- Das Format der Type-1-Schriften wurde von Adobe zusammen mit PostScript entwickelt und ist seit 1985 in Nutzung.
- Anders als bei PostScript wurde jedoch das Format der Type-1-Schriften erst 1990 veröffentlicht.
- 1994 wurde das Format als Teil 3 des ISO-Standards 9541 unter dem Titel *Glyph Shape Representation* als Standard akzeptiert.
- Trotz der Konkurrenz durch TrueType sind die Type-1-Schriften bis heute auf dem Markt präsent.
- OpenType ist die Nachfolge-Technologie, die sowohl von Adobe als auch Microsoft unterstützt wird. OpenType unterstützt dabei sowohl die Type-1- als auch die TrueType-Technologie und fügt noch einige Erweiterungen hinzu.
- Literatur: *Adobe Type 1 Format* und Yannis Haralambous: *Fonts & Encodings*, O'Reilly.

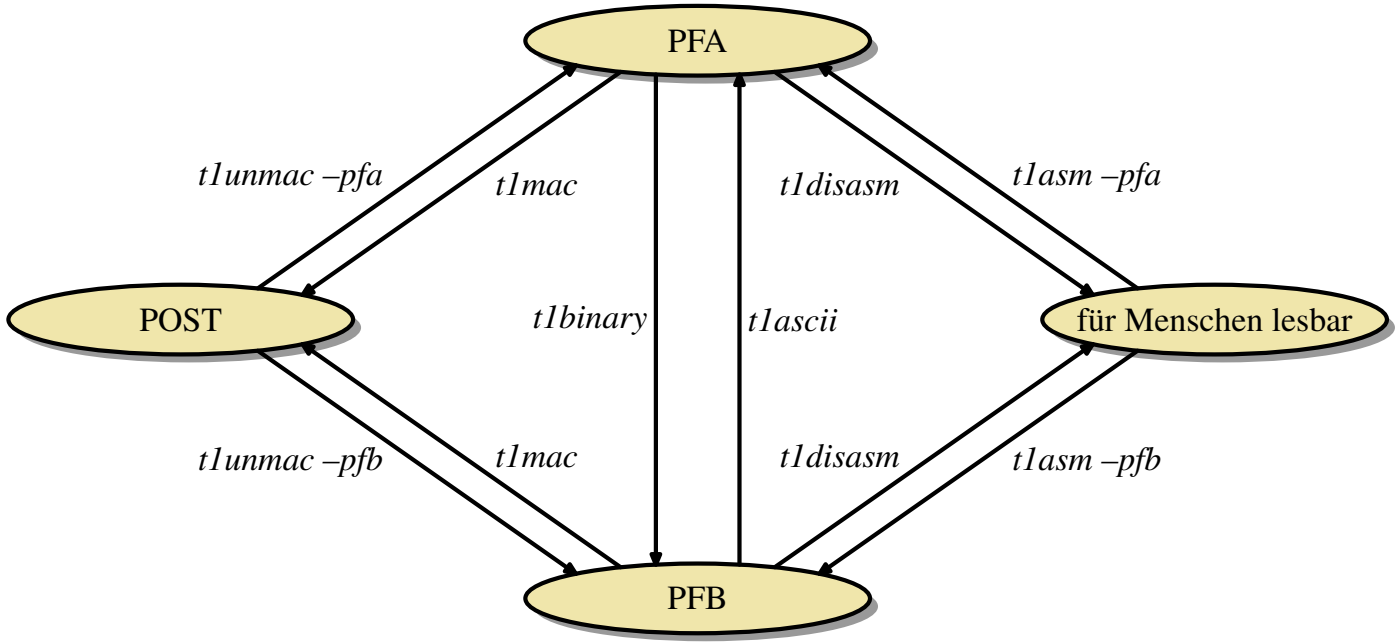
Grundsätzlich ähneln Type-1-Schriften weitgehend den Type-3-Schriften, abgesehen davon, dass

- nicht die Standard-Operatoren von PostScript zur Beschreibung der Kurven verwendet werden, dass
- zusätzlich Hinweise zur Optimierung der Rasterung integriert sind (*hints*) und dass
- das Format größtenteils binär und möglicherweise sogar verschlüsselt ist (auf Basis des *eexec*-Operators, der eine einfache Stromchiffre verwendet, deren Algorithmus und von Adobe verwendeter Schlüssel inzwischen öffentlich sind).

