flossUK

news@UK

The newsletter of FLOSS UK, the new name for the UK's oldest Open Systems User Group, UKUUG

Published electronically at `http://www.flossuk.org/Newsletter`

Volume 21, Number 4 **ISSN 0965-9412** **December 2012**

Contents

From the Secretariat

Jane Morrison

The minutes from the UKUUG Ltd Annual General Meeting held on 20th September have once again not been circulated to members via hard copy. All members were emailed on 4th October and advised that the minutes and associated documents could be found on the web site at: **http://www.flossuk.org/agm2012**.

Of course any one who wants a printed copy should contact me here at the office.

Current Council members are: Kimball Johnson, Paul Waring, Holger Kraus, Martin Houston, Ian Norton, Stephen Quinney, Jon Dowland and Quentin Wright.

Kimball Johnson has taken over the role of UKUUG Chairman and Paul Waring has been appointed as Treasurer.

We thank retired Council members, Howard Thomson, John Pinner and Phil Hands for their past commitment.

We are currently putting in place a full schedule of events for 2013 – to date the following have been agreed:

- Intermediate Python Tutorial – by John Pinner (re-scheduled from 7th November) now on 30th January 2013 – see flyer inserted with this Newsletter
- Perl Tutorial by Dave Cross: 2 day Intermediate (including practical sessions) – 12th and 13th February
- Perl Tutorial by Dave Cross: 2 day Advanced (including practical sessions) – 14th and 15th February – London – see flyer inserted with this Newsletter
- Spring 2013 Tutorial and Conference – 19th, 20th and 21st March 2013 – Newcastle-upon-Tyne – see insert with this Newsletter

We are also planning more tutorials for 2013: if you have any particular topics that you think would be of interest please let us know.

I would like to remind you of a couple of benefits we have in place for members – we have a page on the web site **http://www.flossuk.org/Consultants** which lists members who can provide consultancy services. This is free to members – take a look and send me your details if you are a consultant and get some advertising for free.

Also, I would like to remind you that in addition to the Reilly book discounts (40% online and 37.5% via the Secretariat) we also provide other discounts for members these include: 2nd Quadrant (10% off PostgreSQL courses), GBDirect (10% off various courses), Bytemark Hosting (10% discount on all virtual machines) and Opengear (10% discount across a full range of products).

Take a look at **http://www.flossuk.org/OtherDiscounts**.

The annual membership subscription invoices will be sent out in January, please look out for your invoice and as always prompt payment will be gratefully received!

We are grateful to all our sponsors including our Gold Sponsor member, SUSE.

I should like to take this opportunity to wish you all a very Happy Christmas and a Peaceful New Year.

The Secretariat will close on Friday 14th December and re-open on Wednesday January 2nd 2013.

Please note the copy date for the next issue of the Newsletter (March 2013) is 15th February.

We are always looking for interesting submissions from members, so if you have any news, articles etc. please send copy direct to **newsletter@ukuug.org**.

If you do NOT wish to receive future issues of the Newsletter in hard copy (all issues can be found on our web site in pdf format) please let me know.

Chairman's report

Kimball Johnson

Introduction

Firstly I wish to introduce myself, and thank Council for electing me into the position of Chairman. I have been a member of FLOSS UK for a number of years, attending many conference and tutorials. I was elected to Council in September 2010 and since then have enjoyed getting involved in the running, including taking the lead of running the last Spring Conference in Edinburgh.

Unconference

The unconference was recently held at the BCS complex in Covent Garden. Although not a huge turnout, there was a wonderful variety of talk, from Open Source tractors to Home Automation. Thank you to everyone who came, and to BCS OSSG and Bytemark for their support in running the event.

