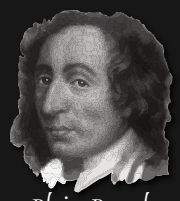
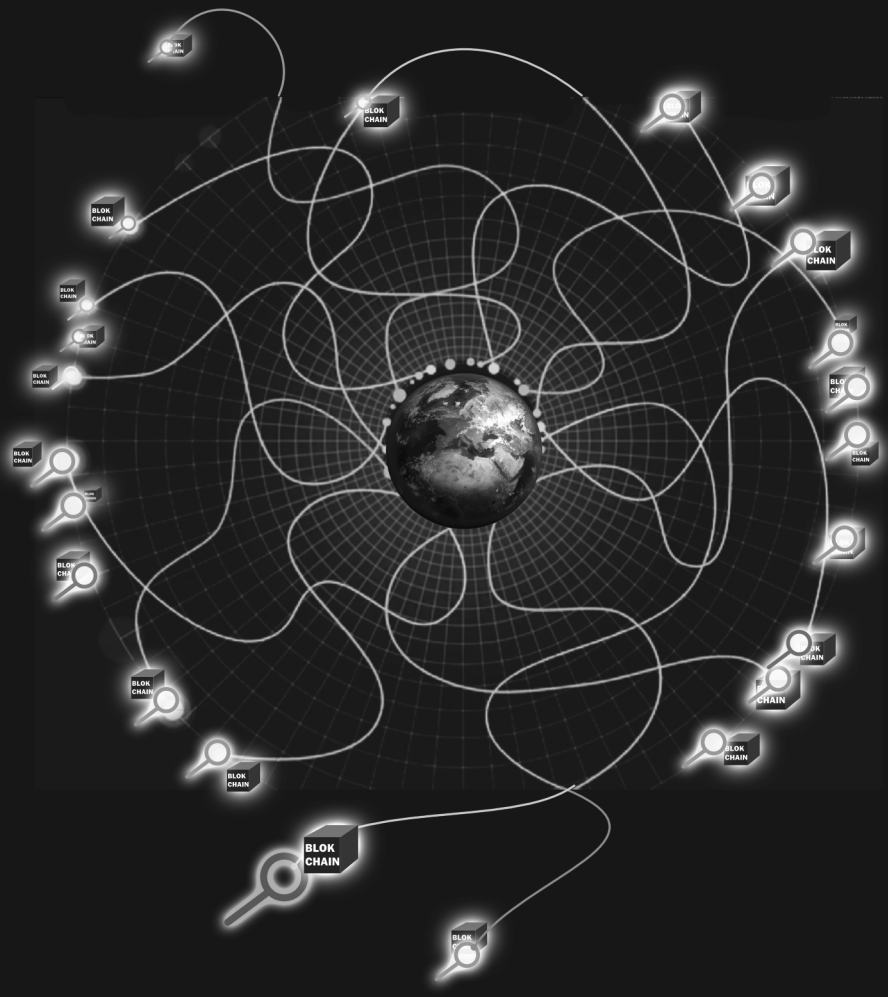
BLAISE PASCAL MAGAZINE109

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux



Blaise Pascal



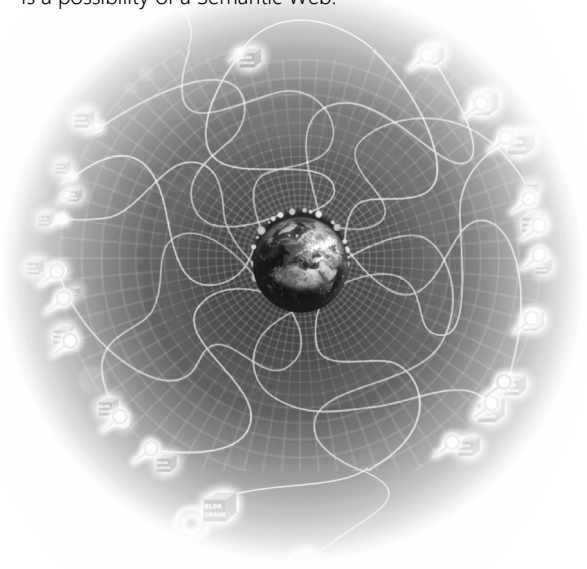
The semantic web3 is coming?
BlockChain
Image to text API
Rounding numbers in FastReport
An integer square root algorithm
Factors and factorising
An integer cube root algorithm
Pseudo random and true Random numbers:
- Quantum computers have now been used to be directly combined with normal machines for true random numbers
Indexing your pdf files for quick search within pages / whole document
Debugging in Lazarus/FPC


Blaise Pascal

CONTENT

ARTICLES

WEB3
Web 3.0 aims to transform the internet into a more regimented version?
There is a lot of excited interest around the idea of a decentralised version of the internet that would give ordinary users more power. And on top of that there is a possibility of a Semantic Web.



ADVERTISING


Niklaus Wirth

Pascal is an imperative and procedural programming language, which Niklaus Wirth designed (left below) in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator: Blaise Pascal (see top right).

Publisher: PRO PASCAL FOUNDATION in collaboration © Stichting Ondersteuning Programmeertaal Pascal

CONTRIBUTORS

Stephen Ball
http://delphiaball.co.uk
DelphiABall

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt
,michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
mail: David @ davdata.nl

Benno Evers
b.evers @
everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger @ flixments.com

Mattias Gärtnernc-
gaertnma@netcologne.de

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni
www.andreamagni.eu andrea.
magni @ gmail.com
www.andreamagni.eu/wp

Paul Nauta PLM Solution
Architect CyberNautics
paul.nauta @ cybernautics.nl

Kim Madsen
www.component4developers.com
kbmMW

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek
- Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Anton Vogelaar
ajv @ vogelaar-electronics.com

Danny Wind
dwind @ delphicompany.nl

Jos Wegman
Corrector / Analyst

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to editor@blaisepascal.eu

Subscriptions can be taken out online at www.blaisepascal.eu or by written order, or by sending an email to office@blaisepascal.eu
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to: **ABN AMRO Bank Account no. 44 19 60 863** or by credit card or PayPal
Name: Pro Pascal Foundation (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Ondersteuning Programmeertaal Pascal)
Subscription department Edelstenenbaan 21 / 3402 XA IJsselstein, Netherlands Mobile: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Member of the Royal Dutch Library KB KONINKLIJKE BIBLIOTHEEK Member and donor of WIKIPEDIA

Subscriptions (2022 prices)		**Internat. excl. VAT**	**Internat. incl. 9% VAT**	**Shipment**	**TOTAL**
Printed Issue (8 per year)	**±60 pages :**	**€ 200**	**€ 218**	**€ 130**	**€ 348**
Electronic Download Issue (8 per year)	**±60 pages :**	**€ 64,20**	**€ 70**		

From your editor

Dear Reader,

It's spring at last, and for me that means sunshine and joy in the garden.
But alas: I have so much to write about and so many new subjects to tell you about that I have not had time to do so.

For many of my authors, I bought a very interesting book that I recommend you read: **The Mind of a Bee**, by Lars Chittka, published by `press.princeton.edu`.
It's fascinating what such a small creature is capable of. You'll be surprised.

What surprised me most was the now ubiquitous ChatGPT. Already version 4.
I find it a very helpful and inspiring tool.
But that's about it.
All I hear from people is that it is very intelligent.
First of all, whatever it is, it's not intelligent. It is a very large algorithm that seems to have some kind of very large knowledge.
Not surprising when you have the whole internet at your fingertips.
And that's where it goes wrong: it is very often far from reality or truth.
It even makes up answers or stories.
Which is at least amusing and shows that it is a toy made by humans.
Nothing is alien to us. Not even ChatGPT.
Play with it and have fun.

About fun: I have just started writing for the next book we want to present: **Learning to Code for the Web with PAS2JS**.
We already have 10 chapters and maybe more to come.
In this book we emphasise simplicity for beginners. It is meant for starters.
Of course you need to have some interest and knowledge of HTML: that helps, but we want to make it as easy as possible for you, so that you can even build your own server with it.
Once you get the hang of it, you will be very encouraged: sheer fun to make a website with PAS2JS and Pascal.
Once the book is finished, we will do some events where you can learn in practice. I will let you know.

I wrote a few lines about the Web3 as well as the Semantic Web.
It is controversial, but maybe we should give it a chance:
If it makes the Web a safer place, I would love that.But it should not become something like Big Brother.

David Dirkse has again produced some articles that might be very interesting for beginners in mathematics.
I really like his examples and he's becoming very famous for them.
Now he has created three of them for you.

Michael van Canneyt and I are working very hard on the **Lazarus PDF Kit**.
It is constantly growing and getting better. Read the article and you will understand that PDF is not an easy thing.

Last but not least, and very interesting, Max Kleiner has written about how to decipher a wine bottle label.
Very helpful if you want to get drunk with knowledge.

Have fun reading…

Detlef

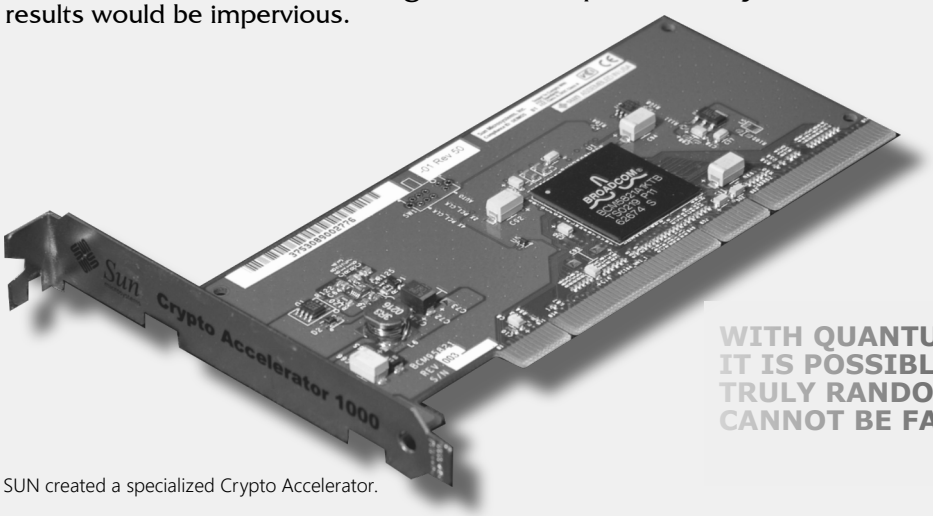
From our technical advisor Jerry King

"Ladies and gentleman, due to flying into clouds full of data, we will be experiencing some turbulence. Please return to your seats and fasten your seat belts."

QUANTUM COMPUTERS HAVE NOW BEEN USED FOR THE FIRST TIME TO BE DIRECTLY COMBINED WITH "NORMAL" MACHINES.

All current techniques for generating the random numbers required for secure communications have some flaw, but current quantum computers may be able to generate random numbers that cannot be faked.
For instance, Edward Snowden's leaks in 2013 revealed that the US government had installed a flaw in the US National Institute of Standards and Technology's random number generator that allowed the output of the generator to be predicted.

All random number generators in use today are pseudo random – they are so hard to predict that their output seems random. But assaults are conceivable. However, a number generator that produces truly random results would be impervious.



SUN created a specialized Crypto Accelerator.

WITH QUANTUM TECHNIQUES, IT IS POSSIBLE TO GENERATE TRULY RANDOM NUMBERS THAT CANNOT BE FAKED OR PREDICTED.

TRULY RANDOM OR PSEUDO RANDOM explained

PSEUDO RANDOM

A RANDOM NUMBER can be faked if the method used to generate the number is not truly random. If a predictable algorithm or formula is used to generate a number, it may appear random, but it can be reproduced by anyone who knows the algorithm or formula. This means that the number is not truly random, but rather pseudo-random.

PSEUDO-RANDOM number generators are commonly used in computer programs to generate random numbers. These generators use mathematical formulas or algorithms to produce a sequence of numbers that appear random, but are actually deterministic. Since the algorithm used to generate the numbers is known, it is possible for someone to reproduce the sequence of numbers by reverse-engineering the algorithm.

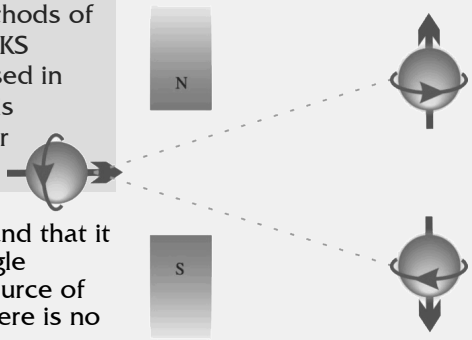
To generate truly random numbers, a source of randomness is required, such as atmospheric noise or radioactive decay. These sources are used to create a seed value for a random number generator, which then uses the seed to produce a sequence of random numbers. While it is still possible for someone to predict the sequence of numbers if they know the seed, the unpredictability of the seed makes it practically impossible to do so.

TRULY RANDOM

In summary, a RANDOM NUMBER CAN BE FAKED if it is generated using a pseudo-random number generator that uses a predictable algorithm or formula. To generate truly random numbers, a source of randomness is required.

In QUANTUM RANDOM NUMBER GENERATION, a quantum measurement is made on a quantum system, and the result is used to generate a random number. Since quantum measurements are inherently probabilistic, the resulting number is TRULY RANDOM and cannot be predicted even if the complete measurement apparatus and the environment are known.

This method of generating random numbers is considered to be unbreakable, as it relies on the fundamental unpredictability of quantum mechanics. It is also considered to be more secure than other methods of generating random numbers, as it is NOT SUSCEPTIBLE TO ATTACKS BASED ON KNOWLEDGE OF THE ALGORITHM OR SEED VALUE used in pseudo-random number generators. With quantum techniques, it is possible to generate truly random numbers that cannot be faked or predicted.



Due to the fact that quantum processes follow probabilistic laws and that it is therefore impossible to accurately predict the outcome of a single process, many people have looked to quantum mechanics as a source of unbreakable random numbers. But this has a fundamental flaw: there is no way to do it.
If you weren't there to observe the procedure, you would have no way of knowing whether a number is real or fake.

The University of Texas at Austin's Scott Aaronson and Shih-Han Hung have now created a protocol that can certify truly random numbers without this requirement. They do this by asking a quantum computer to complete a test in a specific amount of time that only it can pass. A quantum computer can only pass the test by performing the quantum computation it is designed to perform, or one that is reasonably close to it.
According to Carlos Perez Delgado (at the University of Kent in the UK), it is a remarkable piece of work and the only practical way to confirm true randomness without having to physically observe the process.
Given the significant computational resources needed to verify the quantum result, it may not have many applications, he says, as many pseudo random number generators are reliable enough for cybersecurity applications and difficult to crack.

Given the sum of money at risk, one of these could be for the ETHERUM cryptocurrency network. It uses random numbers to verify its transactions and create new money, and it is worth several hundred billion dollars (Milliard in the case of Europe).

The new strategy may not be required in many situations, though. Some detractors of this protocol would merely assert that it is excessive. You don't need to prove that a sequence of numbers is random in the deep sense that the universe considers them random, for pretty much any application.

BLAISE PASCAL CAFE

in IJsselstein NETHERLANDS Saturday 15th of April 2023

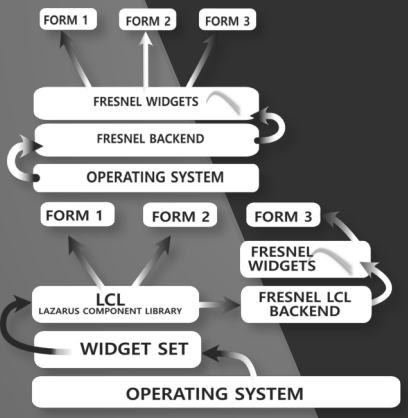
ADDRESS: Touwlaan 36,
3401 CB IJsselstein
NETHERLANDS
Mob +31
06 21.23.62.68

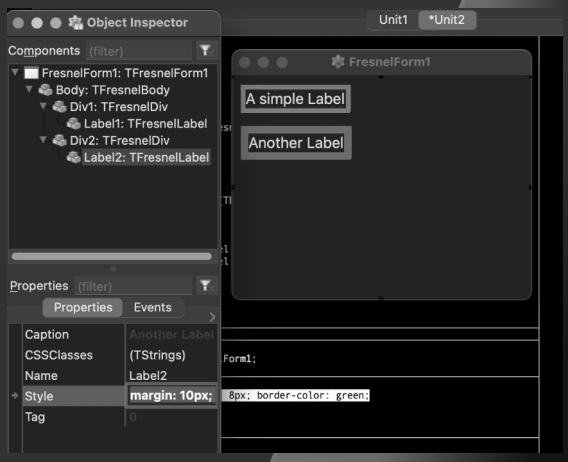
PROGRAM SPRING EVENT

Start: 10.00 Coffee: opening the Pascal Café

- Detlef Overbeek: 2
  Introducing of the new book for Learning how to program with PAS2JS. Emphasis on creating your own Server and a Database on that same Server.

- Mattias Gärtner Showing the latest results of our ROAD MAP for FPC-Lazarus





- Rainer Stratmann:
  He will talk in a short presentation about his software on a separate mini itx computer that runs all the time. It controls the greenhouse climate. There is a build in webserver and the computer communicates with temperature sensors and relais modules and a weather station. He will show what can be achieved with the help of FreePascal and Lazarus

https://www.blaisepascalmagazine.eu/product/14233/

BLAISE PASCAL CAFE

in IJsselstein NETHERLANDS Saturday 15th of April 2023

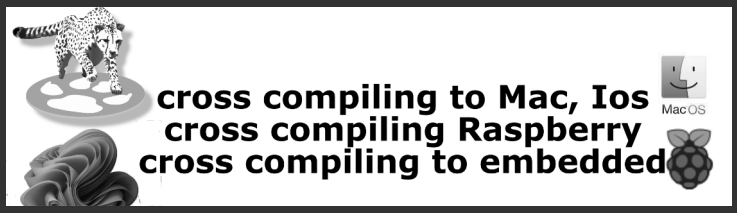
ADDRESS: Touwlaan 36,
3401 CB IJsselstein
NETHERLANDS
Mob +31
06 21.23.62.68

PROGRAM SPRING EVENT

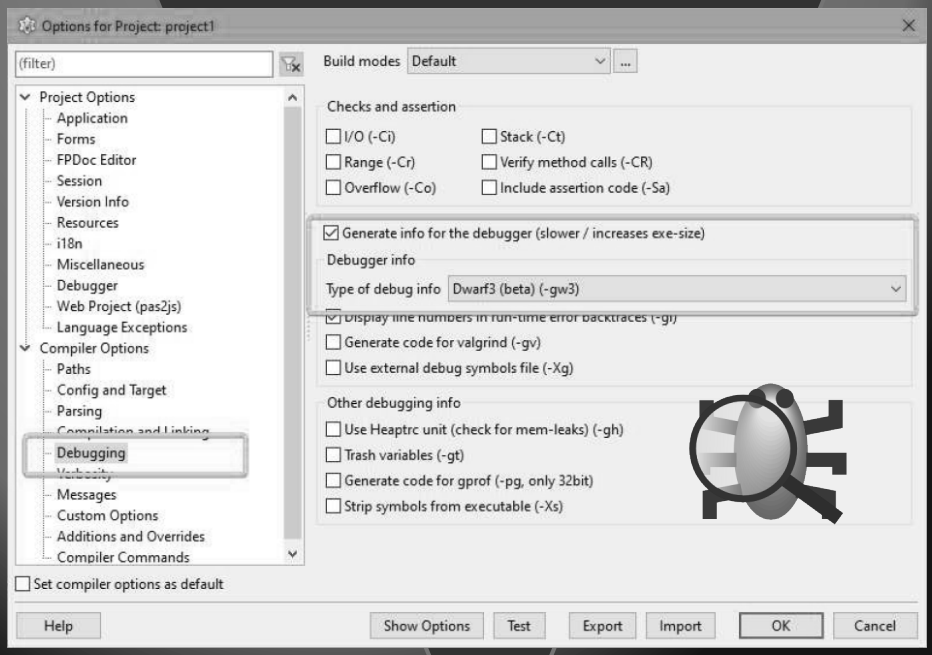
Lunch: 12.00 – 13.00

Afternoon: 13.00

- Alfred Glänzer:
  Cross compiling with FreePascal and Lazarus in Realtime:
  Live Time executed on various possible mini chips like Raspberry and others.