Zu einem Type-1-Schriftschnitt gehören folgende vier Komponenten:

- Ein öffentliches assoziatives Array, das dem eines Type-3-Schriftschnitts ähnelt und in jedem Falle als ASCII-Text abgelegt wird,
- ein privates assoziatives Array, das die Hinweise zur Rasterung aufnimmt und das nur in binärer und möglicherweise sogar nur in verschlüsselter Form vorliegt,
- Prozeduren und
- die Beschreibung der Kurven, die wiederum nur in binärer und möglicherweise sogar in verschlüsselter Form vorliegen.

- Unter Unix sind die Formate PFA (A wie ASCII) und PFB (B wie binär) üblich. Bei PFB werden die binären Teile hexadezimal dargestellt.
- Für den Macintosh gibt es das POST-Format.
- Alle drei Varianten lassen sich dank den Werkzeugen von Lee Hetherington und Claire Connelly ineinander überführen.



- Konvertierungswerkzeuge von `http://www.lcdf.org/type/`

utmr8a.disasm

```
%!PS-AdobeFont-1.0: NimbusRomNo9L-Regu 1.05
% ...
12 dict begin
/FontName /NimbusRomNo9L-Regu def
% ... more contents of the public dictionary ...
currentdict end
currentfile eexec
dup /Private 15 dict dup begin
% ... private dictionary ...
end
readonly put
noaccess put
dup /FontName get exch definefont pop
mark currentfile closefile
cleartomark
```

- Hier und im folgenden werden Auszüge des Times-Roman-Schriftschnitts von URW++ präsentiert, der unter der GPL zur Verfügung steht. Betrachtet wird der Text, den *t1disasm* generiert hat.

Schlüssel	Typ	Bedeutung
FontType	integer	1
FontMatrix	array	Transformations-Matrix für die Kurven innerhalb der Definitionen für die Zeichen; typisch ist ein 1000 × 1000-Koordinatensystem
FontInfo	dict	assoziatives Array mit weiteren Feldern, die den Schriftschnitt beschreiben
Encoding	dict	bildet Werte aus dem Bereich $[0, 255]$ in Namen für die einzelnen Zeichen ab
FontBBox	array	Bounding-Box aller übereinander gezeichneter Zeichen
CharStrings	dict	Kurven der einzelnen Zeichen

Einträge im öffentlichen assoziativen Array 400

Schlüssel	Typ	Bedeutung
FontName	name	Name des Schriftschnitts
PaintType	integer	1 (fill) oder 2 (stroke)
Private	dict	Hinweise zur Rasterung
UniqueID	integer	Eindeutige Zahl zwischen 0 und $2^{24} - 1$
XUID	array	Alternativ zu UniqueID: Eindeutige Folge von Zahlen, wobei die erste von Adobe vergeben wird

utmr8a.disasm

```
/FontName /NimbusRomNo9L-Regu def
/PaintType 0 def
/WMode 0 def
/FontBBox {-168 -281 1000 924} readonly def
/FontType 1 def
/FontMatrix [0.001 0.0 0.0 0.001 0.0 0.0] readonly def
/Encoding StandardEncoding def
/UniqueID 5020931 def
```

- Die Einträge für `/CharStrings` und `/Private` kommen erst nachträglich hinzu.

Einträge im öffentlichen assoziativen Array

utmr8a.disasm

```
/FontInfo 10 dict dup begin
/version (1.05) readonly def
/Notice ((URW)++,Copyright 1999 by (URW)++ ...) readonly def
/Copyright (Copyright (URW)++ ...) readonly def
/FullName (Nimbus Roman No9 L Regular) readonly def
/FamilyName (Nimbus Roman No9 L) readonly def
/Weight (Regular) readonly def
/ItalicAngle 0.0 def
/isFixedPitch false def
/UnderlinePosition -100 def
/UnderlineThickness 50 def
end readonly def
```

- `/FontInfo` gehört mit zum öffentlichen assoziativen Array und ist selbst eines.

