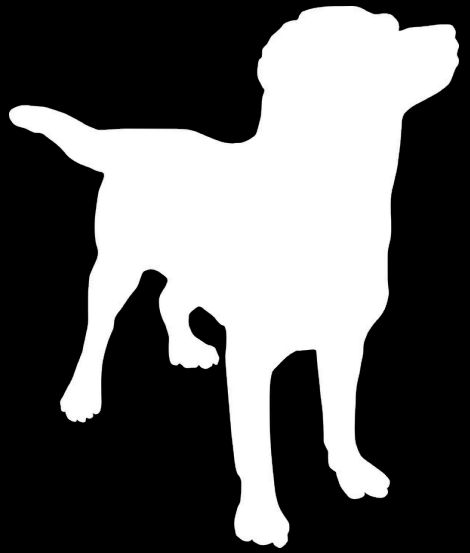
PAL: PDF Automation Language



The Team

Anshuman Singh

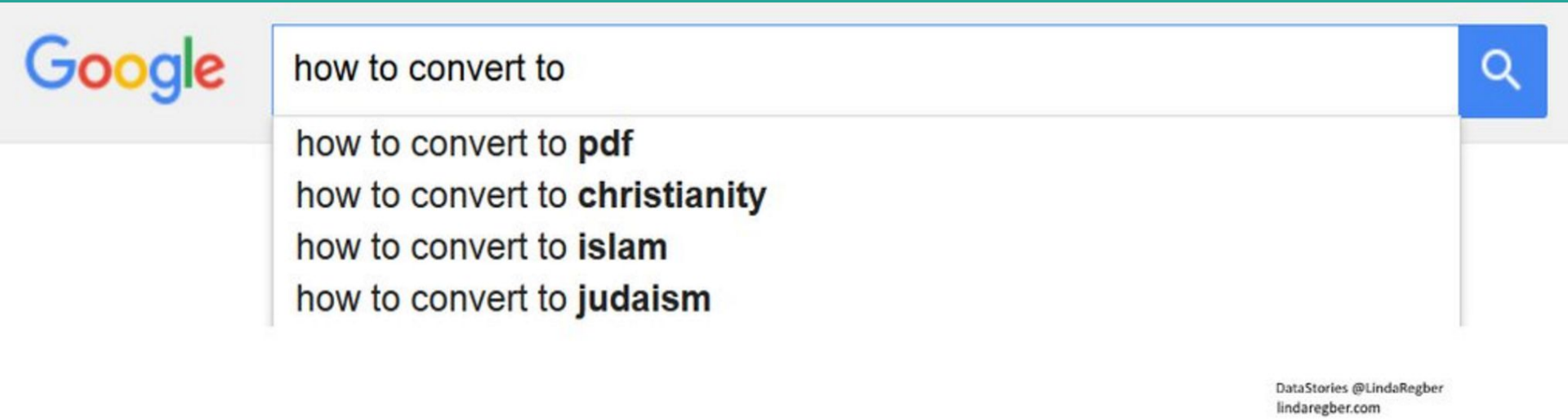
Diksha Vanvari

Vinay Gaba

Viral Shah

Why does PDF creation require its own programming language?

That's Why!



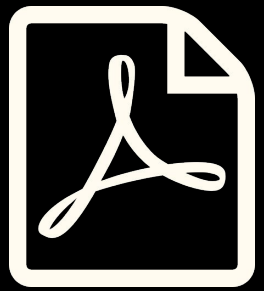
DataStories @LindaRegber · 14h

PDF is now the world's most popular religion.

549 497

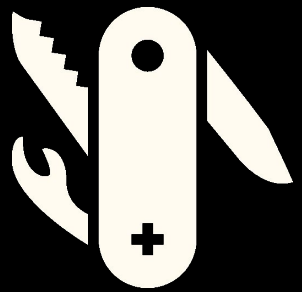
Nuts & Bolts

Compiles to Java



Multiple PDF Operations

Support for Map/List



Useful Standard Library

Objectives

- Make it easy to generate pdfs from strings, textfiles, existing pdfs and tables (charts)
- Find the right balance between a programming language and a markup language
- Provide granular(line level) control to the user
- At the same time, provide high level constructs in the Standard Library which take away the pain of writing the pdf at a granular level.
  - Paragraph level - write_paragraph(...)
  - Page level - write_column_layout(...)
  - PDF level - write_pages(...)

Speak the Language! Woof

Syntax

Comments

```
#This is a single line comment

/*
This
is
a
multi
line
comment
*/
```

Loops

```
for(i=0;i<5;i=i+1){

     awesome = awesome + 1;

}

while(i<5){

     awesome = awesome * 5;

}
```

Primitives and Predefined Constructs

General Primitives

```
intVar:int = 42;

stringVar:string = “COMS 4115”;

boolVar:boolean = false;

floatVar:float = 4.5;
```

PDF Primitives and Constructs

```
pdfVar:pdf;

pageVar:page;

tupleVar:tuple(pdfVar,pageVar);

lineVar:line(s,f,sz,x,y,w);

imageVar:image(s,h,w,x,y);
```

List Data Type

List Operations

```
intList : list int;

intList += 5;

intList -= [0];

intList[0] = 4;
```

List Type Inference

```
int_l : list int;

int_l_l : list list int;

int_l_l_l : list list list int;

List of ‘int’ -> ‘AA”
List of ‘AA’ -> ‘AI’
List of ‘AI’ -> ‘AJ’
```

Map Data Type

Map Operations

```
int_sl_m : map int,list string;

m += 5, “PAL”;

m -= 5;

s : string = m:=4;
```

Map Type Inference

```
int_sl_m : map int,list string;

List of ‘string’ -> ‘AB’
Map of int,string -> int, ‘AB’
```

Primitive Functions

```
readtext(...) / readtable(...)

loadpdf(...) / split(...)

drawpiechart(...) / drawbarchart(...)

substr(...) / length(...)

getpages(...)
```

Standard Library

```
write_paragraph(...)

write_two_column_layout(...)

write_three_column_layout(...)

write_4grid_layout(...)

write_pages(...)
```

Let’s get our hands dirty, shall we?

—

Hello World!

```
main(){

pdfVar : pdf;

pageVar : page;

pageVar = pdfVar . pageVar;

tupleVar : tuple(pdfVar,pageVar);

lineVar : line ("Hello World!", "TIMES_ROMAN" , 12 , 100, 700, 500);

tupleVar = tupleVar . lineVar;

renderpdf(pdfVar,”helloworld.pdf”);

}
```

3 Lines to generate a multi-page PDF

```
import ("stdlib.pal");

main(){

  stringVar:string = readtextfile(filepath);
  pdfVar:pdf = write_pages(stringVar,14,"TIME_ROMAN","THREE_COLUMN");
  renderpdf(pdfVar,"multipages.pdf");

}
```

80%*

That’s the amount of lines PAL helps you reduce over Java

* - on average

Demo Time!