- Martin Friebe: Debugging in Lazarus:
  the latest results of the completely rebuild debugger in FPC/Lazarus



Finish: 17.00

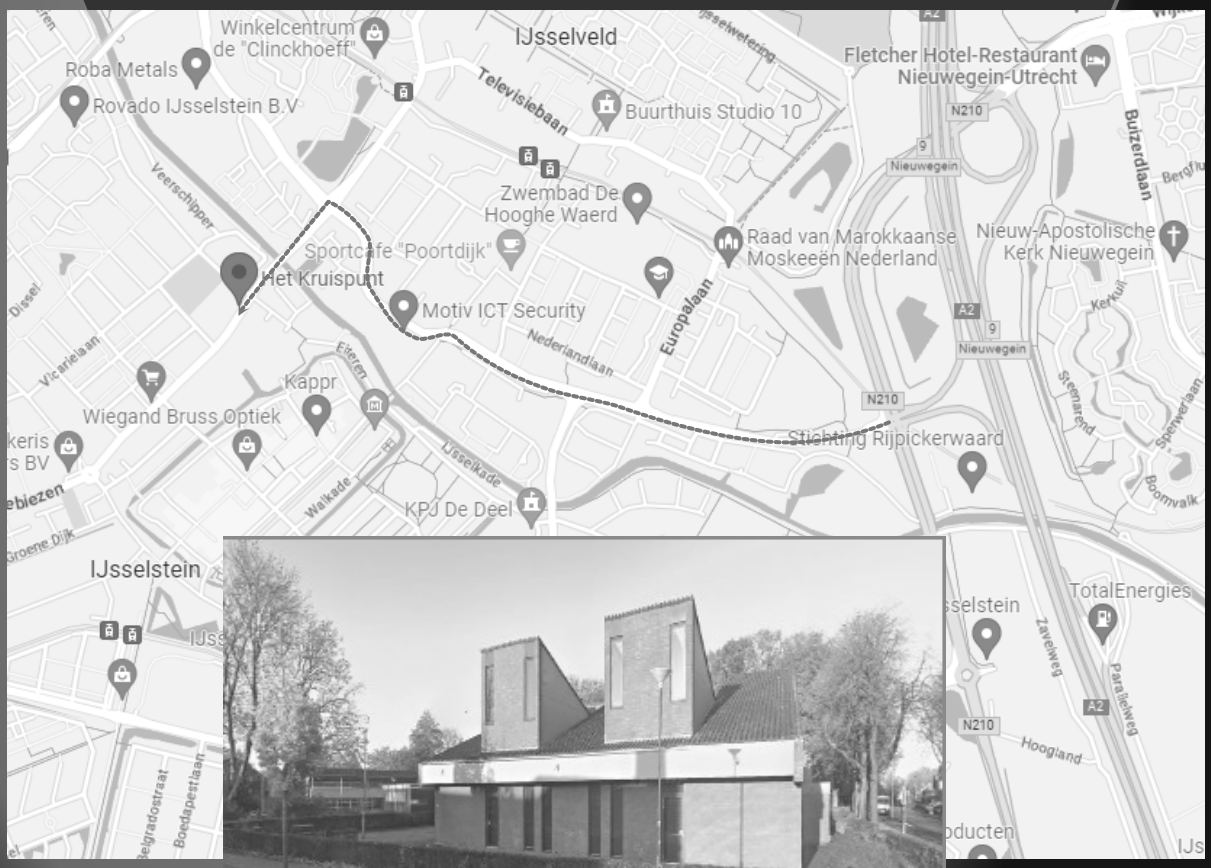
https://www.blaisepascalmagazine.eu/product/14233/

BLAISE PASCAL CAFE

in IJsselstein NETHERLANDS Saturday 15th of April 2023

ADDRESS: Touwlaan 36,
3401 CB IJsselstein
NETHERLANDS
Mob +31
06 21.23.62.68

PROGRAM SPRING EVENT



https://www.blaisepascalmagazine.eu/product/14233/

Image to Text API

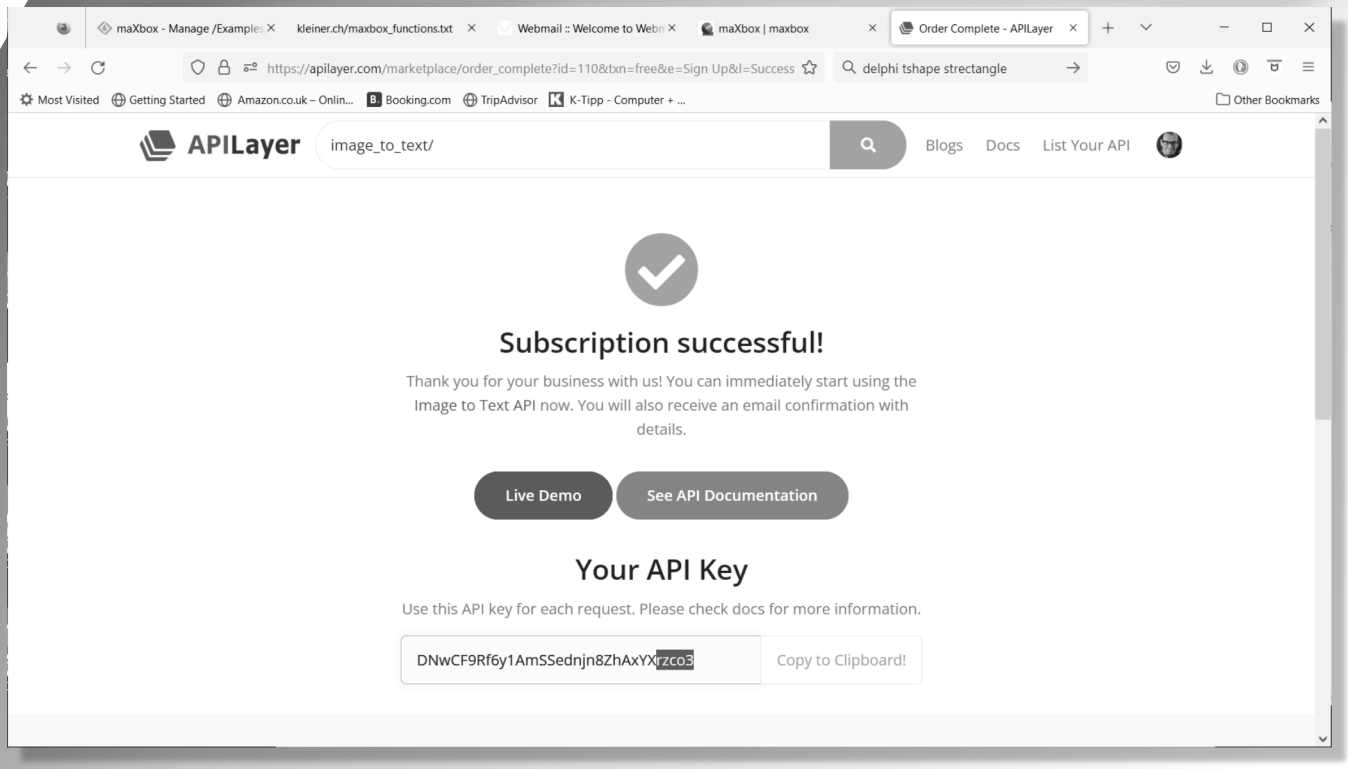
maXbox Starter 103 – Text recognition of characters in images.

Starter Expert

"A picture is worth a thousand words. An interface is worth a thousand pictures.".

This API recognizes and reads a text embedded in pictures or photos.
Image to Text API uses a neural net (LSTM) based OCR engine which is focused on line recognition, but also supports recognizing the character patterns.
It supports both handwriting and printed materials as well as street maps.
APILayer is an API marketplace where also your API can reach a broader audiences, but first you need an API-key for free:

The result of a simple subscription will be the screenshot below:



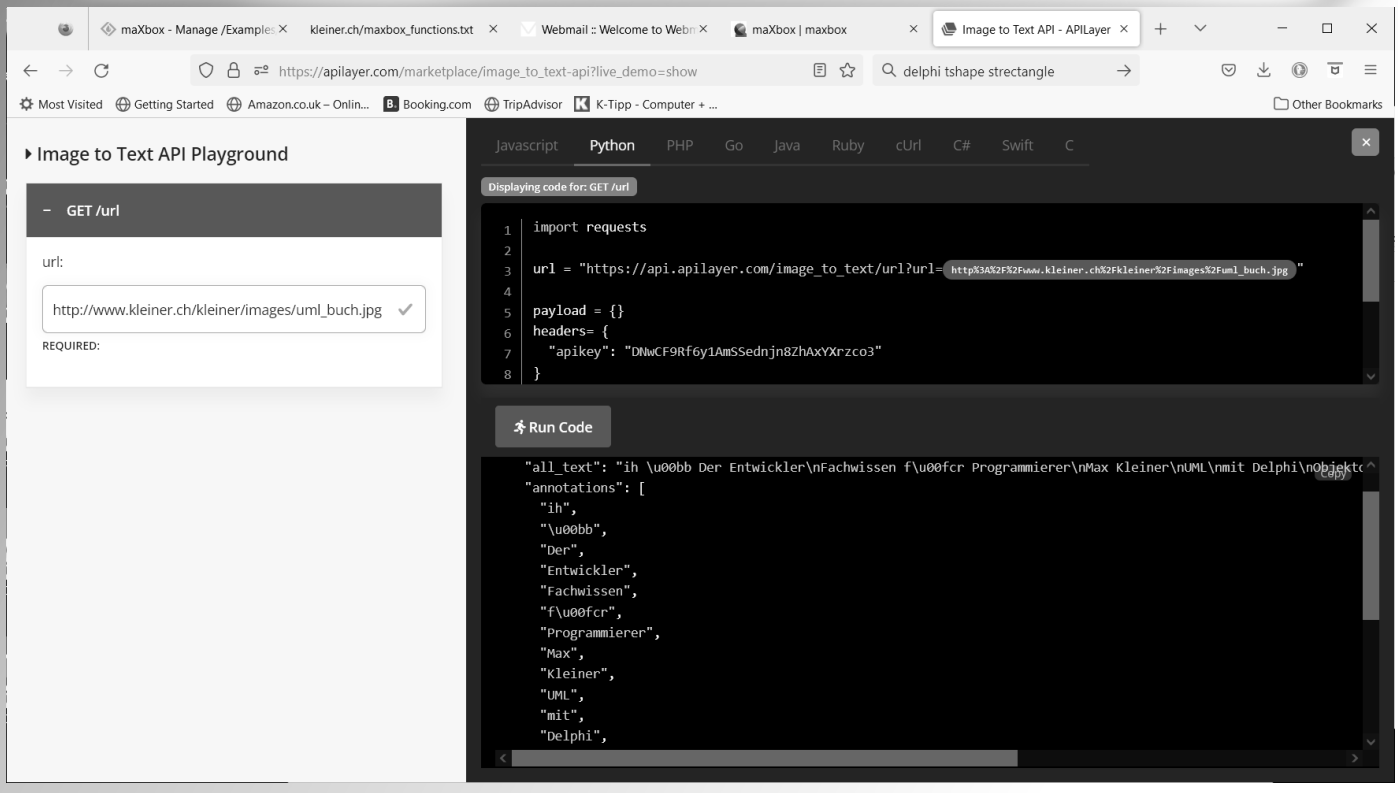
This register allows you a monthly usage of 30 successful calls.
Almost all API's has a free plan to subscribe. Looking at the following book-cover, it will extract the text information easily, even though the cover has shadows and positioned with angle.
We use WinHttp.WinHttpRequest, JSONObjects and `TGraphics` library with loading and testing the REST-client. Also we pass the API-key as a request-header, so get a key first at:
`https://apilayer.com/marketplace`
You can also use a powerful 'OCR' feature (text in picture recognition) to extract text from an image during the conversion process. In this case, you will get an editable text document as a result that you can adjust and
modify as you need.
The data represents is JSON data with all the text extracted and even the language of the text to scan is auto detected. Before we dive into code this is the main part of the script:

Image to Text API

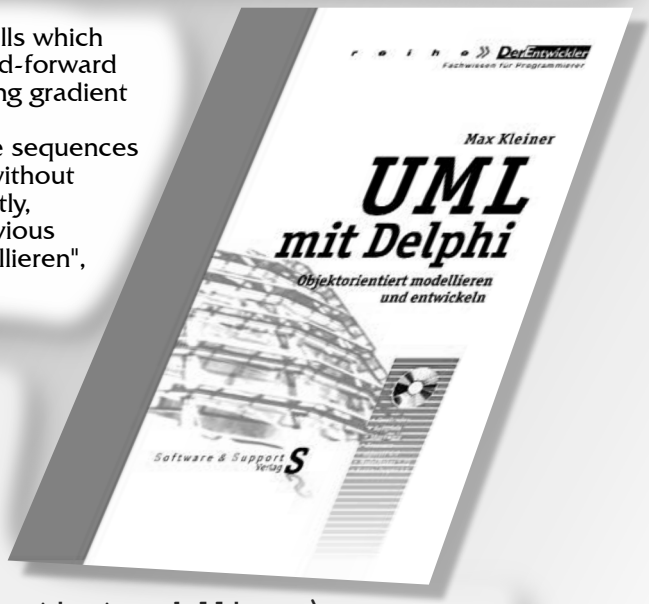
maXbox Starter 103 – Text recognition of characters in images.

```
function Image_to_text_API2(AURL, url_imgpath, aApikey: string): string;
var httpq: THttpConnectionWinInet;
   rets: TStringStream;
   heads: TStrings; iht:IHttpConnection2;
begin
  httpq:= THttpConnectionWinInet.Create(true);
  rets:= TStringStream.create('');
  heads:= TStringlist.create;
   try
    heads.add('apikey='+aAPIkey);
    iht:= httpq.setHeaders(heads);
    httpq.Get(Format(AURL,[url_imgpath]),rets);
    if httpq.getresponsecode=200 Then result:= rets.datastring
     else result:='Failed:'+
         itoa(Httpq.getresponsecode)+Httpq.GetResponseHeader('message');
   except
    writeln('EWI_HTTP: '+ExceptiontoString(exceptiontype,exceptionparam));
   finally
    httpq:= Nil;
    heads.Free;
    rets.Free;
   end;
end;
```

The main part function opens a connection, invokes the API and results a stream which we convert to a datastring.
Image2Text or Image to Text live demo is providing an API service on its APILayer publication platform. Live Demo feature allows you to test the API within your browser; no need to install or code anything. You can modify all the parameters as you like and interact with the API from many languages. The API export format is JSON, e.g. our book cover see below:



The published result datasets are based on LSTM in combination with a OCR. LSTM stands for Long Short-Term Memory and is a type of Recurrent Neural Network(RNN). Talking about RNN, it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory as memory cells for a short period of time (short-term memory). For example take our book-cover as input:



LSTMs have feedback connections and cells which make them different to more traditional feed-forward neural networks with the still existing vanishing gradient problem.
This property enables LSTMs to process entire sequences of data (*e.g. time series, handwriting or sentences*) without treating each point in the sequence independently, but rather, retaining useful information about previous data in the sequence like "Objektorientiert", "modellieren", "und", "entwickeln" as a context.
The output of the call

```
writeln(Image_to_text_API2(URL_APILAY,
URLIMAGEPATH4,
DNwCF9Rf6y1AmSSednjn8ZhAxYXr----'));

is the JSON datastring in about Runtime:
0:0:3.859:
{"lang": "de", "all_text":
"ih \u00bb Der Entwickler\nFachwissen
f\u00fcr Programmierer\
nMax Kleiner\nUML\nmit Delphi\nObjektorientiert modellieren\
nund entwickeln\nSoftware & Support", "annotations": ["ih", "\u00bb",
"Der", "Entwickler", "Fachwissen", "f\u00fcr", "Programmierer", "Max",
"Kleiner", "UML", "mit", "Delphi", "Objektorientiert", "modellieren",
"und", "entwickeln", "Software", "&", "Support"]}
```

Also the well known Tesseract 4.0 (like OmniPage) added a new OCR engine based on LSTM neural networks.

The API can also be triggered with this few lines of P4D code:

```
procedure PyCode(imgpath: string);
begin
 with TPythonEngine.Create(Nil) do begin
 pythonhome:= 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
 try
  loadDLL;
  ExecString('import requests');
  ExecStr('url= "https://api.apilayer.com/image_to_text/url?
                                    url='+imgpath+'"');
  ExecStr('payload = {}');
  ExecStr('headers= {"apikey": "dy5L70eQx72794XBZ8sewEgYTZR85----"}');
  Println(EvalStr('requests.request("GET",url, headers=headers,
                                  data=payload).text'));
 except
  raiseError;
 finally
  unloadDLL;
  free;
 end;
end;
end;
```

When you fail with a restricted call or an invalid key you get a bunch of exceptions like the following:

```
winininet_error: Unauthorized (401). or
{"message":"Invalid authentication credentials"}
```

The fact that error code is not "one of the expected return values" tells for the versions that the error comes from an underlying layer and this API just passes it up on internal failure.To shine a bit more light on those errors a function exists to convert the **ErrorCode** to a string for better understanding:

```
function GetWinInetError(ErrorCode:Cardinal): string;
const
   winetdll = 'wininet.dll';
var
  Len  : Integer;
  Buffer: PChar;
begin
  Len := FormatMessage(
  FORMAT_MESSAGE_FROM_HMODULE or FORMAT_MESSAGE_FROM_SYSTEM or
  FORMAT_MESSAGE_ALLOCATE_BUFFER or FORMAT_MESSAGE_IGNORE_INSERTS or
  FORMAT_MESSAGE_ARGUMENT_ARRAY,
  Pointer(GetModuleHandle(winetdll)), ErrorCode,0, @Buffer,SizeOf(Buffer),
nil);
  try
    while (Len > 0) and {$IFDEF UNICODE}(CharInSet(Buffer[Len - 1],[#0.
.#32,
        '.'])) {$ELSE}(Buffer[Len- 1] in [#0..#32, '.']) {$ENDIF} do
Dec(Len);
    SetString(Result, Buffer, Len);
  finally
    LocalFree(HLOCAL(Buffer));
  end;
end;
```

Our final example is an interesting one. What about an old wine bottle (1993) with shapes (*and grapes* ☺)), an old historic painting from 1619, dates, symbols and position angles:
https://my6code.files.wordpress.com/2022/12/wine_1993_20221230_141947.jpg?w=768

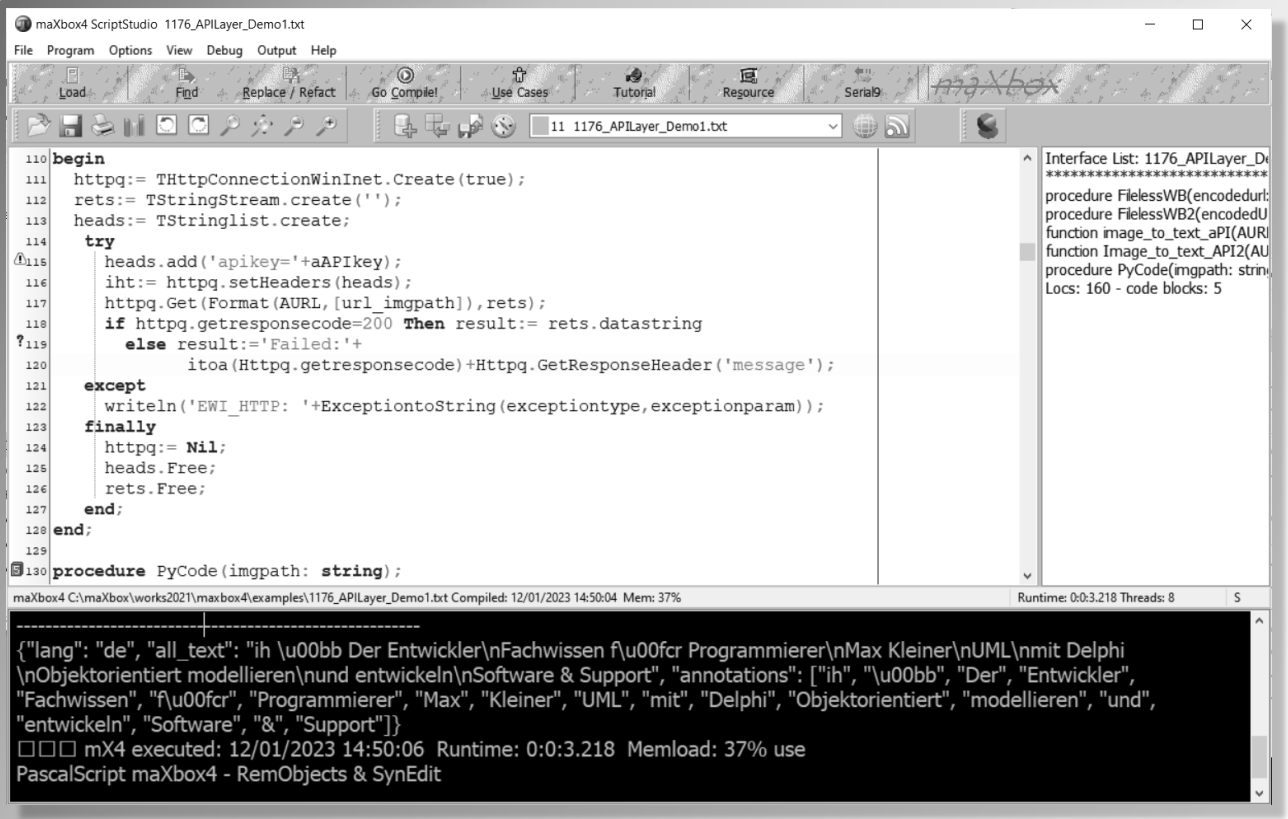
And the result is convincing, also the fact that the year in the label image was recognized correctly as 1619:

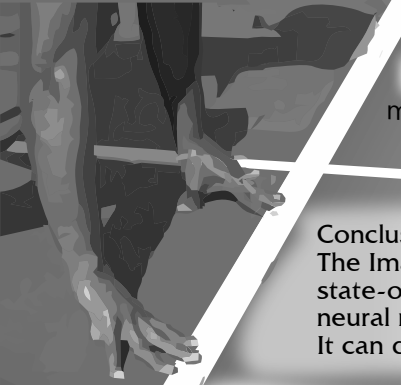
{"lang": "it", "all_text":
"DS₩nPODERE CAPACCIA₩n1619₩nLuggo deffo apacia₩
nQUERCIAGRANDE₩nVino da tavola della Toscana₩n1993₩
nProdotto e imbottigliato all'origine₩nPodere Capaccia₩
nPacini Giampaolo & C. s.a.s.₩nRadda in Chianti (SI) - Italia₩nItalia",
"annotations": ["DS", "PODERE", "CAPACCIA", "1619", "Luggo", "deffo",
"apacia", "QUERCIAGRANDE", "Vino", "da", "tavola", "della", "Toscana",
"1993", "Prodotto", "e", "imbottigliato", "all'origine", "Podere", "Capaccia",
"Pacini", "Giampaolo", "&", "C.", "s.a.s.", "Radda", "in", "Chianti", "(", "SI", ")", "-", "Italia", "Italia"]}

Any missing or incomplete data is difficult to find without knowing the original. But on the other side, it is very easy to use since users just need to screenshot the part they wish to convert and then copy the text after.

Furthermore the API access is provided in a REST-like interface (*Representational State Transfer*) exposing database resources or pre-trained models in a JSON format with content-type in the Response Header.

NOTE: If a programming language is not listed in the Code Example from the live demo, you can still make API calls by using a HTTP request library written in our programming language, as we did with GET or POST.





Conclusion:
The Image to Text API from APILayer detects and extracts text from images using state-of-the-art optical character recognition (OCR) algorithms in combination with a neural network called LSTM.
It can detect texts of different sizes, fonts, and even handwriting or difficult numbers.

Reference:
https://apilayer.com/marketplace/image_to_text-api
https://apilayer.com/docs

https://my6.code.blog/2022/09/02/webpostdata/
http://www.kleiner.ch/kleiner/images/uml_buch.jpg

Doc and Tool: https://maxbox4.wordpress.com

Script Ref: 1176_APILayer_Demo1.txt

Appendix: A Delphi REST client API to consume REST services written in any programming language with a class from maXbox4 integration:
https://github.com/fabriciocolombo/delphi-rest-client-api/blob/master/src/HttpConnectionWinInet.pas

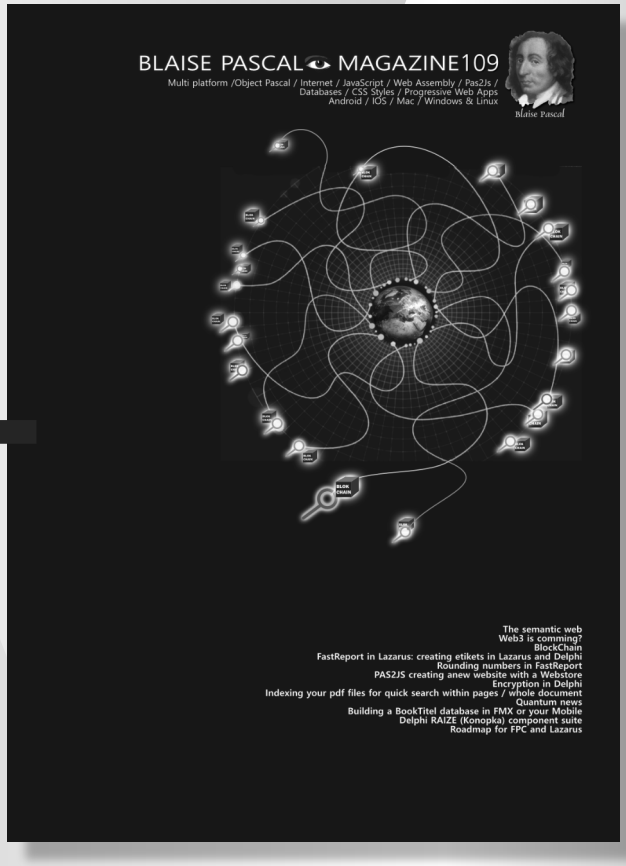
The API it is designed to work with Delphi 7 or later; newer versions takes advantage of Generics Methods.
https://github.com/fabriciocolombo/delphi-rest-client-api

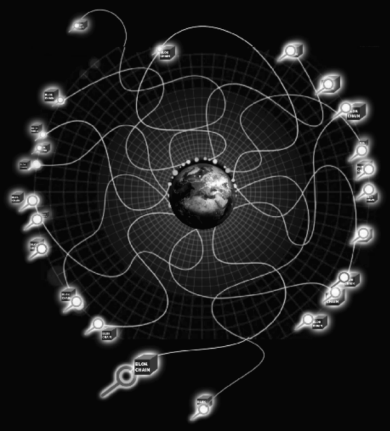
LAZARUS HANDBOOK (PDF) +SUBSCRIPTION 1 YEAR

- **Lazarus Handbook**
- Printed in black and white
- PDF Index for keywords
- Almost 1000 Pages
- Including 40 Examples
- **Blaise Pascal Magazine**
- English and German
- Free Lazarus PDF Kit Indexer
- 8 Issues per year
- minimal 60 pages
- Including example projects and code

SPECIAL OFFER € 75

+





By Detlef Overbeek

WEB3

Web 3.0 aims to transform the internet into a more regimented version?
There is a lot of excited interest around the idea of a decentralised version of the internet that would give ordinary users more power.
And on top of that there is a possibility of a **Semantic Web**.