- Im Prinzip können die Namen der einzelnen Zeichen, die in `/Encoding` aufgenommen werden, beliebig gewählt werden.
- In Bezug auf PDF kann das ein Problem sein: Wenn interaktiv nach einem Text in einem Dokument gesucht wird, wie kann festgestellt werden, welches Zeichen im Schriftschnitt welchem Zeichen (etwa in Unicode) entspricht?
- Die Lösung besteht darin, dass die Namen der einzelnen Zeichen gewissen Konventionen entsprechen müssen.
- Das wurde ursprünglich festgelegt über ISO 10036 zu Zeiten als Unicode noch nicht zur Verfügung stand. Siehe `http://10036ra.org/`
- Inzwischen wurde die Adobe Glyph List (AGL) eingefroren: `https://github.com/adobe-type-tools/agl-aglfn/blob/master/glyphlist.txt`
- Neue Namen werden entsprechend der *Adobe Glyph List Specification* gebildet, die sich jeweils einer Sequenz von Unicode-Codepoints zuordnen lassen.

utmr8a.disasm

```
currentfile eexec
```

- `currentfile` ist ein Operator, der einen Verweis auf die aktuelle Eingabequelle (mit dem Type-1-Schriftschnitt) zurückliefert.
- `eexec` nimmt den Verweis auf eine Eingabedatei, um ihn die damit referenzierte Eingabe zu entschlüsseln und anschließend auszuführen.
- In einer PFA-Datei ist bis zu diesem Punkt (abgesehen von den ersten 6 Bytes) alles Klartext und erst danach beginnt das eigentliche binäre Format.

Zum assoziativen Array `/Private` gehören folgende Komponenten:

- Globale Hinweise zur Rasterung,
- mehrere historische Parameter,
- einige Operatoren, die für die Prozeduren benötigt werden,
- die Prozeduren selbst (`/Subrs`) und
- weitere Prozeduren (`/OtherSubrs`).

Zu den binären Daten gehört ferner noch `/CharStrings`, das jedoch erst nachträglich in das übergeordnete assoziative Array eingeordnet wird. Es enthält die Kurvendefinitionen der einzelnen Zeichen einschließlich individueller Hinweise zur Rasterung.

utmr8a.disasm

```
/BlueValues [-20 0 450 470 662 682] def
/BlueScale 0.039625 def
/StdHW [30] def
/StdVW [85] def
/StemSnapH [30 38 43 53 60 73] def
/StemSnapV [78 85 91 103 109 115] def
```

- `/BlueValues` spezifiziert die y-Koordinaten für die wichtigsten Höhenlinien.
- Die optionalen Parameter `/BlueScale` und `/BlueShift` spezifizieren, wie weit die Ausrichtung aufgrund der Höhenlinien erfolgt.
- `/StdHW` und `/StemSnapH` spezifizieren die wichtigsten horizontalen Strichstärken. Analog gibt es `/StdVW` und `/StemSnapV` für die vertikalen Strichstärken.

- Die `/BlueValues` spezifizieren Höhenbereiche für den gesamten Schriftschnitt.
- Sie werden typischerweise genutzt, um die Basislinie zu markieren, die x-Höhe und die Versalhöhe.
- Glatt abschließende Versalien wie etwa das „H“ tangieren den Bereich, gehen aber nicht in diesen hinein.
- Bei runden Zeichen (wie etwa beim „O“ oder „o“) oder spitz abschließenden (wie beim „h“) dringt der Pfad in den markierten Bereich hinein (Überschuss), geht aber über die andere Linie nicht hinaus.

utmr8a.disasm

```
/password 5839 def
/MinFeature {16 16} def
```

- Diese Parameter sind entsprechend der Spezifikation von Adobe aus historischen Gründen notwendig.
- Eine weitere Erklärung gibt es nicht dazu. Entsprechend finden sie sich in dieser Form in jedem Type-1-Schriftschnitt.

utmr8a.disasm

```
/RD {string currentfile exch readstring pop} executeonly def
/ND {noaccess def} executeonly def
/NP {noaccess put} executeonly def
```

- Diese Operatoren müssen ebenfalls innerhalb des `/Private`-Arrays definiert werden.
- `/ND` bzw. `/NP` fügen eine Prozedur in ein assoziatives Array bzw. ein reguläres Array ein und beschränken den Zugriff.

utmr8a.disasm

```
/Subrs 251 array
dup 0 {
   3 0 callothersubr pop pop setcurrentpoint return
} NP
dup 1 {
   0 1 callothersubr return
} NP
dup 2 {
   0 2 callothersubr return
} NP
dup 3 {
   return
} NP
% ...
ND
```

- Die Prozeduren sind in einem regulären Array und somit nur über die jeweilige Nummer erreichbar.
- Die Prozeduren 0 bis 3 sind vordefiniert. Der entsprechende Text wird deswegen normalerweise ignoriert.
- Mit `callothersubr` erfolgt ein Aufruf der anderen Prozeduren. Mit `return` endet die Ausführung einer Prozedur. Ein `return` ist zwingend.