Looking Forward

As I take on the role of Chairman, various people have asked me where I want to take the group. In my time in FLOSS UK I have seen a little dwindle of numbers at Conferences, and not many new faces. I hope to see this change, by widening the scope of talks further. The change of trading name to FLOSS UK was a good start of this as it helps dispel the myth that we are just about Large Unix Installations. With this in mind we are hoping to expand our conference program to include talks on programming, open hardware such as Arduino, and also provide tracks more aimed at beginners in all areas. Don't worry if you love the existing conference, it is not going to go away! Recent developments like the Raspberry Pi have caused a massive influx in new users of Linux and Free Software, and I believe this is an unmissable opportunity to expand our membership and support the community.

Community Support

This is still just an idea that the council are working on, but we want to do more to support community groups. FLOSS UK as a company can provide administrative support and financial guarantees that are useful for smaller groups attempting to put on a conference, as well as promotion to our members. In addition to this we are discussing the possibility of small grants to worthy projects, in return for write ups in the newsletter. This idea is still under discussion, so we will give more detail when we have it.

Damian Conway and the Dao of Presenting

Mark Keating

In our lives a good number of us are faced with the situation where we must stand in front of an audience of peers, colleagues, friends or strangers and deliver a speech. Sometimes it is with slides, sometimes it is not. The speeches can be of varying length and depth, but they will all share a common feature. You will invariably feel you could have done better at it.

Then you see those people who are masters at it, they make that gut-wrenchingly nightmarish task look like a simple event in which they comfortably engage you. Some of them, like Damian Conway, do it and seem eloquent, witty and urbane at the same time.

Generally we fill ourselves with a mixture of envy, hatred and awe of these people. Life is unfair, why are some people born with the ability to make great presentations and make them look so effortless? How is it that this random Antipodean maniac with his lust for knowledge and Tae Kwan Do can do this? Well the truth is that life isn't unfair, these people are masters because they put in the effort to be that good.

But how do they do it? How can you possibly do the same? I had the great privilege to be at UKUUG's and Damian Conway's Presentation Skills Conference held in October 2012 where all was revealed.

Methods and Techniques

The first thing you will note about a skills event with Damian Conway is the ease he puts you at, and the second thing is that you will be surprised. It is not an ordinary day when you are calmly rendered incapable and in such an utterly charming, and of course disarming fashion. By incapable, I mean quite literally (and not figuratively) unwilling to want to move, but that was what happened to me.

Damian delights in entertaining his audience, he desires to engage with them and he is a master and informing. He is also a man full of skill and trickery and like a magician confounding the audience into stunned silence with smoke, mirrors and illusion, Damian astounds his with competence and knowledge.

So how did he get to be so good?

Do you recall the scene in the original Karate Kid, Ralph Macchio is taught by the wisened Pat Morito to do Karate, but the first lessons are to clean a car (wax on, wax off) and paint a fence (brush up, brush down). The master of the arts shows the exact technique that must be

used and the manner in which to do it. Later we learn these techniques are the pure basics of the principle blocks and strikes in Karate.

What Ralph learns in this film is that a great deal of effort, sometimes repetitive, placed in beforehand to the correct application of method, will provide you with the formative blocks that when assembled will free you for the creativity and inspiration to do exceptional things. He finds his inner self and perfects his own kick that eventual wins the day.

What Damian teaches in this presentation are the many skills that he has perfected over numerous, and I don't want to age him prematurely, decades of professional circuit teaching, presenting and training. Damian is our grand master and we are his pupils, if we follow the techniques, no matter how onerous and lengthy we will progress our proficiency and develop our own styles.

The whole essence of what Damian instills, this isn't natural, this is unnatural, in his own words 'standing in front of a group of people who are smiling at you is not wise, any other animal would take that to mean I am about to be eaten'. Yet this is what we are attempting to achieve.

The Karate Kid, fantasy that it is, also shows only a few techniques, the barest hint of enough for our montage from neophyte to ninja. Damian has eight hours and he uses every last minute of it. This is a smorgasbord of techniques that covers every element from choice of topic, writing, editing, preparation of slides, choice of fonts, colours, imagery, transition/animation techniques, to the actual use of voice, posture, body language, memory, and then beyond into learning, practicing and the actual presenting itself including engagement with the audience. If Damian didn't provide such wonderfully prepared condensed notes you'd have to pay him to present this three or four times more.