There are voices calling for a new version of the internet for obvious reasons:

1. to provide a clearly better digital future for the internet
2. the technology giants that currently dominate our online world (Internet) are rocked by huge abuses, they are losing a lot of money - by damaging their markets - and appear unable or unwilling to solve the problems of online disinformation, harassment and hate speech.

SHOULD IT BE DECENTRALISED?

That means that instead of just a few big providers for monitoring all (Providers etc.) registered data, there are a large number who do this completely independently.
Basically, anyone should be able to retrieve the data. Think of it as a certificate of guarantee that is re-recorded and public with every action.

Go gle

WHAT IS WEB 3.0?

How does it work?Can it really offer a markedly better digital future for the internet?
The idea of a concatenated series of computers - the internet - is now 30 years old.
The World Wide Web, the shared set of websites we can view online through those connected computers, was created in 1989 by Tim Berners-Lee.

In its early days, he already had grand plans for the future of the Web "The Semantic Web", which is explained on page 11 of this article.

In the first generation of the Web, known as Web 1.0, users (actually participants) hosted their own websites and e-mail on servers, either at home or at an institution. It was an idyll for hobbyists, full of promise and revolutionary zeal.
In the end, that could not last.
Participants had to manage their own affairs, keep their computers constantly running to host their own web pages and email in-boxes. It was too burdensome for all but the most dedicated enthusiasts with the result that by the late 1990s, companies and large organisations in.
So there was GeoCities, (**https://wiki.archiveteam.org/index.php/GeoCities**) which hosted your web pages so you didn't have to.
Hotmail and Yahoo did something similar for e-mail.

The term **Web 2.0** was coined in 1999 when people started moving to centralised providers. This era saw the launch of several companies that would have an ever-increasing grip on our lives: YouTube (Google-Alphabet), Facebook - (now Meta), Twitter (Elon Musk), WhatsApp (now Meta) and Instagram (now Meta) and more: Tiktok (China).

The rise of these companies had a host of advantages:
One is making it significantly easier to access the online world.
We can log into very many websites with a Google account and pay online with PayPal and nowadays even more services.
It all seemed to become so easy.

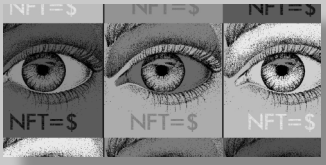
It also meant that it became easier to rigorously deceive the participant or user of this new version and extort money from him.
A succession of more recent scandals shows that Web 2.0 has very serious society-disrupting drawbacks. One of the most glaring examples came in 2018, when it emerged that the company Cambridge Analytica had collected the personal data of millions of Facebook users, profiled them and then sent them politically polarising ads on their feeds.
The company worked for Donald Trump's campaign for the US presidency in 2016.

web3
foundation

In 2021, whistle-blower Frances Haugen, an ex-employee of Facebook, released a series of documents that she said showed that the company prioritised its growth and offered no guarantees to protect young people from potentially harmful content.

In autumn 2022, Elon Musk completed his takeover of social network Twitter. Within days, he had removed staff and launched the idea of charging users for verification. He has also said in the past that he is an "absolutist of free speech". For many users who had invested time in building their profiles on Twitter, it felt like time to leave.
In short, the rose-coloured glasses have been taken off. This summing up shows that the tech giants are too powerful and untrustworthy. Power corrupts.

This malaise is one of the reasons why there is talk of Web3.
You may have caught on to what an NFT (Non-Fungible Token) means.



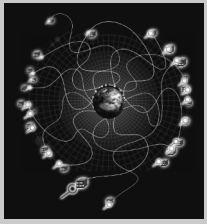
https://simon.rochester.edu/blog/deans-corner/nonfungible-tokens
The university of Rochester will explain a lot: Inside the Use of Non-Fungible Tokens (NFT) to Sell Collectables /
By Professor Derek Mohr

(A non-fungible token (NFT) is a unique digital identifier that cannot be copied, substituted, or subdivided, that is recorded in a blockchain, and that is used to certify ownership and authenticity. The ownership of an NFT is recorded in the blockchain and can be transferred by the owner, allowing NFTs to be sold and traded. NFTs can be created by anybody, and require few or no coding skills to create.[2] NFTs typically contain references to digital files such as photos, videos, and audio. Because NFTs are uniquely identifiable assets, they differ from crypto currencies, which are fungible.

Proponents of NFTs claim that NFTs provide a public certificate of authenticity or proof of ownership, but the legal rights conveyed by an NFT can be uncertain. The ownership of an NFT as defined by the blockchain has no inherent legal meaning and does not necessarily grant copyright, intellectual property rights, or other legal rights over its associated digital file. An NFT does not restrict the sharing or copying of its associated digital file and does not prevent the creation of NFTs that reference identical files).

WIKIPEDIA

Illustration of a non-fungible token generated by a smart contract (a program designed to automatically execute contract terms)



web3
foundation

It is cryptographic assets (blockchain) that can be associated with ownership of digital art and is often talked about in the same breath as Web3. There is a flood of news articles questioning whether Web3 is the next big thing, venture capitalists extol its virtues and the management consulting firm McKinsey released a report on Web 3 in September 2022 stating that Web3 could have "potentially transformative effects".

BLOCKCHAIN

Satoshi Nakamoto (born 5 April 1975) is the name used by the pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation. As part of the implementation, Nakamoto also devised the first blockchain database. Nakamoto was active in the development of bitcoin up until December 2010.

There has been widespread speculation about Satoshi Nakamoto's true identity, with a variety of people posited as the person or persons behind the name.
Though Nakamoto's name is Japanese, and he stated in 2012 that he was a man living in Japan, most of the speculation has involved software and cryptography experts in the United States or Europe.

Satoshi Nakamoto's message embedded in the coinbase of the first block

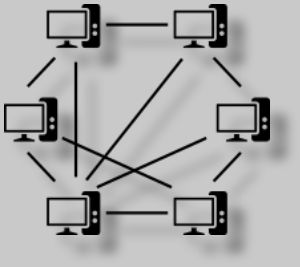
Nakamoto would form the basis for the first cryptocurrency, **Bitcoin**, where each block was a kind of unit of money.
In 2015 came **Ethereum**, another blockchain platform that has been used in many ways, including as the basis for another cryptocurrency (a lot of these companies have been in big trouble,but since I am only interested in the technical aspects, I will not dive into that. I think I have to warn you Cryptocurrency is very dangerous as the downfalls of the last episode has shown), **Ether** a cryptocurrency, and as a way to implement smart contracts, where the terms of an online deal or purchase are written into a blockchain and self-executed when certain conditions are met.
Most importantly, Ethereum offered for the first time a way to build apps hosted not by an individual company, but on a decentralised blockchain.

web3
foundation

A blockchain is a distributed ledger with growing lists of records (blocks) that are securely linked together via cryptographic hashes. Each block contains a cryptographic hash of the previous block, a time stamp, and transaction data (generally represented as a **Merkle tree**, where data nodes are represented by leaves).

In cryptography and computer science, a hash tree or Merkle tree is a tree in which every "leaf" (node) is labelled with the **cryptographic hash** of a data block, and every node that is not a leaf (called a branch, inner node, or inode) is labelled with the cryptographic hash of the labels of its child nodes. A hash tree allows efficient and secure verification of the contents of the contents of a large data structure.
A hash tree is a generalisation of a hash list and a hash chain.

The time stamp proves that the transaction data existed when the block was created. Since each block contains information about the previous block, they effectively form a chain (*compare linked list data structure*), with each additional block linking to the ones before it.
Consequently, blockchain transactions are irreversible in that, once they are recorded, the data in any given block **cannot be altered** retroactively without altering all subsequent blocks.

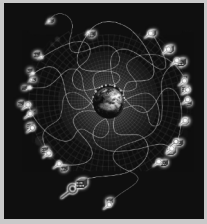


Blockchains are typically managed by a peer-to-peer (P2P) computer network for use as a public distributed ledger, where nodes collectively adhere to a consensus algorithm protocol to add and validate new transaction blocks. Although blockchain records are not unalterable, since blockchain forks are possible, blockchains may be considered secure by design and exemplify a distributed computing system with high **Byzantine fault tolerance.**

A blockchain was created by a person (or group of people) using the name (or pseudonym) Satoshi Nakamoto in 2008 to serve as the **public distributed ledger for bit coin crypto currency transactions**, based on previous work by **Stuart Haber, W. Scott Stornetta, and Dave Bayer**.
The implementation of the blockchain within bitcoin made it the first digital currency to solve the double-spending problem without the need of a trusted authority or central server.

The bitcoin design has inspired other applications and blockchains that are readable by the public and are widely used by crypto currencies.
The blockchain may be considered a type of payment rail.
Private blockchains have been proposed for business use.

COMPUTERWORLD
(A digital magazine - `https://www.computerworld.com`) called the marketing of such privatised blockchains without a proper security model "snake oil"; however, others have argued that permissioned blockchains, if carefully designed, may be **more decentralised and therefore more secure in practice than permissionless ones**.



web3
foundation

THE PROMISE OF WEB3

Web3 relies heavily on these "distributed apps" (dapps) to avoid the problems of centralised control. For Gavin Wood, a co-founder of Ethereum who coined the term web3 in 2014, there are several potential benefits.

First, because dapps are decentralised, they are in a sense hosted by everyone rather than by one company - Wood has described this as "more democratic".
Second, we no longer have to rely on companies to provide online services - the guarantee can be built into the underlying blockchain.
Third, verification of who people (Persons) are, could be more robust on a decentralised web. In other words, fake accounts and bots are shot down immediately.

People could still remain anonymous, but their activities could be tracked over time, again thanks to the blockchain.
In an ideal future, Web3 could challenge the tech giants' monopolies and make it much easier to **root out automated fake accounts**, or bots, that spread hate speech.

In principle, it would also allow users to move their profiles and data more easily between online services, as their data would be stored on a shared blockchain rather than on a company's private servers.

IF YOU DON'T TRY IT, IT WON'T WORK, BUILD IT - AND THEY WILL COME.

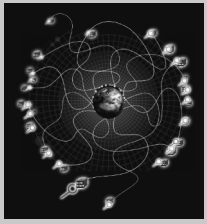
In 2017, Wood and others founded the **Web3 Foundation** to help fund and develop ideas that could enable this new online environment. According to **Alistair Stewart** - principal investigator at the foundation - the fundamental point is decentralisation, one doesn't want to rely on individual companies or even individual people, rather trust a broad group of people."

There is one major drawback of a decentralised solution:
the ease of use has to be addressed very clearly and deeply upfront because an end user always demands simplicity of operation, which also applies to **Web2**.

This technical complexity has led to a **second problem** that can feel like history repeating itself: because Web3 depends on users implementing complex digital technology, companies are popping up to do it for them.
Just as **Google** and other giants popped up because Web1 was a bit tricky.
Moreover, they will do everything in their power not to let this version succeed - that would be their end.

The success of Web3 also depends on a comparison:
how annoyed are people with Web2 and how does that outweigh the difficulty of using Web3?
Most people mainly want **ease of use**, but that can be solved.
While users may regret **Google** or **Meta** finding out more about them every time they log into a third-party website via their **Gmail** or **Facebook** account, they appreciate the convenience they will therefore have to adapt and go small.
There are also many open source options.

The market will indicate how successful this is.
In any case, it is highly desirable.
If you don't try it, it won't work, build it and they will come.
And there is another previously mentioned possibility that needs to be added:
the Semantic Web
(Semantics or meaning theory is a science concerned with the meaning of symbols, involving in particular the building blocks of natural languages that serve for communication either words and sentences).



web3
foundation

WEB3

Web 3.0 aims to transform the internet into a more regimented version? There is a lot of excited interest around the idea of a decentralised version of the internet that would give ordinary users more power. And on top of that there is a possibility of a Semantic Web.

There are voices calling for a new version of the internet for obvious reasons:

❶ to provide a clearly better digital future for the internet.
❷ the technology giants that currently dominate our online world (*Internet*) are rocked by huge abuses, they are losing a lot of money - by damaging their markets - and appear unable or unwilling to solve the problems of online disinformation, harassment and hate speech.

DECENTRALISED?

That means that instead of just a few big providers for monitoring all (*Providers etc.*) registered data, there are a large number who do this completely independently. Basically, anyone should be able to retrieve the data. Think of it as a certificate of guarantee that is re-recorded and public with every action.

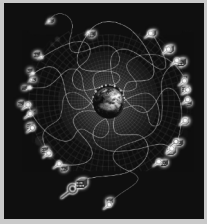
WEB 3 relies heavily on these "distributed apps" (**dapps**) to avoid the problems of centralised control.
For **Gavin Wood**, a co-founder of **Ethereum** who coined the term WEB 3 in 2014, there are several potential benefits.

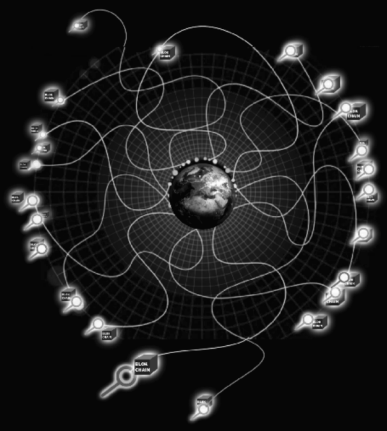
First,
because **dapps** are decentralised, they are in a sense hosted by everyone rather than by one company - Wood has described this as "more democratic".

Second,
we no longer have to rely on companies to provide online services - the guarantee can be built into the underlying blockchain.

Third,
verification of who people are (*Persons*), could be more robust on a decentralised web in other words, fake accounts and bots are shot down immediately.

https://wehsrobotics.weebly.com/





Semantic Web

In 2001, web founder Tim Berners-Lee, together with computer scientists **James Hendler** and **Ora Lassila**, first proposed that the web should be developed so that all content is machine-readable - in other words, can be "understood" by any computer.

Timothy John Berners-Lee, along with his then manager, **Belgian Robert Cailliau,** is the creator and founder of the technology and collection of agreements that made the world wide web or worldwide web (www) possible.
He worked on this when he was a consultant - software - engineer employed at **CERN** in Switzerland, from June to December 1990.
As director of the World Wide Web Consortium (W3C), he oversees the development of web languages and protocols such as **HTML, XML, CSS** and **HTTP.**

This means that the semantic web requires websites to provide metadata(→ *see explanation in right column*) that can then be analysed by software, apps and services.

The **Semantic Web**, sometimes known as **Web 3.0** (**not to be confused with Web3**), is an extension of the **World Wide Web** through standards set by the **World Wide Web Consortium** (**W3C**). The goal of the Semantic Web is to make Internet data machine-readable.

Every element on a web page would be labelled to aid computer comprehension.
For example, the sentence "Detlef Overbeek lives in IJsselstein" would be labelled with visible markers telling computers that "Detlef Overbeek" is a name and "IJsselstein" is a location.

The benefits of such labelling of all content on the web could be enormous. As AI becomes increasingly powerful, faster and more general, improving its ability to decipher our online knowledge base would take AI to an even higher level and, crucially, more effective in interacting with the human world.
In fact this has been used by Apple and Google on your mobile. if you are walking along a restaurant they can tell you the menu of that restaurant, prices, the chef and even show the interior. It might be a very good way to make sure a person is not a bot.

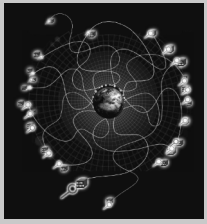
the past is document sharing, the future is data sharing

In his original vision of the semantic web, **Berners-Lee** envisioned a situation where a web browser could pull the treatment requirements for a patient from a doctor's website and then find a combined supply of, say, pharmacies that provides the corresponding medications and possibly services that could offer, say, physiotherapy.
Any appointments could be booked seamlessly by comparing the availability at a physiotherapist with the patient's schedule.
Combined with Web3, logging into Patient - Doctor data could be a simple but well-secured step.

Metadata *is "data that provides information about other data", but not the content of the data, such as the text of a message or the image itself. There are many distinct types of metadata, including:*

- *Descriptive metadata- the descriptive information about a resource. It is used for discovery and identification. It includes elements such as title, abstract, author, and keywords.*
- *Structural metadata- metadata about containers of data and indicates how compound objects are put together, for example, how pages are ordered to form chapters. It describes the types, versions, relationships, and other characteristics of digital materials.*
- *Administrative metadata - the information to help manage a resource, like resource type, permissions, and when and how it was created.*
- *Reference metadata- the information about the contents and quality of statistical data.*
- *Statistical metadata- also called process data, may describe processes that collect, process, or produce statistical data.*
- *Legal metadata- provides information about the creator, copyright holder, and public licensing, if provided.*

Metadata is not strictly bound to one of these categories, as it can describe a piece of data in many other ways.



Labelling internet data in this way is a labour-intensive task:
only a tiny fraction of the world's web pages now contain semantic information.
But this could be automated. Adding it requires searching a website's content and underlying computer code, and then classifying the information according to relevance.
Those engaged in this task do so because they see the promise of the semantic web.
An International Semantic Web Conference is held every year;
the most recent one, organised virtually by China, concluded at the end of October. The fact that it is precisely China that is doing this will certainly not appeal to us. Why not Europe? Why not Pascal? We are constantly extending the language and I think it might be a good idea.
There is work to be done... But therefore also hope for the end of web abuse.

To enable the encoding of semantics with the data, technologies such as Resource Description Framework (RDF) and Web Ontology Language (OWL) are used. These technologies are used to formally represent metadata. For example, ontology can describe concepts, relationships between entities, and categories of things. These embedded semantics offer significant advantages such as reasoning over data and operating with heterogeneous data sources.

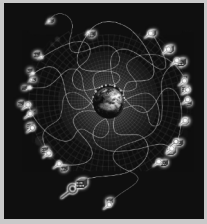
These standards promote common data formats and exchange protocols on the Web, fundamentally the RDF. According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries." The Semantic Web is therefore regarded as an integrator across different content and information applications and systems.

*The term was coined by Tim Berners-Lee for a web of data (or **data web**) that can be processed by machines - that is, one in which much of the meaning is machine-readable. While its critics have questioned its feasibility, proponents argue that applications in library and information science, industry, biology and human sciences research have already proven the validity of the original concept.[8]*

Berners-Lee originally expressed his vision of the Semantic Web in 1999 as follows:

> *I have a dream for the Web [in which computers] become capable of analysing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialise.*

The 2001 Scientific American article by Berners-Lee, Hendler, and Lassila described an expected evolution of the existing Web to a Semantic Web. In 2006, Berners-Lee and colleagues stated that: "This simple idea…remains largely unrealised".In 2013, more than four million Web domains (out of roughly 250 million total) contained Semantic Web markup.



SEMANTIC WEB SOLUTIONS

The Semantic Web involves publishing in languages specifically designed for data:

Resource Description Framework (RDF),
Web Ontology Language (OWL), and
Extensible Markup Language (XML).

HTML describes documents and the links between them.
RDF, OWL, and XML, by contrast, can describe arbitrary things such as people, meetings, or air plane parts.

These technologies are combined in order to provide descriptions that supplement or replace the content of Web documents.
Thus, content may manifest itself as descriptive data stored in Web-accessible databases, or as markup within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout or rendering cues stored separately).

The machine-readable descriptions enable content managers to add meaning to the content, i.e., to describe the structure of the knowledge we have about that content.
In this way, a machine can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more meaningful results and helping computers to perform automated information gathering and research.
An example of a tag that would be used in a non-semantic web page:

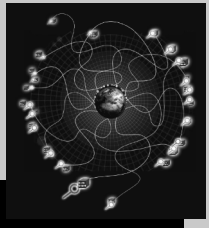
```
<item>blog</item>
```

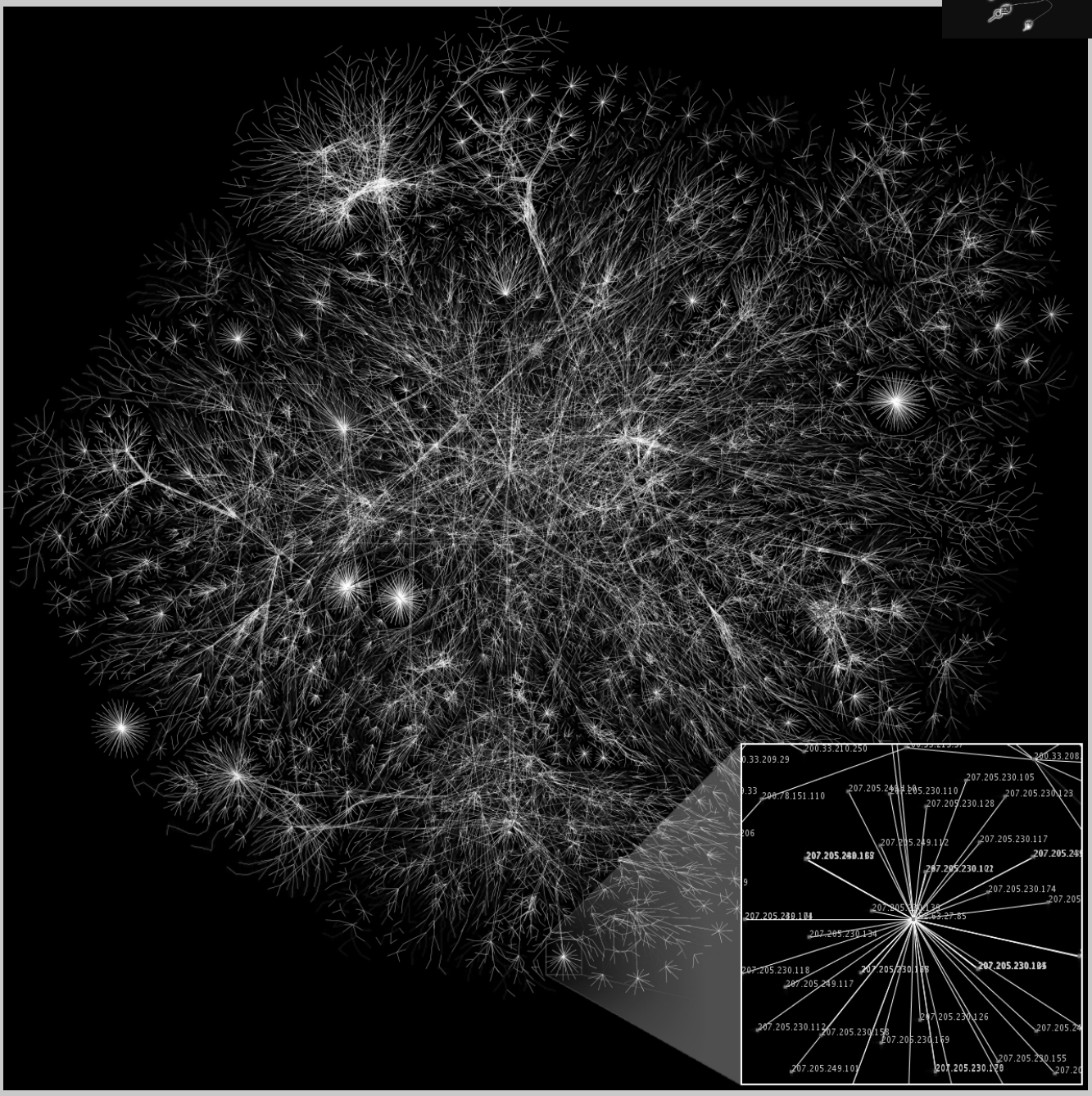
Encoding similar information in a semantic web page might look like this:

```
<item rdf:about="https://example.org/semantic-web/">Semantic Web</item>
```

Tim Berners-Lee calls the resulting network of Linked Data the **Giant Global Graph,** in contrast to the HTML-based **World Wide Web**.
Berners-Lee posits that if the past was **document sharing**, the future is **data sharing.**
His answer to the question of "how" provides three points of instruction.