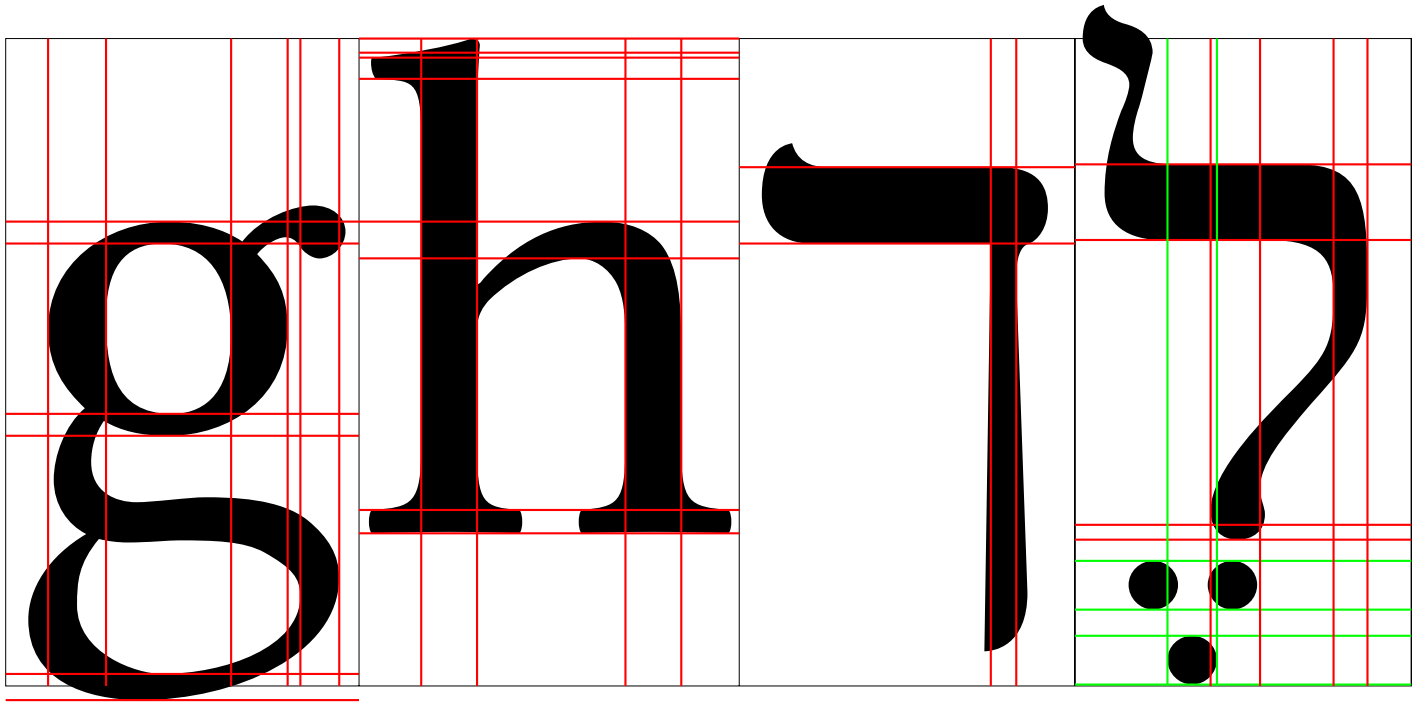
- Der Operator `hsbw` mit den beiden Parametern *sbx* und *wx* spezifiziert mit (*sbx*, 0) den am weitesten links liegenden Punkt der Kurve und den nachfolgenden Weiterbewegungsvektor mit (*wx*, 0).
- Ferner wird die aktuelle Position auf (0, *sbx*) gesetzt. Diese Position geht jedoch nicht in die Kurve selbst ein. Diese muss zwingend mit einer relativen Bewegung beginnen.
- Dies muss der erste Operator sein.
- Alternativ kann `sbw` verwendet werden, falls der Weiterbewegungsvektor eine von 0 abweichende y-Komponente hat.