This is why Damian is so good. He has created a series of steps, lessons, the foundation blocks of all good presentations. He takes them apart and shows you those pieces one by one, we examine them to understand them, then we assemble and give ourselves the foundations from which we can create. What Damian has created is the presentation version Lego™Mindstroms©; learn to use it well and understand how each piece works and you can build amazing things.

Depth and Breadth

But this doesn't yet cover all you are taught. Damian has also spent a great deal of time studying this area, he has learned not just the techniques to being a good presenter, and the practices to present well, but why these techniques work. Part of the genius I experienced was Damian's breadth of understanding and depth of knowledge not only of his presentation subject but the secondary and tertiary thinking that is related to it.

We learned the psychological effects, discussed the functions of memory, perception, we also examined recent theories that helped Damian gain an understanding into why his techniques worked and how they could be refined. This felt like the deep inner magic of the form, where in Eastern mythology we pass from the practice of an art into the understanding of the way. This is where we have the Dao of Presenting, the core understanding and

how it resonates.

Value

I would love to say that this day is value for money, but that actually undersells it. Damian is almost worth paying for just to hear him talk on this subject, the fact that you are taught at the same time feels like a bonus. This is a masterclass and you are taught by a master. The lessons you will learn work in practice, I can attest to this, I have tried them and they work.

Damian will show you all the basics you will need, he will guide you on the path but he will not walk down it for you. The true genius of this course is not just that you are shown how to do these things, but you are encouraged to find your own way into doing them.

I want to thank the UKUUG for making the day possible, and thank Damian for giving me what I will call the Lego™Presentation Kit, one day I will build a ninja robot that will make him proud.

- Level - Any
- Content - Extensive coverage
- Materials - Excellent
- Provisions - Tea++, Coffee+, Food acceptable
- Venue - Suitable (Wifi, provisions, toilets, seats all acceptable to excellent)

A Regular Day Out in London

Jess Robinson

Regular expressions, like all embedded languages, can be lightly or heavily used, They are an entire programming language in their own right. It's worth understanding well how they work and just what they can be used for. Damian's course explains all in a way that should teach something to almost everyone. Complete beginners who have never seen nor tried to use a regex before would do well to do some basic reading before hand, this impression may be due to the fact that most people on the course seemed to be aware of basic regexes in the flashcard check.

The most interesting part of the course for me was the description of how the engine actually works. This was not a trip through the internals, but a visual and analogical[1] representation of how the individual instructions are carried out. The engine tries to map a route through the text it is matching against using each instruction in turn. There is (sadly) no handwavy magic[2], in fact it is quite stupid, it is up to the programmer to make it efficient.

Imagine writing your own navigation code (or a really dumb satnav[3]), without having any details about the roads available until you arrive at them (and ignoring signs, for the purpose of this exercise). You know which direction you want to go, to get from your starting position to your destination, so you set out along the nearest drivable road. The road turns out to be a dead-end, so you turn back and mentally mark that road as not useful for getting from A to B, then you try the next one.

The regex engine works much like this, it will completely run any instruction you give it, even if it seems obvious that it is not going to work, it has no knowledge of the future, and often even forgets the past. To be more concrete, suppose we want to find a number followed by a full stop (period) in a string of numbers. We could write a simple (dumb) regex to say "get me some digits followed by a dot" like this:

```
$str =~ /"(\d\.)"/;
```

We feed in a string that contains several numbers:

```
$str = 'ab432 126 573.89';
```

The engine will step through the input string, looking for a digit character, and note the position it was found (start of current matching attempt). It will then collect more digits until the next character is not a digit, then try the next instruction, the literal ".". If this fails, it backtracks to the marked position, disgards that character and tries again.