❶ a URL should point to the data.
❷ anyone accessing the URL should get data back.
❸ relationships in the data should point to additional URLs with data.

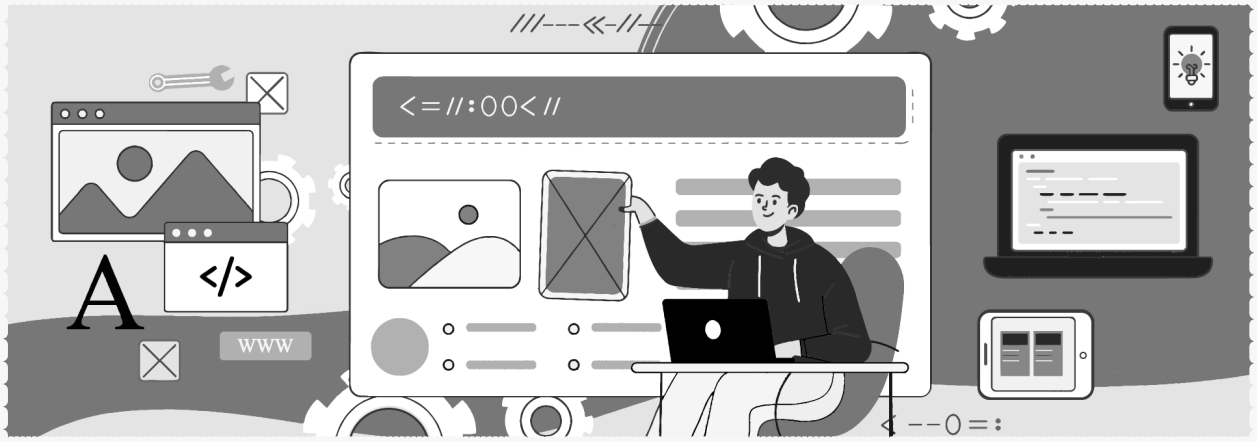




FASTREPORT IS DISCONTINUING SUPPORT FOR OLDER DELPHI VERSIONS April 5, 2023

Extra information you can find here:

HTTPS://WWW.FAST-REPORT.COM/EN/BLOG/SHOW/DISCONTINUING-SUPPORT-OLDER-DELPHI/



Even though we have been using Delphi since 1995, it is evolving and has undeniable advantages in the work of programmers. FastReport products for Delphi do not stand still as well, and we try to offer developers new and more modern features with each release. Important update: FastReport VCL no longer supports Delphi versions below 2010. We'll explain why.

You may be wondering: why update Delphi, and what are the advantages of newer versions? The main ones are many new features that boost productivity in high-end application development. In this article, we will try to talk about the main changes in Delphi since version 7 and answer questions about the end of support for Delphi's old versions.

1. What versions are no longer supported in the FastReport VCL product?
Borland Delphi 7
Borland Delphi 8
Borland Delphi 2005
Borland Delphi 2006
CodeGear Delphi 2007
Delphi 2009
Guaranteed support for FastReport VCL 2023.2 is available only starting with Delphi 2010.

FastReport VCL 2 — FastReport VCL 4.15 (2002-2014)	FastReport VCL 5 — FastReport VCL 2023.1 (2014-2023)		FastReport VCL 2023.2 — New versions (2023-...)	
Borland Delphi	Borland Delphi 7	Delphi XE7	Delphi 2010	Delphi 10.1 Berlin
Delphi 2	Borland Delphi 8	Delphi XE8	Delphi XE	Delphi 10.2 Tokyo
Delphi 3	Borland Delphi 2005	Delphi 10 Seattle	Delphi XE2	Delphi 10.3 Rio
Inprise Delphi 4	Borland Delphi 2006	Delphi 10.1 Berlin	Delphi XE3	Delphi 10.4.1 Sydney
Borland Delphi 5	CodeGear Delphi 2007	Delphi 10.2 Tokyo	Delphi XE4	Delphi 10.4.2 Sydney
Borland Delphi 6	Delphi 2009	Delphi 10.3 Rio	Delphi XE5	Delphi 11 Alexandria
Kylix	Delphi 2010	Delphi 10.4.1 Sydney	Delphi XE6	Delphi 11.1 Alexandria
	Delphi XE	Delphi 10.4.2 Sydney	Delphi XE7	Delphi 11.2 Alexandria
	Delphi XE2	Delphi 11 Alexandria	Delphi XE8	Delphi 11.3 Alexandria
	Delphi XE3	Delphi 11.1 Alexandria	Delphi 10 Seattle	
	Delphi XE4	Delphi 11.2 Alexandria		
	Delphi XE5	Delphi 11.3 Alexandria		
	Delphi XE6			

LibraryForm

Show article | Go to issue Nr | Issue | Search

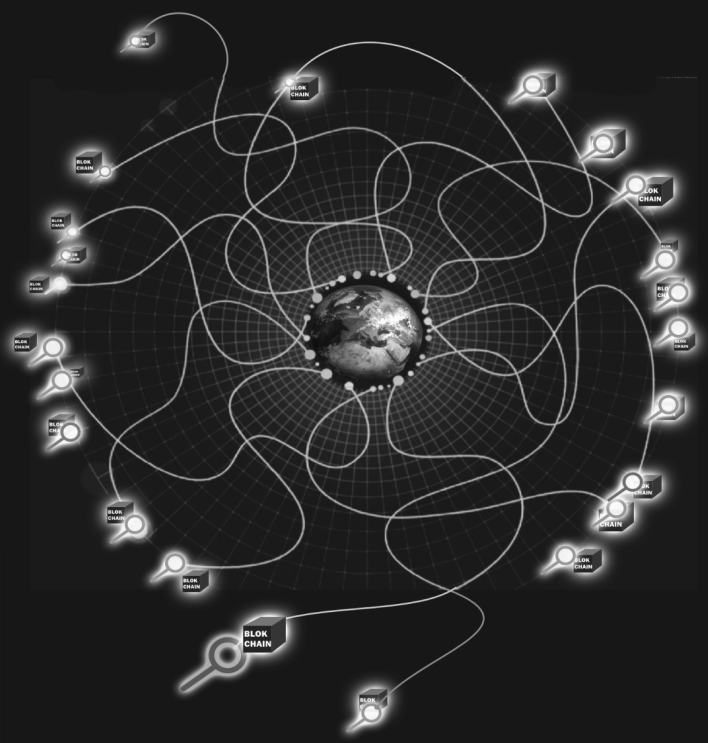
ID | IssueNr | Author | Article | PDF | PageNr

PDF VIEWER 2023
ON CREDIT CARD USB STICK INCLUDING
BLAISE PASCAL LIBRARY

Show Thumbnails | Page | Jump to page

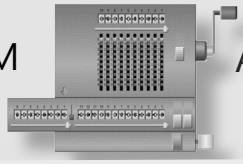
BLAISE PASCAL MAGAZINE109

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux



€ 100

More than 6000 pages



Starter Expert D11

INTRODUCTION

The Delphi programming language has operators div and mod for integer divide and remainder calculations.
For square root however only floating point calculations exist.
This project presents an integer square root algorithm which also supplies the remainder.

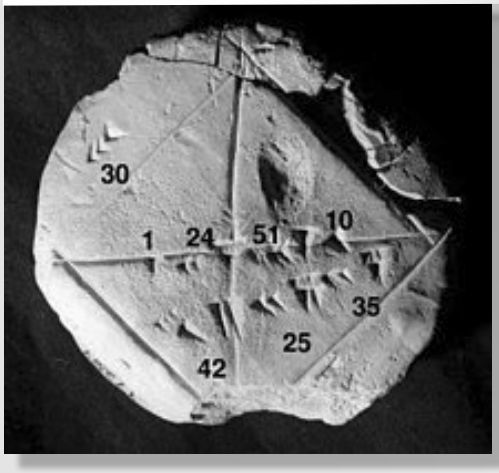


Figure 1: The Yale Babylonian Collection YB C 7289 clay tablet was created between 1800 BC and 1600 BC.

What is Square Root? The square root of a number is defined as the value, which gives the number when it is multiplied by itself. The radical symbol √ is used to indicate the square root. For example, √16 = 4.

div and mod

Division (mathematics), the mathematical operation that is the inverse of multiplication

Mod (modulus) divides two numbers and returns only the remainder. For instance, the expression "a:= 13 mod 4;" would evaluate to 1 (a=1), while "b := 12 mod 4;" would evaluate to 0 (b=0).

"The sign of the result of a Mod operator is the same as the sign of the left side operand of the Mod operator. In fact, the Mod operator is equivalent to the following operation :"

The reserved word mod belongs to the operators.

The reserved word mod is used to divide whole numbers.

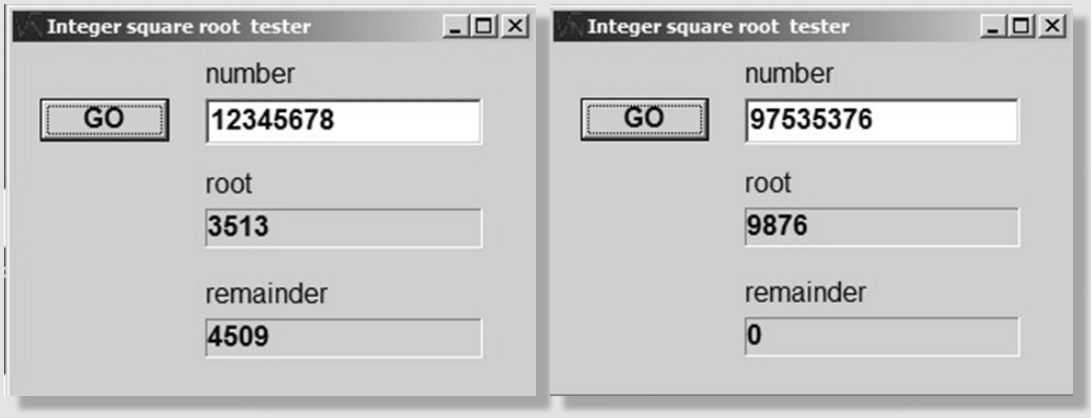
The reserved word mod stands for modulo, and returns the remainder of a division.

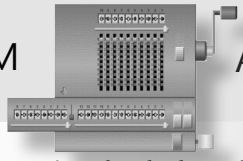
Example:

```
var
  intI: integer;
begin
  ...
  intI := 10 mod 3; // Result: 1
  ...
end;
```

So we write for the square root of number N:
N = root² + remainder.

Please look at the pictures below:





THE ALGORITHM

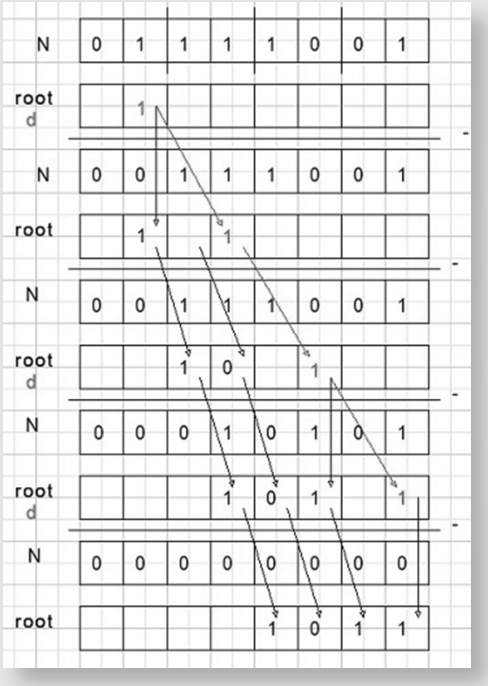
The integer square root project uses 32 bit positive integers (cardinal, dword) for N and 16 bit positive integers (word) as root and remainder.

For clarity an example is given where N is a byte.
We calculate the square root of 121 = $79 = $[0111\ 1001]_2$.

First, the binary number N is organised as groups of 2 bits, counted from the right side. Variables N, root and d are all bytes.

d is a single bit value (painted in red) and is initially positioned right in the leftmost group of 2 bits of N.
The variable root is zero initially (painted in black).
In the picture below, empty cells are zero.

The value root + d is subtracted from N only if N >= root+d.



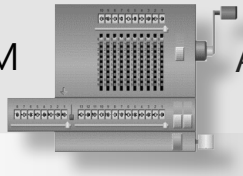
After a (trial)subtraction:
- if subtraction took place,
  the d bit is copied to the root
- the root is shifted one bit right
- the d bit is shifted 2 bits right

This process continues until the d bit is shifted out of the byte.
Then N holds the remainder.

HOW THIS WORKS

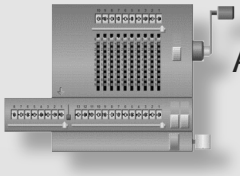
Consider a binary number with 2 bits a,b : $[ab]_2 = 2a+b$.
The square is $(2a+b)^2 = 4a^2 + 4ab + b^2$
So this square is a four bit number.

Now, $a^2 = a$, $b^2 = b$ for single bits.
for a = 1:
$4a^2=[100]_2$ and $4ab+b^2=[101]_2$ if b=1.
If N >= [100] then [100] is subtracted from N and a=1
Remains to find the value of b.
For a=1 , 4ab = [101] must be subtracted from N if b=1.
This applies to 4 bit numbers however:
These bits may be regarded as the 4 upper bits of a larger number.
After each (trial subtraction) step finding bit b, a is replaced by $[ab]_2$.
Then a^2 has already been subtracted from N so a next bit of b
has to be found by trial subtraction of $[a01]_2$.



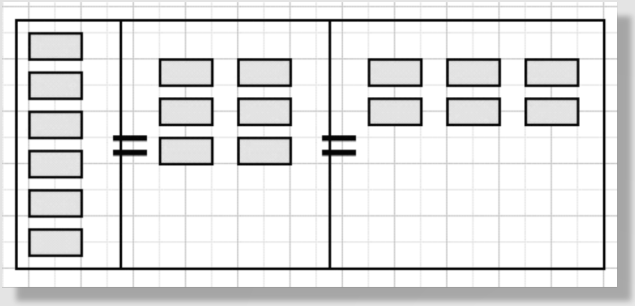
The program
N is the (32 bit) number from which the root is calculated.
At the end, N holds the remainder.
Variable `root` holds the value of a.
Variable `d` holds the value of b.
Initially, a is set to zero, d is set for the first subtraction to find a.
Then root holds the value of a.
Below is the procedure that calculates the integer root:

```
const chrset = ['0'..'9',#08,'$'];

function IntRoot(N : dword; var rem : word) : word;
  //return integer square root of N and remainder rem
var d,root,sub : dword;
begin
  d    := $40000000;
  root := 0;
  repeat
  sub  := root or d;

  if sub <= N then
   begin
     N     := N - sub;
     root  := (root shr 1) or d;
   end
  else root := root shr 1;

  d := d shr 2;
  until d = 0;
  result := root;
  rem    := N;
end;

procedure TForm1.GoBtnClick(Sender: TObject);
var N,root : dword;
   rem : word;
begin
  if length(numberEdit.Text) = 0
  then numberEdit.Text := '0';

  N    := strtoInt(numberEdit.Text);
  root := IntRoot(N,rem);
  rootLabel.Caption      := inttostr(root);
  remainderlabel.Caption := inttostr(rem);
end;

procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if key in chrset then
    begin
      if (key = '$') and (length(numberEdit.Text) > 0)
      then key := #0;
    end
  else key := #0;
end;
```



7 Starter Expert D11

Factorising a number means writing the number as the product of it's prime factors.
Example: $3960 = 2^3.3^2.5.11$
If a quantity can be factorised, it means that it may be rearranged
(*from a single column into 2 columns and even three*.)
See picture below and the explanation at right



EXPLANATION OF FACTORISING
Can we divide 3960 exactly by 2?
$3960 \div 2 = 1980$
Yes it can.
The answer should be a whole number,
and 1980 is 990 :2 (2^3)
Let's try the next prime number, 3:
495 : 3 = 165

165 : 3 = 55 (3^2)
until 10 it doesn't work

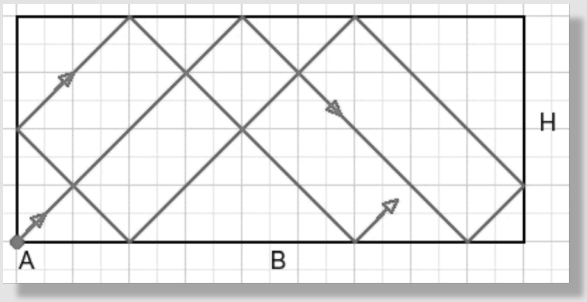
But 11 does, so we get: 5
so here comes:
$3960 = 2^3 * 3^2 * 5 * 11$
$2^3 = (2*2*2) = 8$ $3^2 = (3*3) = 9$
$8 * 9 * 5 * 11$

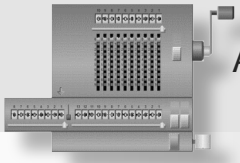
Such a rearrangement is not possible for a prime number.
Exchanging banknotes is an application of factorization in real life.
Simplifying a fraction by dividing both numerator and denominator by common factors is another example.

The common factors of integers
`A` and `B` is written as GCD`(A,B)` {Greatest Common Divisor}
`LCM(A,B)` yields the smallest number
that is a multiple of both `A` and `B`{least Common Multiple}
`GCD(A,B)*LCM(A,B) = A*B`

Problem 1
A supermarket has 255 oranges and 595 apples from which they want to make as much as possible equal packages . Since `GCD(255,595)` = 85 85(`3 + 7`) is the answer.
(85 packages of each 3 oranges and 7 apples)

Problem2
A light beam travels in a room with mirrors.
Which distance is travelled when the beam hits home? (At point A).



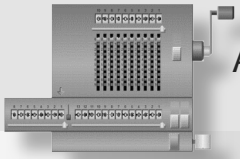


Observe horizontal and vertical movement separately.
After travelling a vertical distance of 2H the beam is at the bottom.
After travelling 2B horizontally the beam is at the left side of the rectangle.
Return at point A is after a joint travel of LCM(2B,2H) distances.
For picture above where B=9 ; H=4
LCM(18,8) = 72 which is distance 36 both horizontally and vertically.
The real distance is (the square root of 2) times 36.

THE FACTORIZATION PROGRAM
The first prime factors are 2 and 3.
Larger prime factors are of the form 6K-1 and 6K+1 where K=1,2,3,.....
because 6K,6K+2,6K+3,6K+4 cannot be prime.
Factorization of number (N) needs division by all possible prime factors and counting the divisions that had a zero remainder.
After trying factors 2,3,5 the factors update by 2 and 4 alternatively (from 6K-1 to 6K+1 to 6K+5).
This saves time. Another time saving consideration is this:
N = Q.d + r {N: number; d: divisor; Q: quotient; r: remainder}
If Q <= d more divisions will not find new factors.
If f is a factor of N, so N = Q*f f will be found for f < Q.
In the factorize procedure the XF (exit flag) is set true to terminate the factorization process.

```
procedure factorize(N : dword);
var N1 : dword;
   count : byte; // same factors
   factor,m : word;
   fplus : byte;
   XF : boolean;
begin
   factor := 2;
   fplus := 2;
   repeat
   count := 0;
   repeat
     m := N mod factor;
     N1 := N div factor;
     XF := N1 <= factor;
     if m = 0 then
       begin
         N := N1;
         inc(count);
       end;
     until  m > 0;

     if count > 0 then report(factor,count);

     if factor < 5
     then factor := (factor shl 1) - 1
     else
        begin
          factor := factor + fplus;
          fplus := 6 - fplus;
        end;
     until XF;
     if N > 1 then report(N,1);
end;
```

The report procedure adds the factor and count to a `TMemo` component.
The GCD of 2 numbers may be obtained from their factorization.
Simply multiply the common factors.
However, there is a more convenient method: the theorem of *Euclides*.



EUCLIDES
The Euclidean theorem amounts to GCD(A,B) = GCD(A-B,B) {A > B}
So, the above operation does not generate any new factors and also no factor is lost.
The next procedure calculates the GCD according to this theorem:

```
function GCD(A,B : dword) : dword;
var H : dword;
begin
  while A > 0 do
    begin
      if A < B then
         begin
           H := A;
           A := B;
           B := H;
         end;
      A := A mod B;
    end;
 result := B;
end;
```

Please refer to the source code for more.