Operator	Beschreibung
dx dy rmoveto	bewegt sich relativ zur aktuellen Position um (*dx*, *dy*)
dx hmoveto	ist äquivalent zu *dx* `0 rmoveto`
dy vmoveto	ist äquivalent zu `0` *dy* `rmoveto`
dx dy rlineto	Linie mit relativen Koordinaten
dx hlineto	horizontale Linie
dy vlineto	vertikale Linie
dx1 dy1 dx2 dy2 dx3 dy3 rcurveto	Bézier-Kurve, wobei alle Koordinaten relativ zum Ausgangspunkt genommen werden
dx1 dx2 dy2 dy3 hvcurveto	äquivalent zu *dx1* `0` *dx2 dy2* `0` *dy3* `rcurveto`
dy1 dx2 dy2 dx3 vhcurveto	äquivalent zu `0` *dy1 dx2 dy2 dx3* `0 rcurveto`
closepath	Ende einer Kurve; mehrere Kurvendefinitionen sind zulässig; der aktuelle Punkt bleibt bestehen
endchar	Ende einer Zeichendefinition

Operatoren für individuelle Hinweise zur Rasterung 413

Operator	Beschreibung
hstem	Spezifikation eines horizonalen Strichs mit der unteren y-Koordinate und der Höhe
vstem	Spezifikation eines vertikalen Schafts mit der niedrigeren x-Koordinate und der Breite
vstem3	adressiert die Problematik des »m«, indem mit insgesamt 6 Parametern die Angaben für drei einzelne `vstem`-Operatoren zusammengefasst werden.
hstem3	analog zu vstem3, jedoch horizontal

Operatoren für individuelle Hinweise zur Rasterung 414



- Das Beispiel zeigt alle mit *hstem* oder *vstem* markierten vertikalen und horizontalen Schäfte der Buchstaben „g“, „h“, dem Kapf (in der Variante für das Ende eines Worts) und das Lamed mit dem Vokalzeichen Segol (drei Punkte), das als Akzent darunter gesetzt wird.
- Schriftschnitt: *LibertinusT1Math*

utmr8a.disasm

```
2 index /CharStrings 316 dict dup begin
% ...
/a {
   % ...
} ND
% ...
/.notdef {
   % ...
} ND
end
```

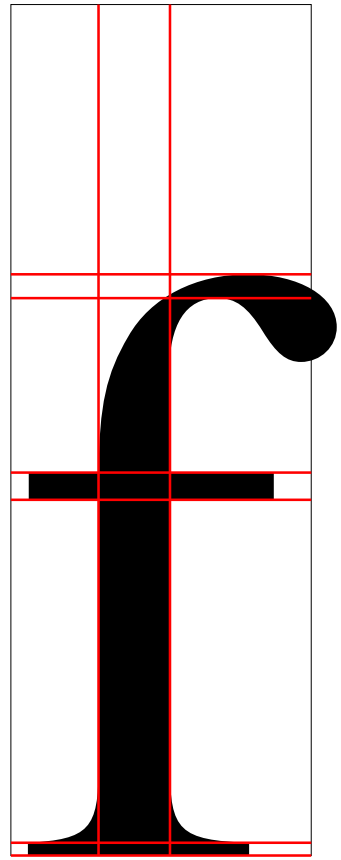
- Da die Definition textuell in der Deklaration von `/Private` eingebettet ist, muss mit `2 index` explizit das übergeordnete assoziative Array referenziert werden (das immer noch auf dem Stack an genannter Position liegt).
- Die einzelnen Prozeduren enthalten jeweils individuelle Hinweise zur Rasterung und die Kurvendefinition. Hierbei dürfen nur die speziellen Operatoren verwendet werden.
- Die Namen müssen mit denjenigen übereinstimmen, die in `/Encoding` referenziert werden.

Definition des Schnitts für den Buchstaben f 416

utmr8a.disasm

```
/f {
   20 333 hsbw
   0 15 hstem 418 32 hstem 655 28 hstem 83 84 vstem
   289 450 rmoveto
   -123 hlineto 116 vlineto
   58 19 31 38 vhcurveto
   21 0 14 -10 18 -29 rrcurveto
   16 -26 12 -10 17 0 rrcurveto
   23 19 18 23 hvcurveto
   36 -44 26 -60 vhcurveto
   -62 0 -53 -27 -26 -46 rrcurveto
   -26 -44 -8 -36 -1 -80 rrcurveto
   -82 hlineto -32 vlineto 82 hlineto
   -314 vlineto
   0 -73 -11 -12 -72 -4 rrcurveto
   -15 vlineto 260 hlineto 15 vlineto
   -82 3 -11 11 0 75 rrcurveto
   314 vlineto 122 hlineto
   closepath endchar
} ND
```

Definition des Schnitts für den Buchstaben f



- PDF steht für *portable document format*.
- PDF folgt der Tradition von PostScript und ist entsprechend geräte- und platformunabhängig.
- Im Gegensatz zu PostScript ist PDF optimiert für die interaktive Präsentation von Dokumenten am Bildschirm.
- (Anders als bei NeWS und Display PostScript geht es wirklich weiterhin um Dokumente und nicht um die Nutzung der Mächtigkeit von PostScript für grafische Benutzeroberflächen.)

