subtype` giving the type of annotation
  - A `/Rect` rectangle defining the location of the annotation
  - An optional `/Border` array
  - Optionally specify a FormXObject to define its appearance

Link annotations

- A simple annotation is the link
- Jumps to another location in the PDF
- Has a `/Subtype /Link`
- An optional string `/Contents` which gives some human-readable form
- A destination `/Dest`

Link `Annot` Object

Link Annot Object

```
<<
  /Type       /Annot
  /Subtype    /Link
  /Rect       [36 36 72 72]
  /Dest       [3 0 R /FitR
               -4 399 199 533]
>>
```

Destinations

- Refers to a particular view of the document
- Destinations specified as an array
- First element is the page to go to (indirect reference)
- Then a name object that describes how to display that page in the window

Later PDF versions added support for web links

Destinations

	Parameters	Description
`/XYZ`	*`left top zoom`*	*left* and *top* specify the coordinates of the top-left corner. *zoom* specifies the zoom level. If `null` then current value retained.
`/Fit`	–	Fit the page to the window
`/FitH`	*`top`*	Fit width to window, *top* specifies the y-coordinate of the top of the window.
`/FitV`	*`left`*	Fit height to window, *left* specifies the x-coordinate of the left edge of window.
`/FitR`	*`left bottom right top`*	Fit the specified rectangle to the window. Numerical smaller zoom is chosen…

Incremental Update

- Annotations are an example of something you'd use incremental update to add
- Append annotation objects to PDF
- Append an updated Page object *with same object id*
- Write new cross-reference table…