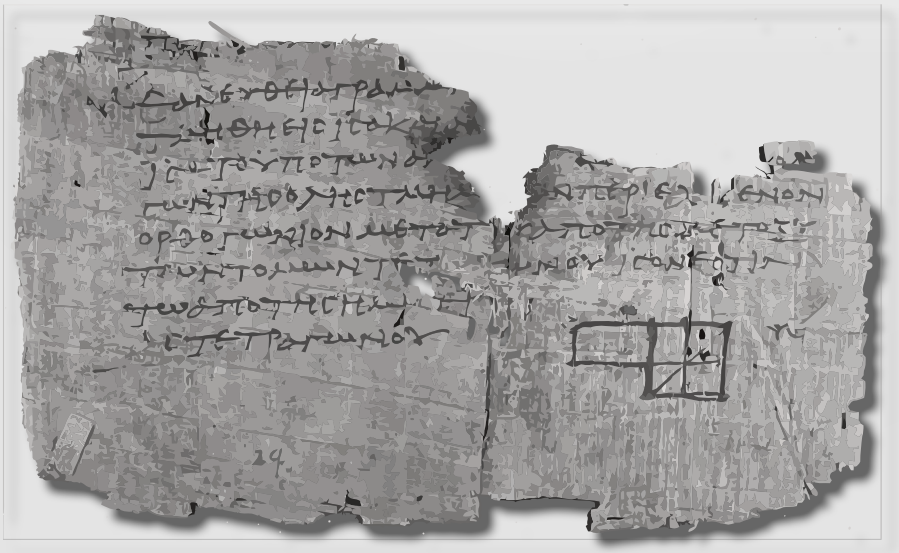
EUCLIDEAN GEOMETRY
Euclidean geometry is a mathematical system attributed to the Alexandrian Greek mathematician Euclid, which he described (though not rigorously by modern standards) in his textbook on geometry, the Elements. Euclid's method is to start with a small set of intuitively appealing axioms, and deducing many other propositions (theorems) from them.
Although many of Euclid's results had been stated by earlier mathematicians,

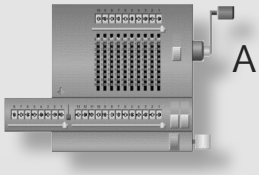
Euclid was the first to show how these propositions could fit into a comprehensive deductive and logical system. The Elements begin with plane geometry, which is still taught in secondary schools as the first axiomatic system and the first examples of formal proof.

It moves on to the solid geometry of three dimensions. Much of the Elements states results of what is now called algebra and number theory, explained in geometrical language.

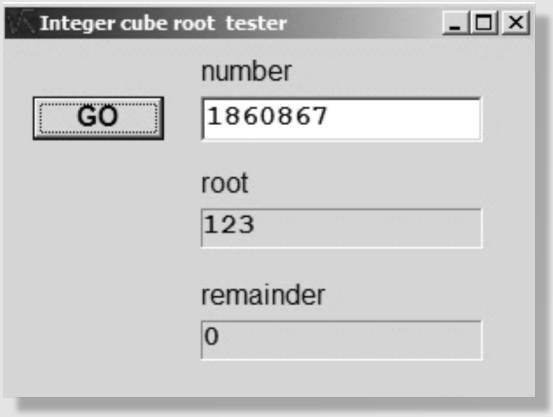
For over two thousand years, the adjective "Euclidean" was unnecessary because no other kind of geometry had been conceived. Euclid's axioms seemed so intuitively obvious (with the possible exception of the parallel postulate) that any theorem proved from them was taken to be true in an absolute, often metaphysical, sense. Today, however, many other non-Euclidean geometries are known, the first of which were discovered in the early 19th century.

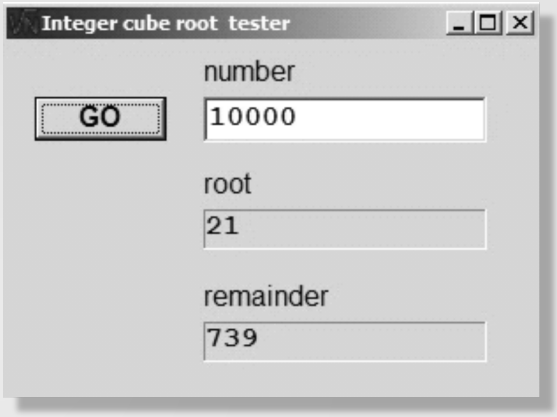
WIKIPEDIA





This article has similarities with the previously described square root algorithm.
Number N = root³ + remainder.
The cuberoot function returns the (cube) root and remainder of N.
N is regarded as groups of 3 bits, counted from the right side.





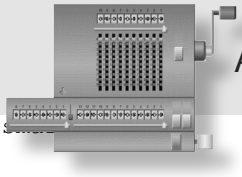
THEORY
$[ab]_2 = 2a+b$
$(2a+b)^3 = 8a^3 + 12a^2b + 6ab^2 + b^3$

At the start, a = 0 and 1 is subtracted from the rightmost bit of the highest group of N.
If successful a=1.
The term $8a^3 = [1000]_2$ was subtracted from N.

Next bit b has to be found, so $12ab^2 + 6ab^2 + b^3$ must be subtracted from N.
If subtraction was possible a becomes $[a1]_2$ else a becomes $[a0]_2$

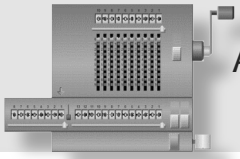
So, to find the next b bit given some value of a requires subtraction of
$12a^2b + 6ab^2 + b^3$ and because b =1 is being tested the substrahend becomes:
$12a^2 + 6a + 1 = [a0]([a0] + 1)*3 + 1$
Note: $[a0]_2 = 2a$

Each trial subtraction yields a next bit for the root.
For 30 bit numbers, these steps are repeated 10 times to generate a 10 bit result.
Note: cube roots may be taken from negative numbers as well.



THE PROGRAM

```
function CubeRoot(N : longInt; var rem : longInt) : s
// -1,073,741,823 <= N <= 1,073,741,823
// return integer cube root of N and remainder rem
const max = 1073741824;

var NX,sub,r2 : longInt;
    root : word;
    p : shortInt;
    neg : boolean;

begin
  if (N <= -max) or (N >= max) then
    begin
      result := 0;
      rem := N;
      exit;
    end;

  if N < 0 then
    begin
      N := -N;
      neg := true;
    end
  else neg := false;

  p := 30;
  root := 0;

  repeat
   r2 := root shl 1;
   sub := r2*(r2+1)*3+1;
   sub := sub shl p;
   root := root shl 1;
   NX := N - sub;

  if NX >= 0 then
    begin
      N := NX;
      root := root or 1;
    end;

  p := p - 3;
  until p < 0;

  if neg then
    begin
      root := -root;
      N := -N;
    end;

  result := root;
  rem := N;
end;

procedure TForm1.GoBtnClick(Sender: TObject);

var N, rem : longInt; root  : smallInt;

begin
  if length(numberEdit.Text) = 0 then
   numberEdit.Text := '0';

  N := strtoInt(numberEdit.Text);
  root := cubeRoot(N,rem);
  rootLabel.Caption := inttostr(root);
  remainderlabel.Caption := inttostr(rem);

end;
```



```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
var OK,mt : boolean;

begin
  OK := false;
  with numberEdit do
    begin
      mt := length(text) = 0;
      case key of
        #8     : OK := true;
        '1'..'9': OK := true;
        '0'    : OK := not(mt);
        '$'    : OK := mt;
        'a'..'f' : OK := text[1] = '$';
        '-'    : OK := mt;
      end;
      if OK = false then key := #0;
    end;
end;
```

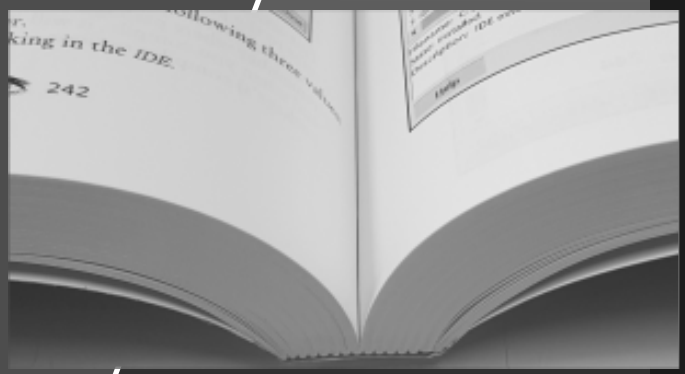
LAZARUS HANDBOOK POCKET

Price: € 26,50

Excluding VAT and Shipping

934 PAGES
PDF & INDEX
INCLUDED

- English
- Printed black & white
- 2 Volumes
- PDF included
- 934 Pages
- Weight: 2kg
- Extra protected
- Including
  40 Example
  projects and
  extra programs

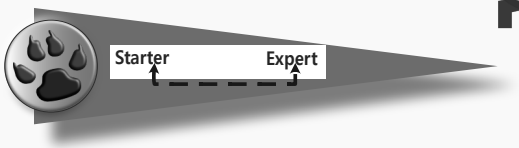


BlaisePascalMagazine
PDF viewer included

https://www.blaisepascalmagazine.eu/product-category/books/

THE LAZARUS DEBUGGER
– An introduction and tutorial.

The Lazarus debugger: Brings your application back to life.



PART 1: INTRODUCTION AND FIRST STEPS

WHAT IS A DEBUGGER?

Many programmers – when they write their first code and encounter their first bug – will use "`writeln`"-debugging to locate the misbehaving code. This means to put "`writeln`" statements every few lines into the code, re-compile it and then run it. Looking at the output, refine it, and repeat it. Until eventually the wrong code is found. After that, it's clean-up time and all the writeln need to be removed again.

A debugger can help doing exactly the same steps, but doesn't require the code to be altered. And a debugger can do a lot more than just the above steps. In this tutorial we are going to look at using a debugger. The first article will show how to replace writeln and then in further articles we will discover what other tools the Lazarus debugger offers.

SETTING UP YOUR IDE AND PROJECT FOR DEBUGGING

We start with a quick look at the minimum required settings. We will look at more details in a later part of the series.

The IDE comes with default settings for debugging, which you can keep as they are. However if you did upgrade from older Lazarus versions, or made changes to those settings yourself, then you should check them. To do so, this, in the IDE you need to chose Options from the Tool menu. The relevant settings are on the **Debugger → Debugger Backend** page. *See Figure 1 on this page*, if you are on Windows or Linux, the selection on the top of the page should read "**FpDebug [FpDebug internal Dwarf-debugger]**". The important part is the name in the square brackets. If you are on Mac it should read "[LLDB debugger (with `fpdebug`) (Beta)]".

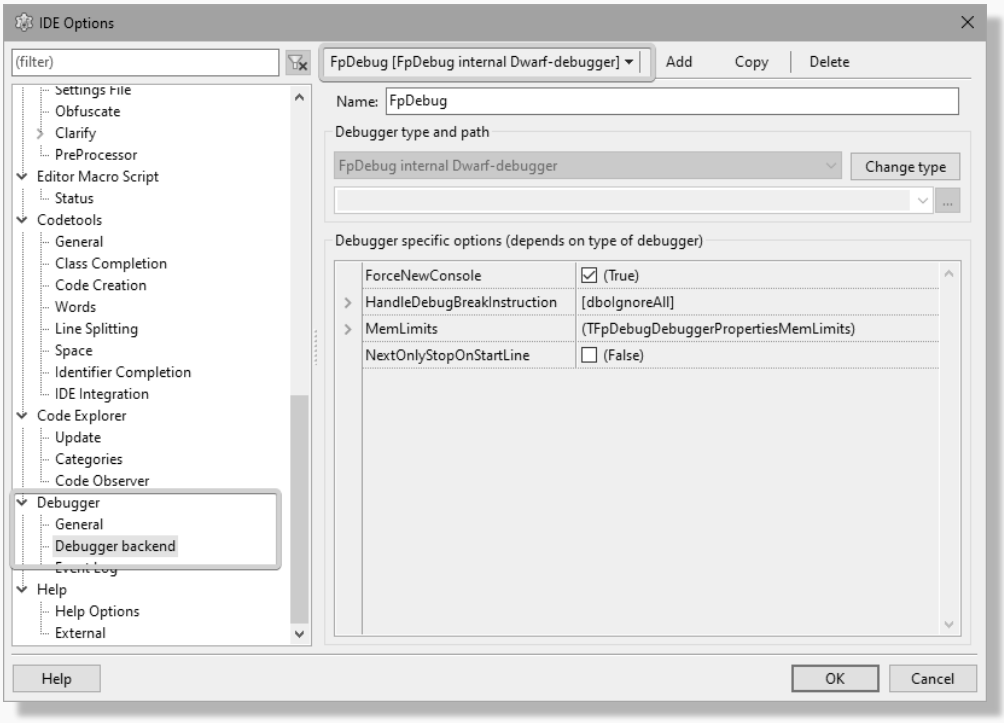


Figure 1

There are also settings for each project that must be set.
Here again, by default most of them are already set. *See figure 2 on this page*, the setting to "**Generate info for debugger**" must be enabled. And the "**Debugger info**" which will be set to "**default**" should be changed to "**Dwarf 3**".

However as long as the generation is enabled, if you don't make that change now, the IDE will ask you later to chose a value (Figure 3).
If that dialog appears, chose "**Dwarf 3**".
NOTE that the layout of the dialog will be different in Lazarus 2.4.

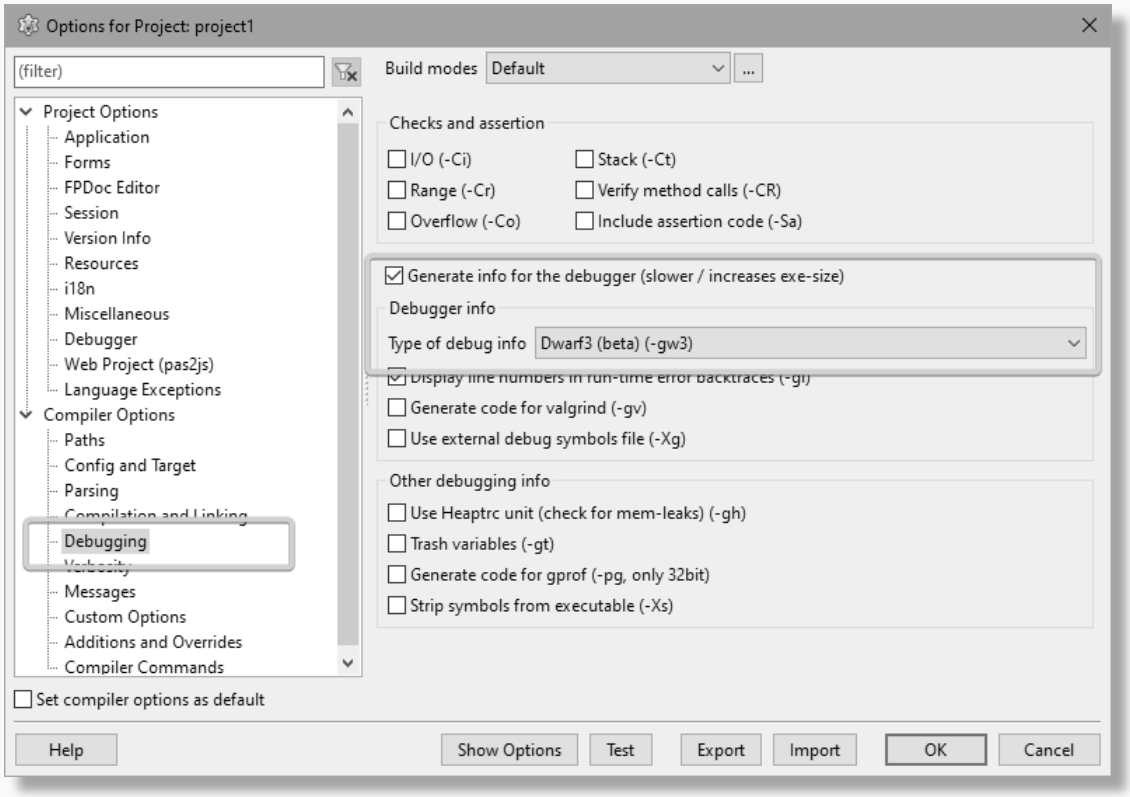


Figure 2

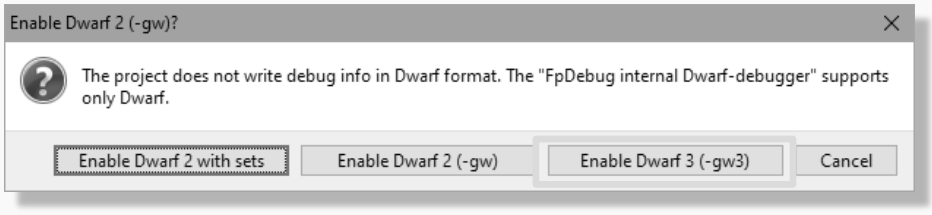


Figure 3

Let's start debugging
For this session we will debug the following example

```
 1.  program example_1;
2.   const SAMPLE_TEXT = 'Just some random text and some letters A B C';
3.
4.   function FindNextWord(AnInput: Ansistring; ACurrentWordStart: Integer;
5.    out ANextStart, ANextLength: Integer): Boolean;
6.   var
7.    AInputLength: SizeInt;
8.   begin
9.    ANextStart := ACurrentWordStart;
10.     while   (ANextStart > 0) and
11.             (ANextStart < Length(AnInput)) and
12.             (AnInput[ANextStart + 1] <> '')
13.   do
14.     inc(ANextStart);
15.   inc(ANextStart);
16.   while (ANextStart < Length(AnInput)) and
17.             (AnInput[ANextStart] = '')
18.   do
19.    inc(ANextStart);
20.
21.   Result := ANextStart <= Length(AnInput);
22.   if not Result then
23.    exit;
24.
25.   ANextLength := 0;
26.   AInputLength := Length(AnInput);
27.   while   (ANextStart + ANextLength < AInputLength) and
28.           (AnInput[ANextStart + ANextLength] <> '')
29.   do
30.    inc(ANextLength);
31.  end;
32.
33.  var
34.   AStart, ALen: Integer;
35.   AWord: Ansistring;
36.  begin
37.   while FindNextWord(SAMPLE_TEXT, AStart, AStart, ALen) do begin
38.    AWord := copy(SAMPLE_TEXT, AStart, ALen);
39.    writeln(AWord);
40.   end;
41.   ReadLn;
42.  end.
```

It will parse the text into words or letters separated by spaces.
Then it will write each word to the console.
And here is the output:

```
Just
some
random
text
and
some
letters
A
B
←  Empty Last Line
```

Looks good… Except, the last line is wrong. There should be a "C", but it is empty.

In order to debug the problem we reflect where we would put the `writeln`, that is where we want to see the status – such as a variable's value - of the program.
We might want to observe the values of `AStart` and `ALen` after each call to `FindNextWord`.
To have the debugger allow us to see those value we need it to pause the app, and we can tell the debugger by setting a breakpoint.
Using the left mouse button we click on the gutter (the grey area on the left) in front of the line numbers. A red dot as in *Figure 4 on this page* will appear.

```
  while FindNextWord(SAMPLE_TEXT, AStart, AStart, ALen) do begin
    AWord := copy(SAMPLE_TEXT, AStart, ALen);
39  writeln(AWord);
40 end;
```

Figure 4

Now we can run the application using `F9` or the green arrow button  . The red point – indicating the breakpoint – will change as the program execution reaches that line.

A green arrow in the breakpoint (*Figure 5 on this page*) and a grey background color for the line will tell us that the debugger paused the app on that line.

```
begin
  while FindNextWord(SAMPLE_TEXT, AStart, AStart, ALen) do begin
    AWord := copy(SAMPLE_TEXT, AStart, ALen);
39  writeln(AWord);
40 end;
```

Figure 5

Now we want to see what the values of `AStart` and `ALen` are. One way do to this is by hovering the mouse over one of the variables and wait for the hint to appear (*Figure 5 on this page*). The hint will show the current value of the variable.
Since we executed `FindNextWord` for the first time, we expect the first word, which starts at `position 1` in the string. The value we see is therefore correct.

```
AStart, ALen));
AStart = 1
var AStart: Integer = longint
project1.lpr(34,3)
```

Figure 6

We can also open the Watches window. `Ctrl-Alt-W` or from the menu View → Debug Windows → Watches. Using the + button we can add the 2 variables by their names: `AStart` and `ALen` (*Figure 7 on this page*). We see the same value for `AStart`. And for `ALen` we see 4, which correlates with the first word "Just".

Watches

Expression	Value
AStart	1
ALen	4
AWord	'Just'

Figure 7

Since the values are what we expected, we can press `Run/F9` and let the program continue to run. It will be paused again as the breakpoint is reached on the next iteration of the loop. This time the value are `AStart=6` And for `ALen=4` for the word "some". That is also correct.
We continue this for each word and check the values.
Eventually we should expect the letter "`C`", which is missing from the output.
For this the watches window *(See figure 8 on this page) will show us:*

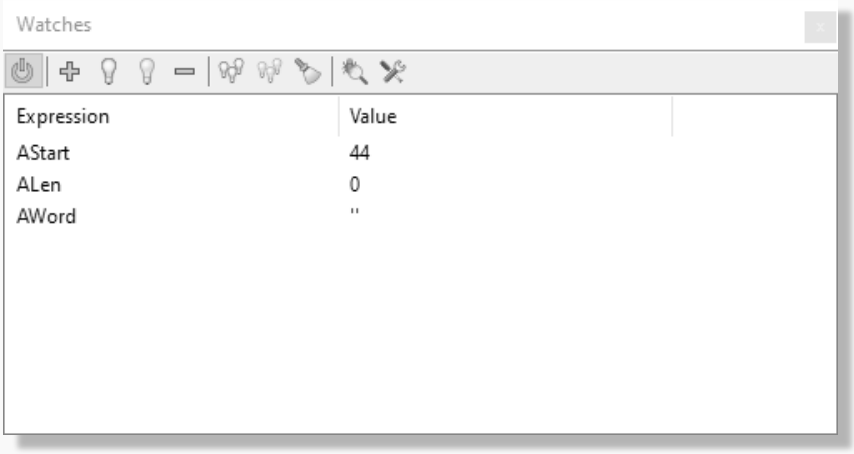


Figure 8

We can check that `AStart` is correct, but `ALen` certainly is not, we expected `1` and got `0`.
With that we can refine our search for the error, we know that `FindNextWord` returns the wrong length. We will need to set a breakpoint on line 27 where we can inspect the source-values from which the length is calculated.

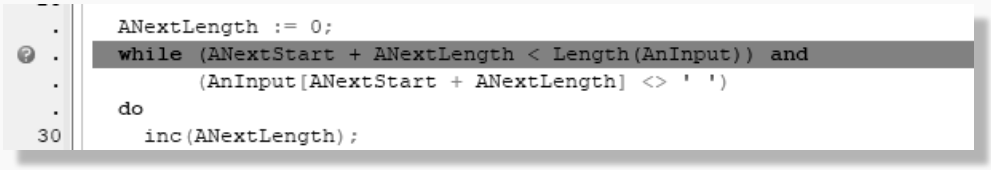


Figure 9

Before we set the new breakpoint, we will restart the program and use the existing breakpoint at line 39 to run the program until the watches window shows us that the current "`AWord`" is "`B`".
Then we know the next iteration will be the one that fails to get "`C`".
At that time the previous breakpoint on line 39 is no longer needed. If we click it again with the left mouse button it will be removed. Now we also set the new breakpoint at line 27 (*See figure 9 on this page*) and then run the application to go to line 27.

We can make those changes while the application is executing without having to recompile it.
And this is a difference to using writeln.
We couldn't have inserted a writeln mid-execution.
When we reach the breakpoint at line 27, we could add the local variables to the watches window.
However, we can just open the Locals window using `Ctrl-Alt-L`, and it will show us all the local variables.