PDF erfüllt folgende Anforderungen:

- Anders als beim Drucker, der alle Seiten hintereinander druckt, sollten die Seiten einzeln vom Benutzer abgerufen werden können.
- Der Aufwand, eine Seite darzustellen, sollte nur linear abhängen von der Zahl der darzustellenden Elemente.
- Im Normalfall sollte ein PDF-Dokumente alle Informationen einschließlich aller verwendeten Schriftschnitte enthalten.
- Zusätzliche Navigationshilfen stehen zur Verfügung einschließlich hierarchischer Dokumentstrukturen und Links.
- Animationen für Präsentationen werden unterstützt.

- Bei PostScript gibt es einen globalen Zustand, der von dem Programm sukzessive verändert wird.
- Entsprechend ist zur Darstellung der n-ten Seite es notwendig, die Seiten 1 bis $n-1$ zu erzeugen. Dies ist im allgemeinen Falle unvermeidlich, da die Generierung der vorherigen Seiten den globalen Zustand so verändern kann, dass dadurch die Generierung der n-ten Seite beeinflusst wird.
- Zwar gibt es bei PostScript Strukturierungskonventionen, die die Selektion einer Seite zulassen. Dies beruht aber nur auf den Konventionen, denen nicht entsprochen werden muss und die auch relativ komplex sind. (Darauf beruhen Werkzeuge wie *gv* oder *psselect*.)

- Bei PDF ist über die Sprachdefinition genau geregelt, welche Informationen notwendig sind, um eine Seite zu generieren.
- All diese Informationen können gefunden werden, ohne das gesamte PDF-Dokument durchzulesen. Das ist möglich dank einem Index am Ende eines jeden PDF-Dokuments, über das alle einzelnen PDF-Objekte gefunden werden können.
- Ein globaler Zustand existiert nicht. Entsprechend beginnt jede Seitengenerierung mit einem vordefinierten Zustand.

- PDF kennt keine Schleifen und keine Rekursion.
- Entsprechend hängt der Ausführungsaufwand direkt linear ab von dem Umfang der Beschreibung einer Seite und deren referenzierter Ressourcen wie etwa Schriftschnitte oder gemeinsame (nicht-rekursive!) Prozeduren.
- Das kann allerdings zur Folge haben, dass der Umfang eines PDF-Dokuments volumenmäßig im Vergleich zur äquivalenten PostScript-Quelle zunimmt.
- Grundsätzlich beschleunigt dies jedoch die Darstellung.

- Bei PostScript kam die Möglichkeit, Type-1-Schriften einzubetten, recht spät mit *Level 3*.
- GhostScript unterstützt dies bis heute nicht, d.h. bei einer Konvertierung von PDF auf PostScript gehen die eingebettenen Schriften verloren. Stattdessen werden diese für eine vorgegebene Auflösung gerastert. (Eine Alternative ist hier pdftops, das zum xpdf-Paket gehört.)
- Im Gegensatz dazu ist bei PDF das Einbetten von Schriften der Normalfall — selbst wenn es prinzipiell möglich ist, diese auch wegzulassen.
- Ziel der Portabilität ist es, dass jeder mit einem PDF-Leser die Dokumente betrachten kann, ohne eigene Kopien der Schriften auf seinem Rechner zu besitzen.
- Nur so ist auch eine gute Rasterung möglich, die am Bildschirm anspruchsvoller ist als auf dem Drucker.