The run above will look like:

```
'a' is not a digit
'b' is not a digit
'4' is a digit, hooray! <- mark this spot and collect '4'
'3' is a digit, hang on to it
'2' is also a digit, hang on to it
' ' is not a digit, try next instruction
' ' is not a dot, bummer
undo to where we last marked
ignore '4' we know it doesnt start there
'3' is a digit, hooray! <- mark this spot and collect '3'
'2' is a digit, hang on to it
' ' is not a digit, try next instruction
' ' is not a dot, bummer
....
```

and so on, character for character, remembering nothing about what it already tried.. To illustrate this in a much less tedious way (it even speeds up over the boring bits), try Damian's new `Regexp::Debugger` module, which he used to demonstrate this process to us.

During the description of how the engine works, I was reminded of `Parse::RecDescent`, which works in much the same way. (Why I understood that before the Regular Expression Engine... who knows!)

For an idea of what else we learned, look at the source code of Regexp::Debugger, which despite being a module for parsing regexes, using regexes, is quite readable.

All in all, I was reminded of SQL **`http://enwp.org/SQL`**, which is another language primarily for embedding. Some folks just use it to fetch a bunch of data, and then manipulate that in Perl, and some use it to make the database do all the work... it's your choice, but these things are purpose built for a reason...

Course Summary:

- Level – intermediate
- Content – Full coverage
- Materials – Excellent

- Provisions – Tea++, Food so-so
- Venue – Suitable (Wifi, provisiions, seats and everything, easily reached)

[1] Huh, that word actually exists...

[2] This is probably a pedagogical fallacy to help you design better regular expressions, it is still magic, its just not as smart as you think.

[3] Satellite Navigation, or GPS navigation, we upgraded ours recently and it got somewhat less dumb...

FLOSS UK Regular Expressions Tutorial with Damian Conway

Charlie Harvey

I've just returned from a fantastic Understanding Regular Expressions tutorial given by Damian Conway for FLOSS UK (formerly the UK UNIX Users Group). Damian is a bit of a legend in the Perl community having won the Larry Wall award 3 times, written Perl Best Practices and spent quite a few years working on Perl 6. He is also a thoroughly engaging and very entertaining presenter.

We started the day reviewing how the regular expression engine actually works. Now, I've been using regexen for a fair few years now, but I found Damian's explanation that the regex engine is a mini-language within Perl a much more powerful way to reason about what actually occurs when running a regular expression than the way I've been accustomed to conceptually modelling it – as describing the parts of a string I want to match. We looked at a number of examples of regexen and broke them down into state ("railway") diagrams which help immensely in deconstructing their behaviour. We then looked at the railway diagrams from another angle – as trees. Search trees, of which we do a depth-first ordered traversal. The sort of thing that is the bread and butter of a jobbing computer scientist and suddenly makes regexps seem very straightforward indeed.

Damian talked a little about the design choices in the engine, coming out with what for me was his quote of the day, Perl would rather be fast and wrong that slow and correct. He talked about the design choice to shy away from optimisations to the engine that could be exponentially slow in some cases, in favour of an engine that was slower in many cases but not disastrously slow in unusual cases.

Part of the finite state machine/DSL/tree explanation of Perl's regexes was "pedagogical facilitation" (white lies) – a simplification to help us reason about our regex, the real situation is a little more complicated, given that our "tree" can contain various branching and looping structures.

We also looked at predefined subpatterns (`\w\W\d\D`), and also examined some more interesting post Perl 5.10 ones such as `\h` and the "plausible EOL sequence" `\R` – which is shorthand for "`\x0D\x0A?|[\x0A-\x0C\x85\x{2028}\x{2029}]`" and preferable for humans!

Damian introduced us to his incredible `Regexp::Debugger` module, which he made frequent use of in his explanations. I've not encountered the module before, but I have to

say I was very excited when I considered the hours of time it might have saved me. You should check it out.