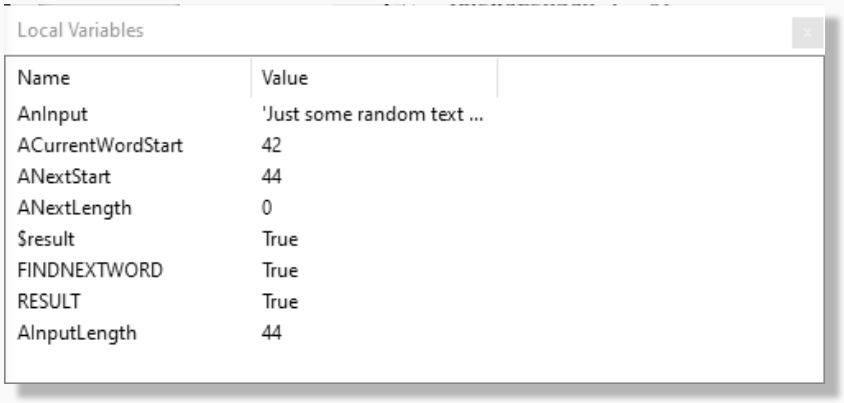


Figure 10

We can see that `ANextStart=44`. This is the value that will be returned as the start of "`C`" and is the same as we had seen in the watches window for `AStart` before.
`ANextLength` is still zero as it was just initialized.
It needs to be incremented once by the loop at the start of which we are paused.
So the while condition needs to be true, and execute the "`inc(ANextLength)`".
But we know already it will return zero.
Lets add the first part of the condition to the watches window, select the text "`ANextStart + ANextLength < AinputLength`" in the editor, and press **Ctrl-F5** to add it to watches.

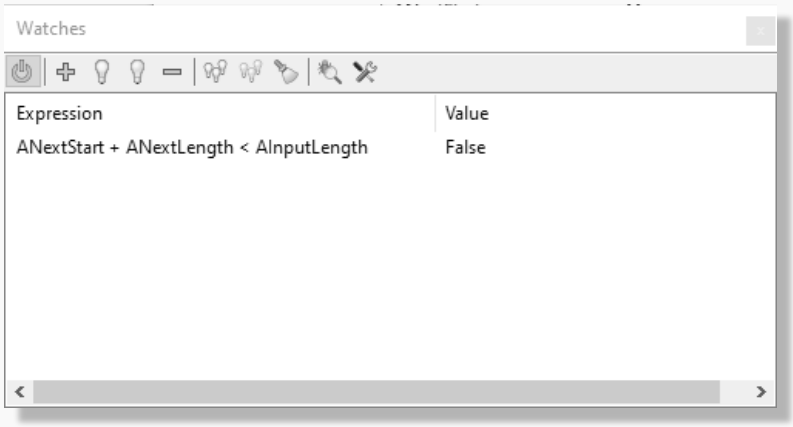


Figure 11

And we can see that it returns false (*Figure 11 on this page*).
This is not what it should be. So we found something that is wrong.
If we go back to the Locals window, we can see that not only is `ANextStart=44`, but we also have `AnInputLength=44`.
So `AnextStart+ANextLength` can not be less than `AnInputLength`.
Since we can access the last char at `pos 44` in the string, the condition should not check for "**less than**" but for "**less or equal than**". We found the problem and we can change the line to have a "<="

Restart line numbers at line 27

```
43.   while (ANextStart + ANextLength <= AInputLength) and
44.         (AnInput[ANextStart + ANextLength] <> ' ')
45.   do
46.     inc(ANextLength);
47.
```

SUMMARY

In this article we have explored the following features.

- Setting and unsetting breakpoints
  - Mouse click in gutter
  - F5
  - Menu: Run → Add Breakpoint → Source Breakpoint
- Running to breakpoint
  - Run button
  - F9
  - Menu: Run → Run
- Using hint to see variable values
  - Mouse hovering
- Using Watches window to see variables and expression results
  - Ctrl-Alt-W
  - Menu: View → Debug Windows → Watches
  - [+] Button to add watches
  - Ctrl-F5 to add editor selection or word at cursor
- Using Local window.
  - Ctrl-Alt-L
  - Menu: View → Debug Windows → Locals

In the next article we will look at "stepping", which allows us to execute the program line by line.

Starter Expert

ABSTRACT
In a previous contribution, we've shown how to create an index of words in a PDF file. In this article, we'll show how to use this index in a search program and use that to implement a search in a website.

1 INTRODUCTION

In the first article on searching PDF files, we showed how to create a PDF indexer: a program to analyse a bunch of PDF files, and store occurrences of words in the PDF files in a database.
All this was accomplished with classes that are part of Free Pascal: the end result was that we have a database with all the words in the PDF files and the locations on which these words occur.
In this article, we'll show how to query this database: we'll make a small webpage in which the user can enter a search term.
The search term is sent to the server, which will reply with a list of pages and the title of the article of which the page is a part. The user can then choose which article he wants to read.
The correct PDF will be downloaded and opened on the correct page. Once the PDF is downloaded, the user can search locally in the PDF. For simplicity, the indexed PDF files are assumed to be in a directory on the server, but they can of course be located in a database or distributed over many directories.
To help the user, we'll also create an auto-complete mechanism for the search box: as soon as the user typed 2 letters, the program will present a list of words that contain the letters he or she typed. The user can then select the complete words and let the backend search for it. This will not only save typing, but will also increase the accuracy of the search.

2 THE SERVER SEARCH PART: OVERVIEW

The server is a simple HTTP application which has 3 functions:

❶ Return a list of words from the database, based on some letters in the word.
❷ Return a list of matches for a word, returning the pdf name, the pages with a match and the article name to which the page belongs.
❸ Return a PDF file.
To make this HTTP application, we start a new HTTP application as usual in the Lazarus IDE.

The main program source code is changed somewhat, so it reads as follows:

```
begin
  ConfigureApp;
  {$ifndef usecgi}
  Application.Port:=3010;
  {$endif}
  Application.Initialize;
  Application.Run;
  ASearch.Free;
end.
```

The configureApp is a call to a routine that creates the search mechanism (the 2 first statements in the routine) and configures the HTTP router, which is responsible for routing the 3 HTTP requests that the application supports:

```
Var
   aSearch : TSearcher;
Procedure ConfigureApp;
begin
   // Create search mechanism
   aSearch:=TSearcher.Create(Application);
   aSearch.Init;
   // Register routes for the search mechanism
   HTTPRouter.RegisterRoute('/search',@aSearch.DocSearch,true);
   HTTPRouter.RegisterRoute('/list',@aSearch.WordList,False);
   // Set default file downloader
   DefaultFileModuleClass:=TCorsFileModule;
   // Where are the PDF files located.
   RegisterFileLocation('pdf',aSearch.PDFLocation);
end;
```

The last 2 lines use a descendent of a standard Free Pascal file down loader module (`TFPCustomFileModule`)
named `TCorsFileModule` to set up a download functionality for the PDF files.
NOTE that the path must of course be matched

The `TCorsFileModule` is there to enable CORS:

```
TCorsFileModule = Class(TFPCustomFileModule)
   Procedure HandleRequest(ARequest: TRequest; AResponse: TResponse); override;
end;

procedure TCorsFileModule.HandleRequest(ARequest: TRequest; AResponse: TResponse);
begin
   Cors.Enabled:=true;
   if Cors.HandleRequest(aRequest,aResponse) then exit;
   inherited HandleRequest(ARequest, AResponse);
end;
```

The standard `TFPCustomFileModule` does not allow CORS requests,
so a descendent is made that allows this. It enables the CORS support (standard part of every FCL-Web module) and if a CORS preflight request is detected, it is handled. If it is a regular request, the PDF download request is handled: We need not do more than this to serve the PDF files from a URL formed as:

`http://localhost:3010/pdf/somepdf.pdf`

(*the host & port may differ depending on your setup*)
all the rest will be handled by the file module itself.

The actual searching happens in the TSearcher class, which is defined as follows:

```
TSearcher = Class(TComponent)
private
  procedure ConfigSearch(aRequest: TRequest; aResponse: TResponse);
  procedure ConfigWordList(aRequest: TRequest;
                           Out aContaining : UTF8string;
                           Out Partial : TAvailableMatch;
                           Out aSimple : Boolean);
  procedure ConnectToDatabase;
  procedure DisConnectFromDatabase;
  function FindArticle(Position: Integer;
                           const aFile: String): TArticleData;
  function SearchDataToJSON(aID: Integer;
                           const aRes: TSearchWordData): TJSONObject;
  procedure SendJSON(J: TJSONObject; aResponse: TResponse);
  procedure SetupMetadata;
  procedure LoadArticles;
Protected
  function InitSearch(aResponse: TResponse): Boolean;
  function SetupDB(aIni: TCustomIniFile): TCustomIndexDB;
  Property DB : TCustomIndexDB;
  Property Search : TFPSearch;
  Property MinRank : Integer;
  Property FormattedJSON : Boolean;
  Property Articles : TArticleDataArray;
Public
  Procedure Init;
  procedure DoDBLog(Sender: TSQLConnection; EventType: TDBEventType;
                                          const Msg: String);
  Function CheckParams(aRequest : TRequest; aResponse : TResponse) : Boolean;
  Function CheckSearchParams(aRequest : TRequest; aResponse : TResponse) : Boolean;
  Procedure DocSearch(aRequest : TRequest; aResponse : TResponse);
  Procedure WordList(aRequest : TRequest;aResponse : TResponse);
  Property AllowCors : Boolean;
end;
```

The `DocSearch` and `WordList` routines are the entry points for handling the browser requests, we'll treat them in a moment. The `Init` which is called during program start initializes the search engine. It reads a configuration file with the settings for the database connection and some further settings that configure the search mechanism. The configuration file is searched in several directories, in the `GetConfigFileName` routine, which we will not present here.

```
procedure TSearcher.Init;
Const
   // Adapt this default to your setup....
   DefaultLocation = '/home/michael/Documents/pdf/blaise/';
Var CFN : String; aIni: TMemIniFile;
begin
   CFN:=GetConfigFileName;
   aIni:=TMemIniFile.Create(CFN);
   try
     FFormattedJSON:=aIni.ReadBool('search','formatjson',False);
     FDefaultMinRank:=aIni.ReadInteger('search','minrank',1);
     FDefaultMetadata:=aIni.ReadBool('search','metadata',true);
     FAllowCors:=aIni.ReadBool('search','allowcors',true);
     FPDFLocation:=aIni.ReadString('Files','PDF',DefaultLocation);
     FDB:=SetupDB(aIni);
     FSearch:=TFPSearch.Create(Self);
     FSearch.Database:=FDB;
   finally
     aIni.Free;
   end;
   SetupMetadata;
end;
```

4 variables are initialized from the configuration file, they influence the returned results to the client requests:

FFormattedJSON Should the JSON returned be formatted or not.
FDefaultMinRank Minimum rank for a word occurrence to be included in the result.
FDefaultMetadata Should metadata be returned in absence of a parameter that specified it.
FAllowCors Should CORS HTTP headers be set on the result ?
FPDFLocation The location of the PDF files on disk.

The **FDB** variable is an instance of `TCustomIndexDB` as presented in the previous article: it handles the database connection for the `TFPSearch` instance which does the actual search in the database.
The `TFPSearch` class is part of the `fpIndexer` unit presented in the previous article. It handles searching in the index database. The class is declared as follows:

```
TFPSearch = class (TComponent)
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  function Execute: int64;
  procedure AddResult(index: integer; AValue: TSearchWordData);
  procedure SetSearchWord(AValue: UTF8String);
  Function GetAvailableWords(out aList : TUTF8StringArray;
                             aContaining : UTF8String;
                             Partial : TAvailableMatch) : Integer;
  property Count: integer;
  property RankedCount: integer;
  property Results[index: integer]: TSearchWordData ;
  property RankedResults[index: integer]: TSearchWordData;
published
  property Database: TCustomIndexDB;
  property Options: TSearchOptions;
  property SearchWord: TWordParser;
  Property UsePositionInRank : Boolean;
end;
```

The published properties can be used to configure the search mechanism:
The `Database` property represents the connection to the database.
`Options` can be set to `soContains` in which case the search will be for words containing the search term.
`UsePositionInRank` will determine whether the position should be taken into account when ranking words (normally, only the file/name is taken into account)
The methods of the class have the following purpose:

`SetSearchWord` Set the word to search for.
`Execute` Start the search. Returns the number of found results.
`AddResult` add a match to the results index.
`GetAvailableWords` Get the list of words that contain the word `aContaining`.
The matching mechanism is determined by the value of the `Partial` parameter: one of
`amAll` (all words),
`amExact` (exact match on the search term),
`amContains` (words that contain the search term),
`amStartsWith` (words that start with the search term).

The results are returned in the `aList` array, and the function returns the number of results.

The public properties of the class allow to examine the results of the search. The `Results` and `Count` properties describe the raw results of the search operation.

The last line of the Init routine is a call to `SetupMetadata`. This routine sets up metadata definitions for the 2 REST result sets that can be handled by the server program: the program is set up in such a way that the result it returns is a JSON structure that can be consumed by the `TJSONDataset` implementation in pas2js.

The JSON structure that is consumed by a `TJSONDataset` dataset contains a "meta-data" element, that describes the data. Rather than creating this structure every time we create a result set, we create it once and reuse it when a result is being returned.

```
procedure TSearcher.SetupMetadata;
begin
  FMetadata:=TJSONObject.Create([
    'root', 'data',
    'idField','id',
    'fields',TJSONArray.Create([
      TJSONObject.Create(['name','id','type','int']),
      TJSONObject.Create(['name','rank','type','int']),
      TJSONObject.Create(['name','articlePage','type','int']),
      TJSONObject.Create(['name','articleIssue',
                          'type','string','maxlen',10]),
      TJSONObject.Create(['name','articleAuthor','
                          type','string','maxlen',127]),
      TJSONObject.Create(['name','articleTitle',
                          'type','string','maxlen',255]),
      TJSONObject.Create(['name','url',
                          'type','string','maxlen',100]),
      TJSONObject.Create(['name','context',
                          'type','string',
                          'maxlen',MaxContextLen]),
      TJSONObject.Create(['name','date','type','date'])
    ])
  ]);
  FWordsMetadata:=TJSONObject.Create([
    'root', 'data',
    'idField','id',
    'fields',TJSONArray.Create([
      TJSONObject.Create(['name','id','type','int']),
      TJSONObject.Create(['name','word',
                          'type','string','maxlen',100])
    ])
  ]);
end;
```

3 THE SERVER SEARCH PART: THE WORD LIST FOR AUTO-COMPLETION

The `WordList` routine is called whenever the user types a search term, and the browser wants to show a completion list.
The `WordList` resource accepts several query parameters:

`q` (for query),
this mandatory parameter contains the letters typed by the user.

`t` (for type)
determines which words will be returned: The value is one of
`all` will return all words, the value of q must be empty.
`contains` will return all words that contain the value of `q`.
`exact` will return only an exact match: this allows to determine whether the word is present in the database.
starts with will return all words that start with the value of `q`.

`s` a boolean (0 or 1), if `True`, then the return only contains an array of words. If `False`, then an id field is returned as well.

`m` a boolean (0 or 1): include metadata in the response or not. If omitted, the default as in the config file is taken.

When invoked, it starts by initializing the response that will be sent to the browser, including the CORS headers for optional CORS support, and then proceeds by checking the query parameters: The `CheckSearchParams` routine checks whether the supplied query parameters are valid.
If not, it puts an error code in the HTTP request and returns `False`.
It will not be presented here. The `InitSearch` routine simply checks whether the database is available.

```
procedure TSearcher.WordList(aRequest: TRequest; aResponse: TResponse);
Var
    I : Integer;
    J : TJSONObject;
    A : TJSONArray;
    w,  aContaining  : UTF8String;
    aPartial         : TAvailableMatch;
    aSimple          : Boolean;
    aList            : TUTF8StringArray;
begin
  aResponse.ContentType:='application/json';
  if AllowCORS then
    AResponse.SetCustomHeader('Access-Control-Allow-Origin','*');
  if not CheckSearchParams(aRequest,aResponse) then exit;
  if not InitSearch(aResponse) then exit;
  ConfigWordList(aRequest,aContaining,aPartial,aSimple);
  FSearch.GetAvailableWords(aList,aContaining,aPartial);
  J:=TJSONObject.Create;
  try
    if FIncludeMetadata then J.Add('metaData',FWordsMetadata.Clone);
    A:=TJSONArray.Create;
    if aSimple then
      For W in aList do A.Add(W)
    else
      begin
        For I:=0 to Length(aList)-1 do
          A.Add(TJSONObject.Create(['id',I+1,'word',aList[i]]));
      end;
    J.Add('data',A);
    SendJSON(J,aResponse);
    finally
    J.Free;
  end;
end;
```

The `ConfigWordList` routine extracts the query parameters from the request, and makes sure the returned values (`aContaining`, `aPartial`, `aSimple`) are consistent.
One could integrate this functionality with the `CheckSearchParams` routine. The `FSearch.GetAvailableWords` function is used to get the actual results.
Once the results are in, the rest of the routine is just constructing the JSON to be returned to the browser: here the 'simple' parameter is used to determine the structure of the elements in the result array.
The `SendJSON` simply returns the constructed JSON in the HTTP response:

```
procedure TSearcher.SendJSON(J : TJSONObject; aResponse: TResponse);
begin
  if FormattedJSON then
   aResponse.Content:=J.FormatJSON()
  else
    aResponse.Content:=J.AsJSON;
  aResponse.ContentLength:=Length(aResponse.Content);
  aResponse.SendContent;
end
```

With this, the functionality so get a word list is complete. Being a simple HTTP request, it can be tested simply in the browser using for example the following URL:
`http://localhost:3010/list?q=class&t=contains&s=1`
A sample result for the word `class` can be seen in *figure 1 on page 7 of this article*.
NOTE that the browser recognizes the result is JSON based on the content-type, and formats the result accordingly.

4 THE SERVER SEARCH PART: THE LIST OF PAGES WITH WORD MATCHES

The second call needed in the server will find all occurrences for a word in the indexed PDFs. The PDF indexer stores for each word the name of the file in which the word occurs, and the page on which the word occurs.

For the application that we wish to make, the page number needs to be matched with an article: To match a page with an article, we have a table with the PDF name (*and issue*), starting and ending pages, author and title. Since there is a limited amount of articles and the list changes not so often, it makes sense to cache the list in the server. We'll do so in the following array of records:

```
TArticleData = record
   StartPage,
   EndPage : Integer;
   Author,
   Title,
   Issue,
   PDF : String;
   function Match(aPDF : String; aPage : Integer) : boolean;
end;
TArticleDataArray = Array of TArticleData;
```

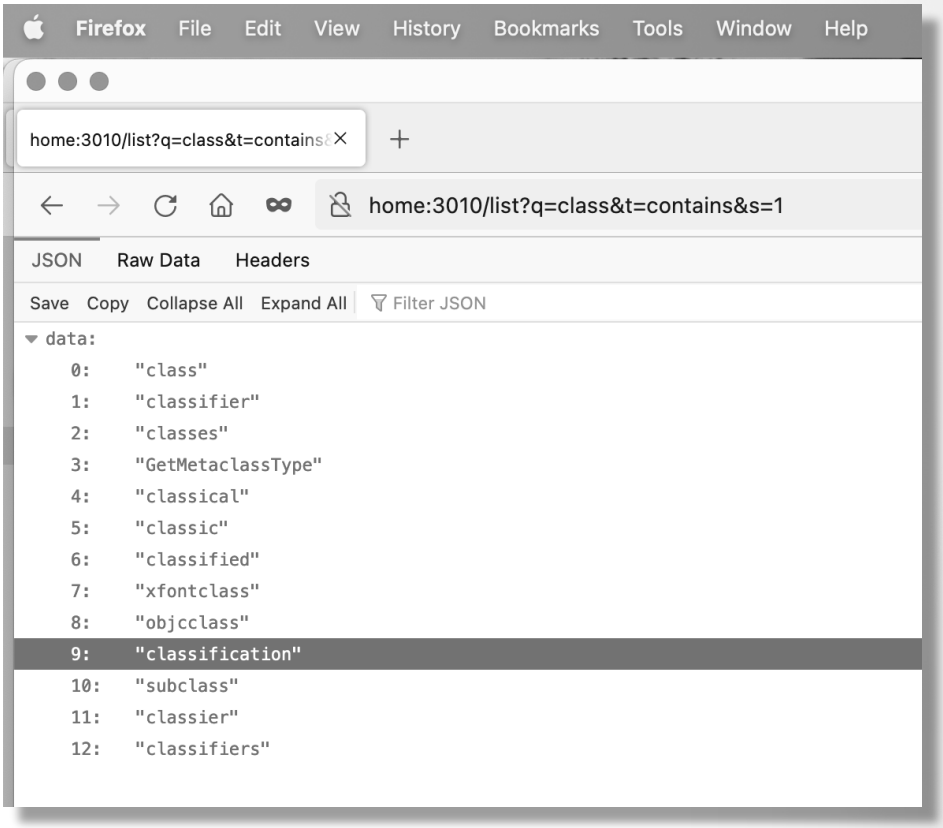


Figure 1: The result of a word list request

This information will matched with the result of the PDF index data. The HTTP entry point for the search mechanism is the 'search' path, linked to the `PDFSearch` method. Similar to the `WordList` entry point, the `PDFSearch` method accepts some options through the query variables:

- `q` (for query), this mandatory parameter contains the word to search for.
- `r` (for rank) determines the minimal rank a match must have for the match to be included in the result. The default is zero.
- `c` a boolean (0 or 1), if `True`, words containing the query will be returned. If `False` (the default) then only an exact word match is returned.
- `m` a boolean (0 or 1): include metadata in the response or not. If omitted, the default as in the config file is taken.