- 1993 wurde das erste Referenzdokument zu PDF von Adobe veröffentlicht zusammen mit der ersten Version des Acrobat Readers.
- 1994 begann die Unterstützung von PDF durch GhostScript. 1995 kam *xpdf* hinzu.
- 1995 begann die Entwicklung von PDF/X, einer Teilmenge von PDF, die für Druckvorlagen geeignet ist.
- 1997 kam PostScript Level 3 heraus, d.h. erst dann konnten in PostScript Type-1-Schriften eingebettet werden.
- 2001 und 2002 wurde PDF/X in die ISO-Normen 15929 und 15930 übernommen.
- 2005 wurde mit PDF/A eine Teilmenge von PDF für Archivierungen als ISO 19005-1 standardisiert.
- 2006 kam PDF 1.7 heraus – das ist die zur Zeit aktuelle Fassung.
- 2008 wurde PDF 1.7 als ISO-Standard akzeptiert: ISO 32000-1:2008.

- PDF ist ein Datenformat von Programmen für Programmen. Es ist (anders als PostScript) nicht dafür vorgesehen, von Menschen erstellt, verändert oder gelesen zu werden.
- Im Prinzip ist PDF eine lineare Repräsentierung einer komplexen Datenstruktur, die in einzelne sich untereinander referenzierende PDF-Objekte zerlegt worden ist.

- PDF-Objekte haben einen Typ, wobei alle Typen der Sprache PostScript eingeschlossen sind: Boolean, ganze Zahlen, Gleitkommazahlen, Zeichenketten, Arrays und assoziative Arrays (*dictionaries*).
- Hinzu kommen Streams und das *null*-Objekt.
- PDF-Objekte können von anderen PDF-Objekten referenziert werden. Damit dies möglich ist, können PDF-Objekte mit einem Tupel bestehend aus einer ganzen Zahl und einer Versionsnummer (beginnend mit 0) definiert werden.
- Alle so referenzierbaren PDF-Objekte werden *indirekte Objekte* genannt. Sie müssen allesamt im Index am Ende der PDF-Datei aufgeführt werden.
- Die Versionsnummer erlaubt das spätere Ändern von PDF-Objekten, ohne die gesamte PDF-Datei neu erzeugen zu müssen.

- Die folgenden Seiten stellen ein minimales PDF-Beispiel vor, das nur eine Seite mit dem Text »Hello World« produziert.
- Das Beispiel beruht auf dem Beispiel G.2 aus dem PDF-Referenzmanual von Adobe. Im Vergleich zum Original wurden einige Teile eliminiert, die ab PDF-1.4 obsolet sind.
- Das Referenzmanual ist unter `http://www.adobe.com/devnet/pdf/pdf_reference.html` zu finden.

Deklaration einer Schrift als PDF-Objekt

HelloWorld.pdf

```
5 0 obj
   << /Type /Font
      /Subtype /Type1
      /Name /F1
      /BaseFont /Helvetica
      /Encoding /MacRomanEncoding
   >>
endobj
```

- Mit **obj** und **endobj** wird die Definition eines PDF-Objekts geklammert.
- Vor **obj** stehen die Objekt- und die Versionsnummer. Erstere ist typischerweise fortlaufend, letztere im Normalfall 0. (Nur bei nachträglich veränderten PDF-Dateien entstehen Objekte mit höheren Versionsnummern.)
- Dieses PDF-Objekt definiert eine Schriftschnitt-Ressource, die bei einzelnen Seiten genutzt werden kann.

HelloWorld.pdf

```
3 0 obj
   << /Type /Page
      /Parent 2 0 R
      /MediaBox [0 0 612 792]
      /Contents 4 0 R
      /Resources << /Font << /F1 5 0 R >> >>
   >>
endobj
```

- Dieses PDF-Objekt ist die Datenstruktur für eine Seite.
- Seiten sind in PDF in einem Baum hierarchisch geordnet.
- Diese Hierarchie hat nichts mit Kapiteln oder Abschnitten zu tun, sondern dient nur der Vererbung von Einstellungen. Normalerweise ist dieser Baum ziemlich flach mit einer Wurzel, die alle einzelnen Seiten als Kinder hat.

HelloWorld.pdf

```
3 0 obj
   << /Type /Page
      /Parent 2 0 R
      /MediaBox [0 0 612 792]
      /Contents 4 0 R
      /Resources << /Font << /F1 5 0 R >> >>
   >>
endobj
```

- `2 0 R` referenziert das PDF-Objekt, das mit der identifizierenden Sequenz `2 0 obj` definiert ist.
- Das Feld `MediaBox` spezifiziert die Dimension der Seite (hier US-Letter-Format), `Contents` referenziert das PDF-Objekt mit der Seitenbeschreibung und `Resources` zählt die für die Seitengenerierung benötigten Ressourcen auf. Hier wird nur der zuvor deklarierte Schriftschnitt unter dem Namen `F1` eingebunden.