The `PDFSearch` method is surprisingly short, and has the same overall structure as the `WordList` method. It starts by preparing the response, setting CORS headers and checking the parameters. The next step is to initialize and configuring the search. If the initializing goes wrong, the routine exits.

```
procedure TSearcher.PDFSearch(aRequest: TRequest; aResponse: TResponse);
Var
  I : Integer;
  J : TJSONObject;
  A : TJSONArray;
begin
  aResponse.ContentType:='application/json';
  if AllowCORS then
     AResponse.SetCustomHeader('Access-Control-Allow-Origin','*');
  if not CheckParams(aRequest,aResponse) then exit;
  if not InitSearch(aResponse) then
    begin
      aResponse.Code:=500;
      aResponse.CodeText:='Internal error';
      aResponse.SendResponse;
      exit;
    end;
ConfigSearch(aRequest,aResponse);
```

The `ConfigSearch` will configure the FPC index search object from the query variables:

```
procedure TSearcher.ConfigSearch(aRequest: TRequest; aResponse: TResponse);
Var
    S : string;
    O : TSearchOptions;
    B : Boolean;
begin
  FMinRank:=StrToIntDef(aRequest.QueryFields.Values['r'],0);
  if FMinRank=0 then
    FMinRank:=FDefaultMinRank;
  S:=aRequest.QueryFields.Values['m'];
  if (S='') or Not TryStrToBool(S,FIncludeMetaData) then
    FIncludeMetaData:=FDefaultMetaData;
    FSearch.SetSearchWord(aRequest.QueryFields.Values['q']);
  O:=[];
  S:=aRequest.QueryFields.Values['c'];
  if (S<>'') and TryStrToBool(S,B) and B then
    Include(O,soContains);
  FSearch.Options:=O;
end;
```

When the call to the `CongfigSearch` routine has configured the search mechanism, the `PDFSearch` routine starts the actual search:
The call to `FSearch.Execute` will execute the necessary SQL statements on the index database. The results will be available in the `Results` and `RankedResults` array properties. The latter will be combined with the articles array in the `SearchDataToJSON` routine to form the actual result set:

```
FSearch.Execute;
A:=nil;
J:=TJSONObject.Create;
try
  if FIncludeMetadata then
    J.Add('metaData',FMetadata.Clone);
  A:=TJSONArray.Create;
  For I:=0 to Search.RankedCount-1 do
    begin
    if Search.RankedResults[I].Rank>=MinRank then
     A.Add(SearchDataToJSON(I+1,Search.RankedResults[I]));
    end;
  J.Add('data',A);
  SendJSON(J,aResponse);
finally
  J.Free;
end;
```

The result is combined with the metadata if needed, and is sent to the browser with the `SendJSON` command, just as in the `WordList` routine.
The search data is combined with the article list in the `SearchDataToJSON` routine, which starts by finding the article which matches the page of the word match (*stored in Position*). The found article and the search result are then unified in a single JSON structure:

```
function TSearcher.SearchDataToJSON(aID: Integer; const aRes: TSearchWordData): TJSONObject;
Var
   Article : TArticleData;
begin
   Article:=FindArticle(aRes.Position,aRes.URL);
   Result:=TJSONObject.Create([
     'id',aID,
     'rank',aRes.Rank,
     'articlePage', aRes.Position,
     'articleTitle', Article.Title,
     'articleAuthor', Article.Author,
     'articleIssue', Article.Issue,
     'url',aRes.URL,
     'context',ares.Context,
'    date',FormatDateTime('yyyy"-"mm"-"dd"T"hh":"nn":"ss',aRes.FileDate)
   ]);
end
```

The routine to find the articles is simplicity itself:

```
function TSearcher.FindArticle(Position: Integer; const aFile: String): TArticleData;
Var
   A : TArticleData;
begin
   Result:=Default(TArticleData);
   if Length(Articles)=0 then
      LoadArticles;
   for A in Articles do
     if A.Match(aFile,Position) then
       Exit(A);
end;
```

The `LoadArticles` routine executes a simple select SQL query and stores the results in the `Articles` array, part of the `TSearcher` class. The interested reader can consult the code supplied with the articles for the details. The `Match` method of the `TArticleData` record returns `True` if the article is located in the PDF and contains the page number:

```
function TArticleData.Match(aPDF : String; aPage : Integer) : boolean;
begin
  Result:=SameText(Pdf,aPdf)
    and (StartPage<=aPage)
    and (aPage<=EndPage)
end;
```

With this routine we've covered the search mechanism in the index database. Again, the result can easily be tested in the browser, as shown in *figure 2 on page 10 of this article.*

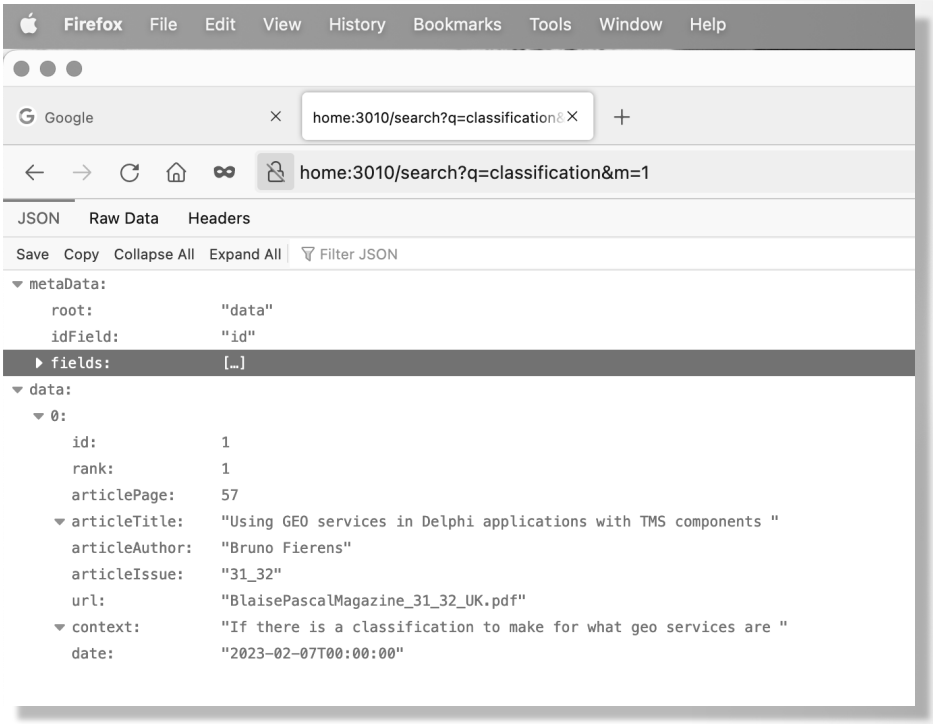


Figure 2: The result of a search request

5 THE BROWSER APPLICATION

Now that we have an API in place for getting word lists and matches of a word in our collection of PDF documents, we can create a browser application that uses these API's to query and show a collection of PDF documents.

To do so, we will refactor the PDF viewing application presented in an earlier contribution, *issue 105 page 27*, and change it so we can either search locally in the shown PDF or look in the collection of PDFs on the server.
We start by refactoring the application: initially, the application consisted of a single application class. This will now be separated in 4 different classes: `TPDFPanel` this class abstracts away the PDF viewer PDF.js. It has a method to load a PDF, a method to set the current page, and 2 events that are called when the PDF is loaded and the page is shown, respectively.

```
TPDFPanel = Class(TComponent)
  Procedure QueueRenderPage(aNum : Integer);
  Procedure ShowPDF(aSource : TPDFSource; AtPage : Integer = 1);
  Property CanvasID : String;
  Property PageRendering : Boolean;
  Property Scale : Double;
  Property DisplayedPage : Integer;
  Property PageCount : Integer;
  Property OnPageLoaded : TNotifyEvent;
  Property OnLoaded : TNotifyEvent;
  Property PDFDoc : TPDFDocumentProxy;
end;
```

The properties are pretty straightforward: The `CanvasID` can be set to the `ID` of the canvas element in which the PDF element is drawn. The `Scale` property is there to set the zoom factor for the PDF. The read-only properties `PageRendering`, `DisplayedPage`, `PageCount` and `PDFDoc` have obvious meanings.

`TPDFControlPanel` This class contains the controls for the PDF viewer: The page navigation buttons and the zoom button. The component has in fact only some public properties to set the PDF panel and the IDs of various HTML elements.

```
TPDFControlPanel = class(TComponent)
  Procedure BindElements;
  Property PDFPanel : TPDFPanel Read FPDFPanel Write SetPDFPanel;
  Property EditPageID        : string;
  Property PreviousButtonID  : string;
  Property ZoomLabelID       : string;
  // Properties for all other buttons
end;
```

The `BindElements` call will fetch the references to all needed elements in the HTML page.
`TPDFSearchControl` This class takes care of the actual search mechanism: it has some properties for the various IDs of the HTML tags that it needs to do its job, and some methods to show/hide/clear the search results panel .

```
TPDFSearchControl = class(TComponent)
  procedure BindElements;
  Procedure ShowResultPanel;
  Procedure HideResultPanel;
  Procedure ClearResultPanel;
  Property pdfPanel                : TPDFPanel;
  Property LocalCheckboxID   : String;
  Property SearchButtonID    : String;
  Property SearchEditID      : String;
  Property DivAutoCompleteID : String;
  Property ResultPanelID     : String;
  Property SidebarPanelID    : String;
  Property ShowResultsPanelButtonID;
end;
```

Again, the `BindElements` call will fetch the references to all needed elements in the HTML page. The code of `TPDFControlPanel` and `TPDFPanel` classes can be found in the previous article on showing a PDF, so it will not be repeated here. The only changed code is that the `TPDFControlPanel` class does not access the low-level `TPdfDoc` class, but instead uses the `TPDFPanel` methods (which in turn of course call the methods of the `TPdfDoc` class).
The `TPDFIndexApp` application class ties everything together: it has instances of the above three classes, and some references to the HTML elements to load a PDF file from disk:

```
TPDFIndexApp = class(TBrowserApplication)
  pnlSidebar,
  lblFileLocation : TJSHTMLElement;
  btnLoad,
  btnClosePane : TJSHTMLButtonElement;
  edtPDFFile: TJSHTMLInputElement;
  FPDFPanel : TPDFPanel;
  FPDFControls : TPDFControlPanel;
  FPDFSearch: TPDFSearchControl;
  procedure doRun; override;
  procedure BindElements;
  function DoLoadFile(Event: TEventListenerEvent): boolean;
  procedure onLoad(aEvent: TJSEvent);
  procedure ShowPDF(aSource: TPDFSource);
  Procedure DisplayFileLocation(const aLocation : String);
  procedure onClosePane(aEvent: TJSEvent);
end;
```

The various event handlers in the application class will not be treated here, the code has not changed from the code presented in the article on showing a PDF file. The `DoRun` code has changed, so we will present it here:

```
procedure TPDFIndexApp.doRun;
const
    TheURL = 'https://mozilla.github.io/pdf.js/build/pdf.worker.js';
begin
  pdfjsLib.GlobalWorkerOptions.workerSrc:=TheURL;
  FPDFPanel              :=TPDFPanel.Create(Self);
  FPDFControls           :=TPDFControlPanel.Create(Self);
  FPDFControls.PDFPanel  :=FPDFPanel;
  FPDFSearch             :=TPDFSearchControl.Create(Self);
  FPDFSearch.PDFPanel    :=FPDFPanel;
  BindElements;
  Terminate;
end;
```

As can be seen in the code, the code is reduced to creating the various panels and passing a reference to the `TPDFPanel` instance to the `TPDFControls` panel. The `BindElements` code is much reduced, since most of the elements have now been moved to the other classes. Their `BindElements` method is called from the application class `BindElements` method.

```
procedure TPDFIndexApp.BindElements;
begin
  btnLoad              :=TJSHTMLButtonElement(GetHTMLElement('btnLoad'));
  btnLoad.addEventListener('click', @onLoad);
  lblFileLocation      :=GetHTMLElement('lblFileLocation');
  edtPDFFile           :=TJSHTMLInputElement(GetHTMLElement('edtPDFFile'));
  edtPDFFile.onchange  :=@DoLoadFile;
  btnCloseSidebar      :=TJSHTMLButtonElement(GetHTMLElement('btnClosePane'));
  btnCloseSidebar.addEventListener('click', @onClosePane);
  pnlSidebar           :=TJSHTMLInputElement(GetHTMLElement('pnlSidebar'));
  FPDFPanel.CanvasID   :='PDFCanvas'; // Will call bindelements
  FPDFControls.BindElements;
  FPDFSearch.BindElements
end;
```

The other elements and events are related to loading a user-selected PDF file: the original functionality of being able to search any file is preserved. The `TPDFPanel` and `TPDFControlPanel` classes do not contain code that differs from the article on showing a PDF file, so their code will not be repeated here.
The `TPDFSearchControl` does change, since now 2 search mechanisms must be supported, as well as a mechanism to show an auto-complete list.We'll start with the auto-complete list. It is activated in the on input event of the edit element. The event handler is set in the `BindElements` method:

```
edtSearch           :=TJSHTMLInputElement(GetHTMLElement(SearchEditID));
edtSearch.onkeyup   :=@DoSearchKeyUp;
edtSearch.oninput   :=@DoCompleteWord;
```

The `DoCompleteWord` method is responsible for starting (*or restarting*) a timer (*200 ms*), and when the timer expires, the word completion list is fetched. Fetching the word completion list is done by opening a dataset:

```
function TPDFSearchControl.DoCompleteWord(Event: TEventListenerEvent): boolean;

    procedure DoServerSearchWord;
    begin
      if Length(edtSearch.Value)>1 then
      begin
        FSearchTerm:=edtSearch.Value;
        FWords.Close;
        FWords.Load([],Nil);
      end;
    end;

begin
  Result:=False;
  if FSearchTimerID<>0 then
    window.clearTimeout(FSearchTimerID);
  FSearchTimerID:=window.SetTimeout(@DoServerSearchWord,200);
end;
```

As mentioned in the first part of this article, the result of the server calls can be in a format that is suitable for a rest dataset. The `FWords` variable in the above code is a `TRestDataset`, and it is initialized in the `SetupDatasets` method of the `TPDFSearchPanel` class:

```
procedure TPDFSearchControl.SetupDatasets;
Const
    ServerURL = 'http://localhost:3010/';
begin
  FConn               :=TRESTConnection.Create(Self);
  FConn.BaseURL       :=ServerURL;
  FConn.OnGetURL      :=@DogetURL;
  FResult             :=TRestDataset.Create(Self);
  FResult.Connection  :=FConn;
  Fresult.AfterOpen   :=@DoOpenResults;
  FWords              :=TRestDataset.Create(Self);
  FWords.Connection   :=FConn;
  FWords.AfterOpen    :=@DoWordsOpen;
end;
```

The method also sets up the `FResult` dataset, which will handle the result of the search for word matches, later on.
The `FConn.OnGetURL` is an event called by the `TRestConnection` component, which can be used to construct the URL for a dataset that gets its data from a REST server. In our implementation, we need to choose the correct endpoint depending on what dataset is opened:

```
procedure TPDFSearchControl.DogetURL(Sender: TComponent; aRequest: TDataRequest;
var aURL: String);
var
  Q : String;
begin
  Q:=encodeURIComponent(FSearchTerm);
  if aRequest.Dataset=FResult then
    aURL:=FConn.BaseURL+'search?m=1&q='+q
  else
    aURL:=FConn.BaseURL+'list?t=contains&m=1&q='+q;
end;
```

You can see from the above that the 2 endpoints we defined in the server are used for both datasets. In both cases the query variables are set up to request metadata, and the q is set to the `FSearchTerm` variable, which is the value of the `edtSearch` HTML input tag.
The `AfterOpen` event of the `TDataset` class is used to generate the HTML for the word completion list. The HTML for this completion list is actually quite simple. The edit control is wrapped in 2 classes:

```
<p class="control is-small">
  <div class="dropdown">
    <div class="dropdown-trigger">
      <input id="edtSearch" class="input" style="max-width: 15em;">
      <div class="dropdown-menu" id="mnuAutoComplete" role="menu" />

    </div> <!-- .dropdown-trigger -->
  </div> <!-- .dropdown -->
</p>
```

And the html for the completion list is inserted below the tag with id `mnuAutoComplete`.
It is generated by looping over the records in the dataset and generating the necessary HTML elements. The routine starts by hiding the element, clearing the content and then adding the elements:

```
procedure TPDFSearchControl.DoWordsOpen(DataSet: TDataSet);
Var  S : String;
     F : TField;
     P : TJSHTMLELement;
     A : TJSHTMLAnchorElement;
begin
   mnuAutoComplete.style.setProperty('display','none');
   mnuAutoComplete.InnerHTML:='<div class="dropdown-content"></div>';
   P:=TJSHTMLELement(mnuAutoComplete.firstElementChild);
   if Dataset.RecordCount<=0 then exit;
   F:=Dataset.FieldByName('Word');
   While Not Dataset.EOF do
   begin
     S:=F.AsString;
     a:=TJSHTMLAnchorElement(Document.createElement('a'));
     a.href:='#';
     a.classlist.Add('dropdown-item');
     a.innerText:=s;
     a.dataset['value']:=s;
     a.addEventListener('click',@DoWordSelected);
     P.appendChild(a);
     Dataset.Next;
   end;
   mnuAutoComplete.style.setProperty('display','block');
end;
```

NOTE that the '`OnClick`' event is set for every word. At the end, the menu is made visible again. *Figure 3 on page 14* of this article shows what the effect is of this code. Needless to say, using some CSS the list can be made to look much nicer.
The `OnClick` event handler on the words sets the word in the search edit, which is a really simple operation:

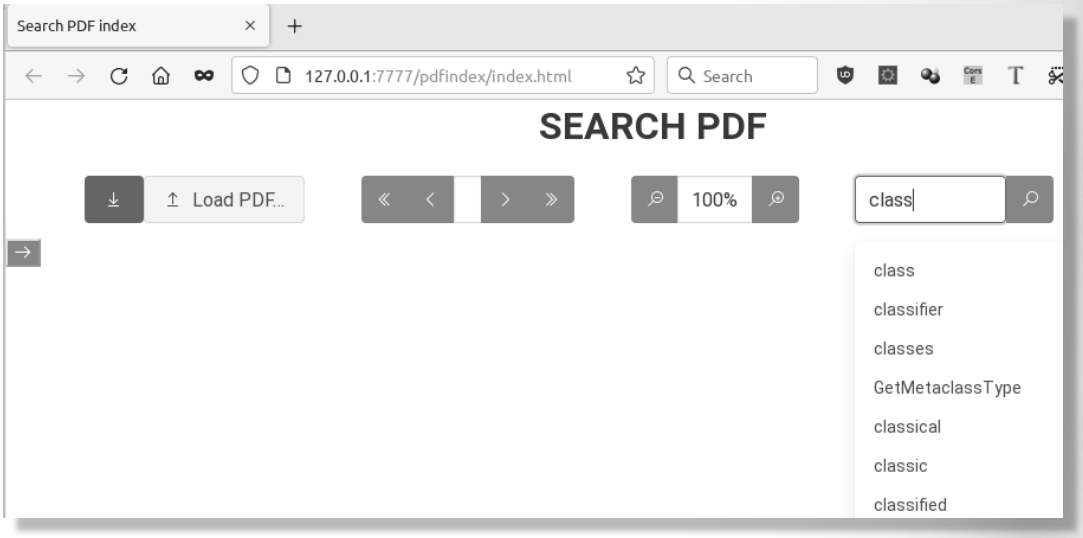


Figure 3: The autocomplete list

```
procedure TPDFSearchControl.DoWordSelected(Event: TJSEvent);
begin
  event.PreventDefault;
  edtSearch.value:=event.targetelement.innerText;
  mnuAutoComplete.style.setProperty('display','none');
end;
```

When the user presses Enter in the search edit, or presses the search button, the resulting action depends on the value of the `cbLocal` checkbox: when checked, a local search is performed. If unchecked, then the server is queried:

```
function TPDFSearchControl.DoSearchKeyUp(aEvent: TJSKeyBoardEvent): boolean;
begin
  Result:=False;
  if (aEvent.Key<>TJSKeyNames.Enter) then exit;
  onSearch(aEvent);
end;

procedure TPDFSearchControl.onSearch(aEvent: TJSEvent);
var
  aterm : string;
begin
  aTerm:=edtSearch.Value;
  if Length(aTerm)<=2 then exit;
  if cbLocal.Checked then
    begin
      if not assigned(pdfPanel) then exit;
      DoLocalSearch(aTerm);
    end
  else
    DoIndexSearch(aTerm);
end;
```

As you can see, only words of length 3 or longer will be searched. The `DoLocalSearch` is the search mechanism as implemented in the article on showing a PDF, and will not be repeated here.
The `DoIndexSearch` method is the method we are interested in, and it is very simple. It opens the `FResult` dataset:

```
procedure TPDFSearchControl.DoIndexSearch(aTerm : String);
begin
   FResult.Close;
   FSearchTerm:=aTerm;
   FResult.Load();
end;
```

Again, the `AfterOpen` event of the `FResult` dataset is where the real work is done. It is again a simple loop over the dataset. The `TServerResultsMap` is a little helper class, which contains a field definition for every field in the result set (*comparable to persistent fields, only created at runtime*). Every record from the dataset is stored in a record of type `TServerMatch` and passed on to a routine `ShowServerMatch`,which generates the HTML for the record.

```
procedure TPDFSearchControl.DoOpenResults(DataSet: TDataSet);
var
   i        : Integer;
   aResult  : TServerMatch;
   aMap     : TServerResultsMap;
   NoFilter : Boolean;
begin
   I:=0;
   ShowResultPanel;
   aMap:=TServerResultsMap.Create(Dataset);
   While not aMap.Dataset.EOF do
     begin
       Inc(I);
       aResult.FromMap(aMap);
       ShowServerMatch(aResult);
       Dataset.next;
     end;
end;
```

For completeness, here is the `TServerMatch` record:

```
TServerMatch = record
  ID       : Integer;
  Rank     : Integer;
  Page     : Integer;
  Issue    : String;
  Author   : String;
  Title    : string;
  URL      : String;
  Context  : String;
  Date     : TDateTime;
  Procedure FromMap(aMap : TServerResultsMap);
end;
```

The FromMap copies the values of all dataset fields into the typed record fields.
The `ShowResultMatch` routine uses the `TServerMatch` record to generate the HTML for a match. Basically, this routine is a set of search-and-replace operations on a HTML template, and the resulting HTML is inserted in a HTML element for which the `OnCLick` is then set.
The `ResultContent` constant contains the HTML template, it is not shown here, the interested reader can find it in the source code. It has variable placeholders in {{ }} brackets: the name in the brackets is a field name and is replaced by the value of the named field.

```
function TPDFSearchControl.ShowServerMatch(aResult: TServerMatch): TJSHTMLElement;
Var
  aReplace, Content, Res : String;
begin
  Result:=TJSHTMLElement(Document.CreateElement('a'));
  Result.Dataset['page']   :=IntToStr(aResult.Page);
  Result.Dataset['idx']    :=IntToStr(aResult.ID);
  Result.Dataset['title']  :=aResult.Title;
  Result.Dataset['url']    :=aResult.url;
  Result.ClassName         :='panel-block result-item';
  Res:=ResultContent;
  Res:=StringReplace(Res,'{{page}}',IntToStr(aResult.Page),[rfReplaceAll]);
  Res:=StringReplace(Res,'{{author}}',aResult.Author,[rfReplaceAll]);
  Res:=StringReplace(Res,'{{issue}}',aResult.Issue,[rfReplaceAll]);
  Res:=StringReplace(Res,'{{title}}',aResult.Title,[rfReplaceAll]);
  aReplace:=StringReplace(Highlight,'{{match}}',edtSearch.Value,[rfReplaceAll]);
  Content :=StringReplace(aResult.Context,edtSearch.Value,aReplace,[rfReplaceAll]);
  Res     :=StringReplace(Res,'{{content}}',Content,[]);
  Writeln('Res : ',Res);
  Result.InnerHTML:=Res;
  Result.AddEventListener('click',@DoServerMatchSelected);
  pnlResults.appendChild(Result);
end;
```

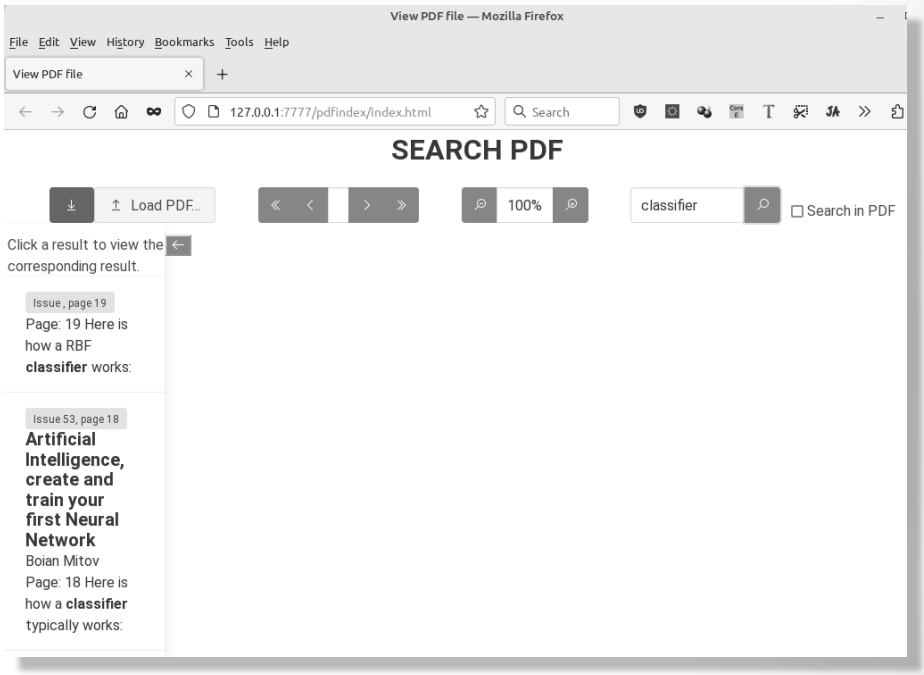


Figure 4: The search result

NOTE that the HTML element in which the generated HTML is inserted (*the ResultHTML element*) has several data attributes with all information needed to locate the article from which the match forms a part.
The result of this code is shown in *figure 4 on page `17 of this article*. The last piece of the puzzle is the `DoServerMatchSelected` event handler, which is invoked when the user selects a match in the result list.

It is actually a quite simple method. It uses the data-attributes of the generated HTML to select the PDF and page to show. If there is a valid PDF name and page number, the `ShowPDF` method of the `TPDFPanel` instance is used to show the correct page.

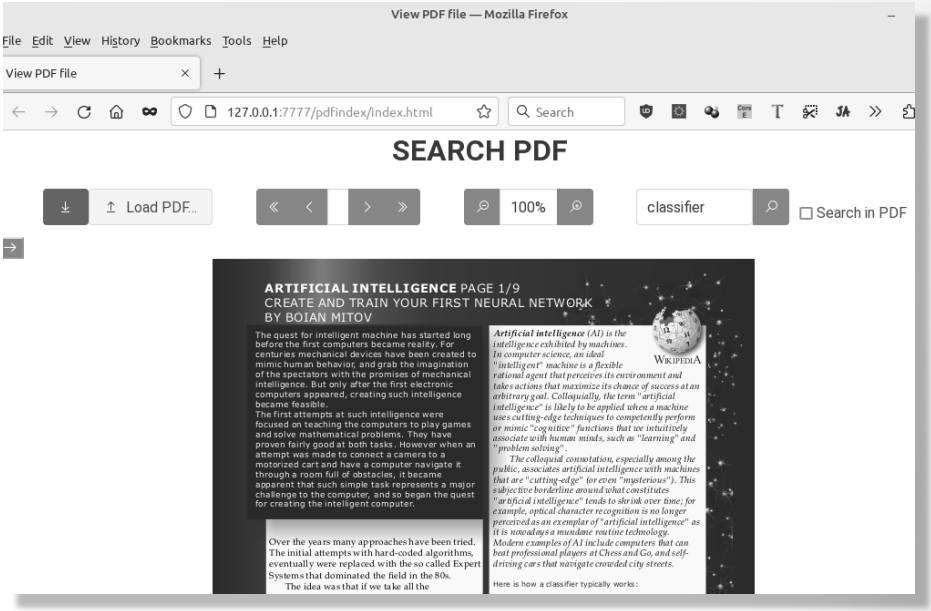


Figure 5: Selecting a result match.

```
procedure TPDFSearchControl.DoServerMatchSelected(aEvent: TJSEvent);
var
  aPage      : Integer;
  aTitle,aURL : string;
  Src        : TPDFSource;
  El         : TJSHTMLElement;
begin
  aEvent.currentTargetElement.classList.add('is-active');
  El     :=TJSHTMLElement(aEvent.currentTargetElement);
  aPage  :=StrToIntDef(String(El.Dataset['page']),-1);
  aTitle :=String(El).Dataset['title'];
  aUrl   :=El.Dataset['url'];
  if (aPage<>-1) and (aUrl<>'') then
    begin
      Src     :=TPDFSource.new;
      Src.url :=ServerURL+'pdf/'+aUrl;
      pdfPanel.ShowPDF(Src,aPage);
      HideResultPanel;
    end;
end;
```

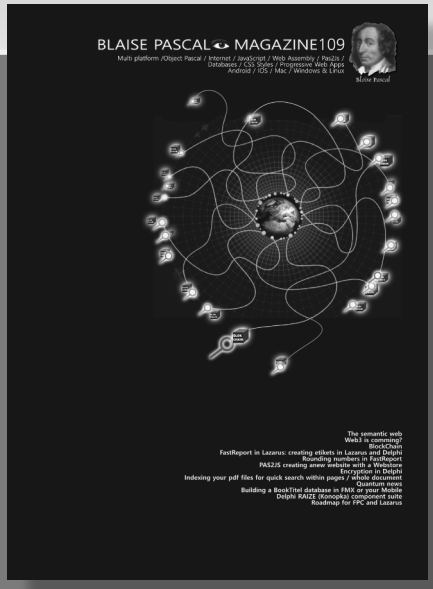
As a last step, the results panel is closed to maximize the space for the PDF viewer. The user can open the results panel with a button, and select another match, if so desired. The result can be seen in *figure 5 on page 17*

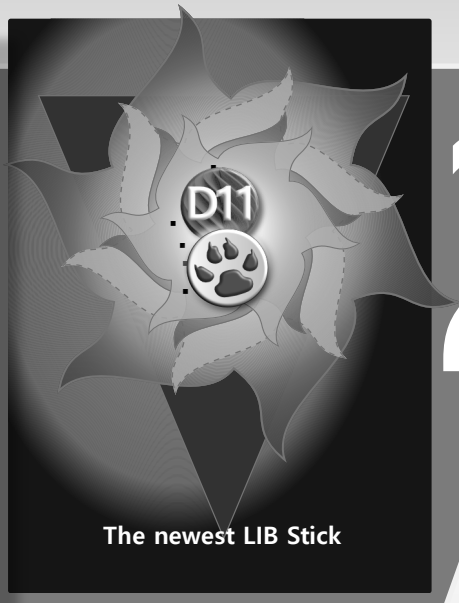
6 CONCLUSION

In this second article on PDF indexing, we showed how to use the PDF index database constructed in the previous article on PDF indexing.
All this was accomplished with standard components of Free Pascal and Pas2JS.

Although the resulting mechanism works satisfactorily and does what is needed, there is room for improvement.
For example the search mechanism can be switched to **manticore** search (*which has more options than the FPC mechanism*).
Additionally the client code can be reduced by using some of the data-aware widgets distributed by Pas2JS instead of the custom code written here. These improvements will be the subject of a future contribution.

1





2

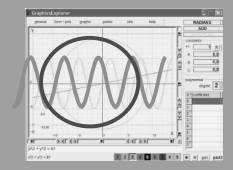
LAZARUS HANDBOOK

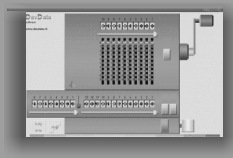
3

4

LEARN TO PROGRAM USING LAZARUS
HOWARD PAGE-CLARK

DAVID DIRKSE
including 50 example projects





BLAISE PASCAL MAGAZINE

COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

INCLUDING THE NEW LAZARUS KIT PDF INDEXER

1. One year Subscription
2. The newest LIB Stick
   - All issues 1- 109
   - On Credit Card
3. Lazarus Handbook
   - PDF including Code
4. Book Learn To Program
   - using Lazarus PDF including 19 lessons and projects
5. Book Computer Graphics Math & Games
   - PDF including ±50 projects

SPRING SUPER PACK 2023

PRICE € 150

NORMAL PRICE € 275

Delphi 11.3

DEVELOPER DAG

Delphi Developer Dag

Op 8 juni a.s. organiseren wij een informatieve dag voor Delphi ontwikkelaars. In één dag laten wij u zien hoe er optimaal gebruik gemaakt kan worden van Delphi en inspireren wij u om nieuwe technieken toe te passen in uw eigen ontwikkelomgeving.

Bevlogen sprekers

We hebben vertrouwde en nieuwe sprekers uitgenodigd waar onder Antonio Zapater – de nieuwe Technical Lead van Embarcadero. Hij zal een kick-off geven over de nieuwste versie én een hands-on presentatie met behulp van een zelfontwikkelde app. Ook zijn MVP's Bob Swart, Danny Wind en Marco Geuze van de partij. Aan de hand van praktijkvoorbeelden laten zij u de mogelijkheden en de kansen zien van het ontwikkelen in Delphi.

8 juni a.s.

We zijn bijna rond met een geschikte locatie die goed per auto en OV bereikbaar is. Zodra deze lokatie definitief is zullen wij een mailing uitsturen met alle details en de mogelijkheid om u in te schrijven. Maar donderdag 8 juni wordt de datum dus wilt u deze dag alvast noteren in uw agenda?

Wij zien er naar uit om u binnenkort weer te ontmoeten! Het Barnsten team

- ✓ Ontwerp op high-DPI 4k+ schermen
- ✓ Bekijk VCL Styles in ontwerptijd
- ✓ Nieuwe & gemoderniseerde VCL-componenten
- ✓ Verbeterde ondersteuning voor extern bureaublad
- ✓ Enkele broncode, elk platform
- ✓ Doel Windows 11, iOS 16, Linux, macOS Ventura, Android 13
- ✓ Compileren voor Android API 32
- ✓ Compileer Delphi-apps voor macOS M-serie (Apple Silicon) processors

Save the date: 8 juni

embarcadero

barnsten

Update for FastReport

Creating a single ecosystem of report generators for Delphi in the **FastReport 2023.2** release:

One installation system with online authorization — install and update all your products at once.

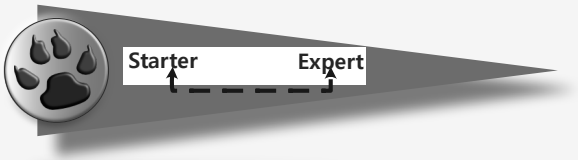
Common release system — major releases are published as a general.

Shared library for all products — fixes and new functionality can be available in multiple products at once.

Convenient to work with all products of the Delphi family already from the current release!

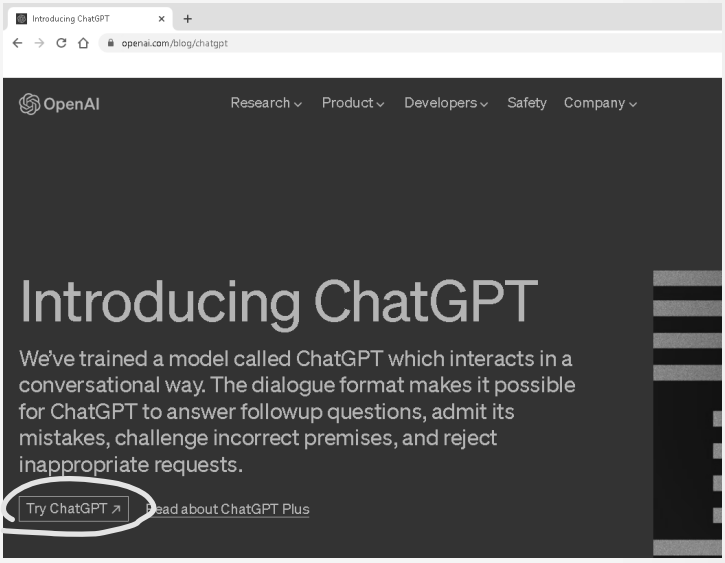
> **Also:**
>
> Improved compatibility with the current version of the IDE, HiDPI support for FastCube, and improved work with styles.

Upgrade to version 2023.2 or Download a free demo here:
www.fast-report.com

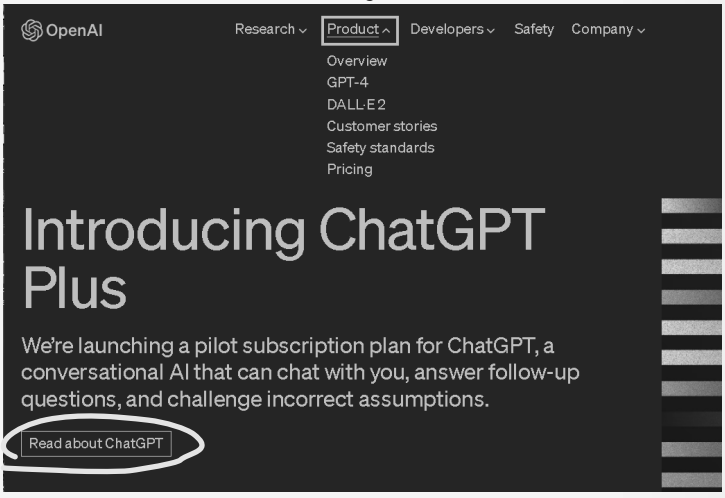


ABSTRACT:

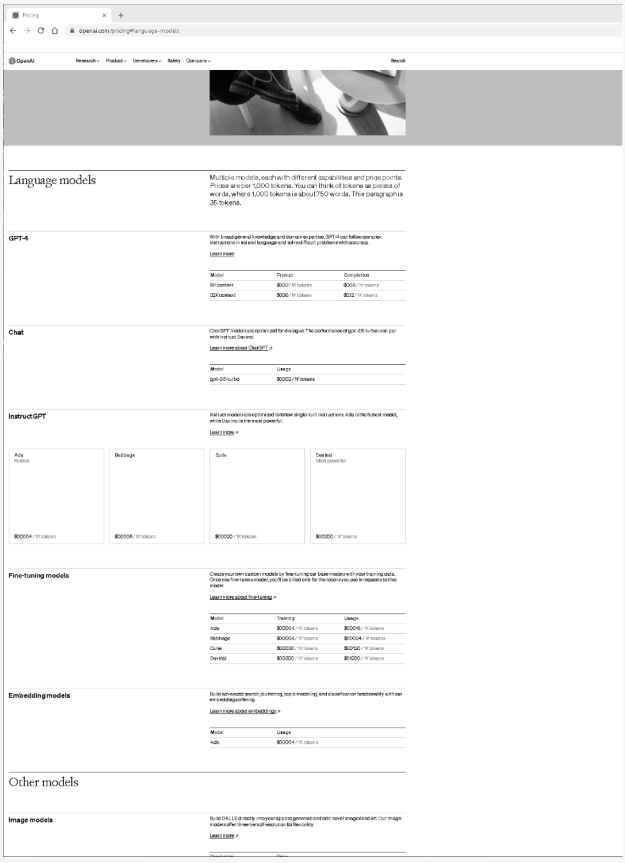
This is a small example of using code inside Fastreport
I was annoyed because of a calculation that always came up with Euro xxx,01 cent at the end and I wanted to solve that fore once and for all.I lost a lot of time solving it.
Since we now have ChatGPT I thought it might be a nice question for it. ChatGPT can simply be found on the internet and you and the address is here:
`https://openai.com/blog/chatgpt`



Use the button with the text Try ChatGPT



PRICING IS HERE:
`https://openai.com/pricing`



I used the trial version. and started the following request:

Can you explain how the round factor in Fastreport has to be handled?

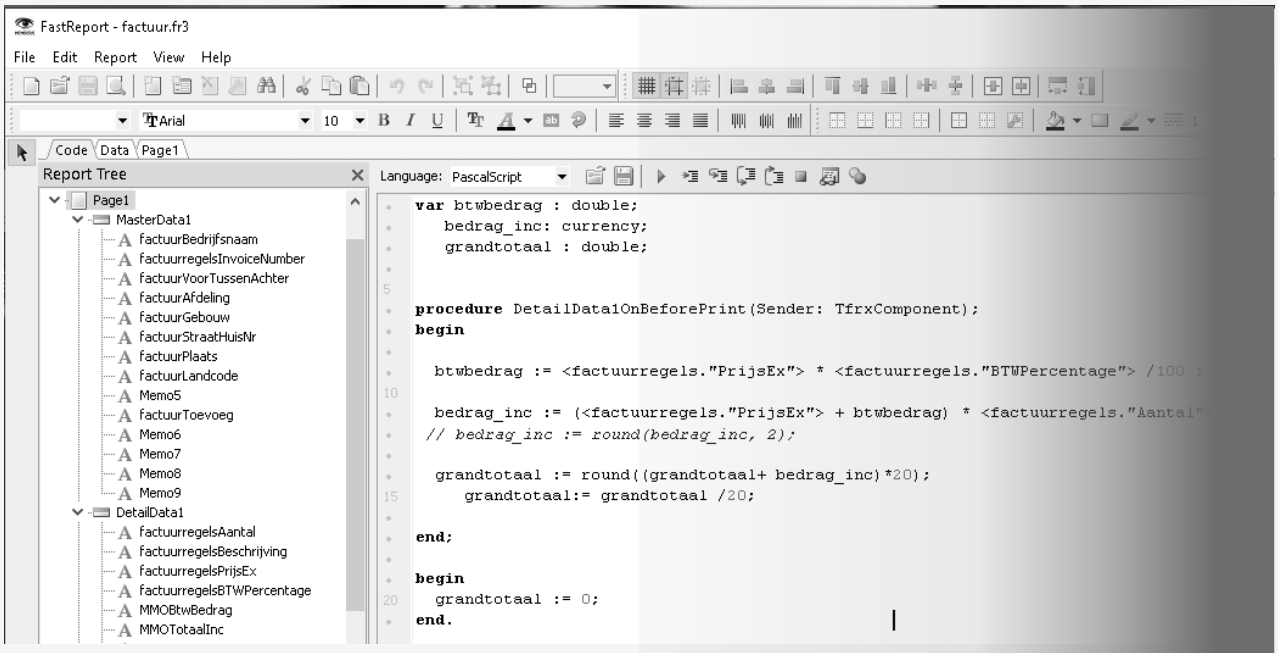
Sure,
the Round function in FastReport is used to round a numeric value to the nearest integer or a specified number of decimal places. The function accepts two arguments:
the first argument is the number to be rounded, and the second argument specifies the number of decimal places to which the number should be rounded.

Here's an example of how to use the Round function in FastReport:

Suppose we have a numeric field named "Price" that contains a value of 12.3456. We want to display the price rounded to two decimal places. Here's how we would use the Round function in the expression:

Round([Price], 2)

The above expression will return the value 12.35, which is the original value rounded to two decimal places.



The above expression will return the value 12.35, which is the original value rounded to two decimal places.

In summary, the Round function is used to round numeric values to the nearest integer or specified number of decimal places in FastReport.

To use the round function in FastReport, you can follow the below steps:

1. Open the report in the report designer.
2. Select the text object that you want to apply the round function to.
3. Go to the Properties window and find the "Text" property.
4. In the text property, you can use the following syntax to apply the round function:

```
=Round(<<your expression>>, <<number of decimal places>>)
```

For example, if you want to round a numeric field called "Price" to two decimal places, you would use the following expression:

=Round(Fields.Price, 2)

This will display the rounded value of the price field in the text object.

Note that the round function can be used with any numeric expression or field in your report.

So I answered that it was incorrect, because it doesn't accept extra arguments

ChatGPT continues

I apologize for any confusion or misinformation earlier. I did not have access to a FastReport environment to test my previous answers. Upon further research and testing, I have confirmed that continues in FastReport, the Round() function does not support specifying the number of decimal places to round to. Instead, it rounds to the nearest integer.

I apologize for any inconvenience or confusion that my previous responses may have caused. If you have any further questions or concerns, please let me know and I will do my best to assist you.

Funny but nothing but a repetition.

I stopped and started to find other solutions:
In the report you have to open the code page where you create your calculations

```
var btwbedrag   : double;    // VAT amount
    bedrag_inc  : currency;  // Vat including amount
    grandtotaal : double;    // Grand Total


procedure DetailData1OnBeforePrint(Sender: TfrxComponent);
begin

 btwbedrag := <factuurregels."PrijsEx"> * <factuurregels."BTWPercentage"> /100 ;

 bedrag_inc := (<factuurregels."PrijsEx"> + btwbedrag) * <factuurregels."Aantal">;

// bedrag_inc := round(bedrag_inc, 2);  // ? this is not accepted by fastreport
```

The working answer is like this:

First round the number and multiply that by a number: 20
in this case I want to round the 70,01 to 0 and if it would have been a 70,05 up to 70,10.
After this you'll have to divide the number by 20.
Be carefully because in some countries this is not legally allowed.

```
   grandtotaal  := round((grandtotaal+ bedrag_inc)*20);
   grandtotaal  := grandtotaal /20;

end;

begin
 grandtotaal := 0;
end.
```

EXAMPLE IN LAZARUS

```
uses math;
  1. ...
  2. function round2(const Number: extended; const Places: longint): extended;
  3. var t: extended;
  4. begin
  5.   t := power(10, places);
  6.   round2 := round(Number*t)/t;
  7. end;
```

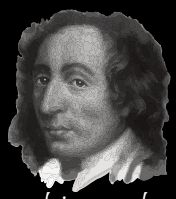


If you are from Ukrainian origin you can get a free Subscription for Blaise Pascal Magazine,
we will also give you a free pdf version of the Lazarus Handbook.
You need to send us your Ukrainian Name and Ukrainian email address (*that still works for you*), so that it proofs you are real Ukrainian.

please send it to editor@blaisepascal.eu and you will receive your book and subscription

BLAISE PASCAL MAGAZINE107/108

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux


Blaise Pascal


COMPONENTS 4 DEVELOPERS

Donate for Ukraine and get a free license at:
https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

COMPONENTS 4 DEVELOPERS



Donate for Ukraine and get a free license at:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/
(Just click)

kbmMW Professional and Enterprise Edition v. 5.21.00 Studio XE5 to 11 just now released! kbmMemTable v. 7.97.00 Standard and Professional Edition released

This is a significant new release with new high performance transports, **OpenSSL v3 support,** WebSocket support, further improvements to SmartBind, new high performance hashing algorithms, improved RemoteDesktop sample and much more.

This release requires the use of **kbmMemTable** v. 7.96.00 or newer.

- RAD Alexandria supported
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OS X client and server support
- Native high performance 100% developer defined application server
- Full support for centralised and distributed load balancing and fail-over
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multi thread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronounceable password generators.
- High performance LZ4 and J peg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, J SON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- Easily supports large datasets with millions of records
- Easy data streaming support
- Optional to use native SQL engine
- Supports nested transactions and undo
- Native and fast build in M/D, aggregation/grouping range selection features
- Advanced indexing features for extreme performance

- New: full Web-socket support.
  The next release of kbmMW Enterprise Edition will include several new things and improvements.
  One of them is full Web-socket support.
- New I18N context sensitive internationalisation framework to make your applications multilingual.
- New ORM LINQ support for Delete and Update.
  Comments support in YAML.
- New StreamSec TLS v4 support (by StreamSec)
  Many other feature improvements and fixes.

Please visit http://www.components4developers.com
for more information about kbmMW

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

COMPONENTS 4 DEVELOPERS

D11