HelloWorld.pdf

```
4 0 obj
   << /Length 68 >>
stream
   BT
      /F1 24 Tf
      100 100 Td
      (Hello World) Tj
   ET
endstream
endobj
```

- Eine Seitenbeschreibung besteht aus einem Stream-Objekt, das aus einem assoziativen Array, das mindestens die Länge in Bytes spezifizieren muss, und dem eigentlichen Stream-Inhalt besteht.
- Der Stream-Inhalt wird von den Schlüsselworten **stream** und **endstream** eingegrenzt.

HelloWorld.pdf

```
BT
   /F1 24 Tf
   100 100 Td
   (Hello World) Tj
ET
```

- Mit **BT** und **ET** wird ein Textobjekt geklammert. Textobjekte können nicht verschachtelt werden und verwalten einen lokalen Zustand mit beispielsweise dem aktuellen Schriftschnitt.
- Der Operator **Tf** entspricht dem **selectfont**-Operator in PostScript und wählt eine Schrift aus. Die Schrift muss in den zur Seite gehörenden Ressourcen explizit registriert sein.
- Der Operator **Td** entspricht in etwa dem **moveto**, bezieht sich aber nur auf die Positionierung von Texten.
- **Tj** entspricht dem **show**-Operator in PostScript. Anders als in PostScript berücksichtigt dieser aber mehr Statusparameter.

Wurzel der Seitenhierarchie

HelloWorld.pdf

```
2 0 obj
   << /Type /Pages
      /Kids [3 0 R]
      /Count 1
   >>
endobj
```

- Dieses PDF-Objekt ist die Wurzel der Seitenhierarchie.
- Unter `Kids` werden die einzelnen Seiten über entsprechende Referenzen (wie etwa hier `3 0 R`) aufgezählt.
- `Count` spezifiziert die Zahl der Blattknoten, die direkt oder indirekt über diesen Baum erreichbar sind.

HelloWorld.pdf

```
1 0 obj
   << /Type /Catalog
      /Pages 2 0 R
   >>
endobj
```

- Der Katalog verweist auf alle globalen Datenstrukturen des PDF-Dokuments.
- In diesem einfachen Beispiel findet sich nur der Verweis auf die Wurzel der Seitenhierarchie.
- Zusätzlich denkbar wären Navigationshierarchien, Hyperlinks, Benennungsregeln für die einzelnen Seiten, Voreinstellungen für Betrachter etc.

HelloWorld.pdf

```
xref
0 6
0000000000 65535 f
0000000009 00000 n
0000000071 00000 n
0000000147 00000 n
0000000304 00000 n
0000000425 00000 n
```

- Der Index beginnt mit dem Schlüsselwort **xref**, gefolgt von einer oder mehreren zusammenhängenden Bereichen.
- Jeder Bereich beginnt mit zwei ganzen Zahlen, die die niedrigste Objektnummer und die Zahl der Einträge spezifizieren. Dieser Bereich nennt dann fortlaufend die Byte-Positionen der entsprechenden PDF-Objekte. In diesem Beispiel beginnt es mit der Nummer 0 und es sind nur 6 PDF-Objekte deklariert.

HelloWorld.pdf

```
xref
0 6
0000000000 65535 f
0000000009 00000 n
0000000071 00000 n
0000000147 00000 n
0000000304 00000 n
0000000425 00000 n
```

- Der Eintrag zu einem PDF-Objekt besteht aus genau 20 Bytes, bestehend aus
  - einer 10-stelligen Byte-Position (mit führenden Nullen), einem Leerzeichen,
  - einer fünf-stelligen Versionsnummer, einem weiteren Leerzeichen,
  - dem Buchstaben `n` (belegt) oder `f` (frei) und
  - einem Zeilentrenner aus zwei Zeichen (CR LF).

HelloWorld.pdf

```
trailer
   << /Size 6
      /Root 1 0 R
   >>
startxref
564
%%EOF
```

- PDF-Dateien werden von hinten beginnend gelesen.
- Dort ist ganz am Ende bei `startxref` die Byte-Position des Index angegeben.
- Unmittelbar davor findet sich die `trailer`-Spezifikation, die die Zahl der PDF-Objekte insgesamt angeben und auf das Wurzel-Objekt (den Katalog) verweisen muss.