After this overview we moved on to some specifics. Capturing, backreferences and interpolation firstly, then we looked at the new smart matching operator `~~` before moving on to substitution, alternative delimiters (hint `{}` is probably best) and optimisation with `qr//` and `/o`. We spent a good amount of time looking at ways to optimise backtracking, the demo code went from having to perform 1109 operations to performing 80-something – an order of magnitude performance increase gained with very little complexity overhead, simply adding `(?>)` round our match to avoid unnecessary backtracking and `^` used when we can.

The later parts of the day, we looked in detail at some of the newer and more advanced features available to Perl programmers including deferred subpatterns, named captures and named subpatterns. These Damian brought elegantly back to be relevant to his central theme of regexen as a language within the language Perl tying them back to conceptual territory that is familiar to programmers and not a mystery wrapped in an enigma wrapped in line-noise.

Reflections and insights gained

I gained a new way of looking at regexen that I'll definitely be using in my future Perl hacking. Some specific lessons that I drew from the day were

- `Regexp::Debugger` is seriously cool. Again quoting Damian, "the worst thing about writing `Regexp::Debugger` was not having `Regexp::Debugger` to debug it with".
- Using `/x` is a good idea.
- Named captures and named subpatterns rule if you're trying to build a complex recursive regex.
- Regexen are just another programming language, once you add in the branching constructs, and programming them can feel just like "normal programming".
- The regex engine is both more stupid and more clever than I had given it credit for.

Thanks to FLOSSUK for organising the day and to Damian for a thoroughly enlightening and very enjoyable tutorial.

This article is reproduced by kind permission of the author. The original (together with a colour screenshot of `Regexp::Debugger` *in use) can be seen at:*
http://charlieharvey.org.uk/page/flossuk_workshop_2012_10_11_perl_regexps

Unconference 2012

Roger Whittaker

This year's unconference was held on Saturday 27th October at the BCS premises in London. This was the third such event that we have held, and it was an enjoyable and stimulating day.

There were a good number of delegates who had something to talk about, and the spontaneous filling up of the programme worked out well. Talks ranged from the amusing and whimsical (Carles Pina's talk on "Olfactory Notifications") through more the practical but still hilarious home automation talk by Steve Godwin to serious discussions of open source technologies in broadcasting, and cutting edge discussions of the latest in databases, filesystems and scripting.

Phil Downer's described a DVB card capable of capturing and recording all free-to-air TV channels at once, while Kieran Kunhya described the current state of FLOSS in broadcasting more generally.

David Jones from ScraperWiki described CoffeeScript and the ways in which he is using it.

Gordon Banner gave two talks: one on the current state of "NoSQL" databases, and one on the current state of Perl. Dave Langley described MogileFS.

Alan O'Donohoe gave an interesting and inspiring talk on ways in which computing education in schools could be restored to a sensible state: this led to a great deal of discussion, as did John Pinner's somewhat deliberately contrarian anti-Linux rant.

The BCS premises were a pleasant place to hold this event, and there was a good lunch.

The turnout was reasonable but not huge: this could possibly have been remedied by better publicity, but also maybe by having a keynote speaker announced in advance as was done at the Birmingham unconference when Simon Phipps opened the event.

I would also recommend using a slightly more high tech scheduling system than a single whiteboard – I have seen how this can be helpful at other events run on this principle.

Overall, this was a very enjoyable event, and I think the principle of running an annual unconference in addition to the main events has now proved itself and I hope will continue into the future.

PostgreSQL: Up and Running

Regina Obe and Leo Hsu
O'Reilly Media
ISBN: 978-1-4493-2633-3
168pp.
£ 15.50
Published: July 2012

reviewed by Paul Waring

One of the things which has put me off using PostgreSQL in the past has been the lack of up to date introductory documentation, preferably in dead tree format. The website has excellent reference documentation, but this is only useful if you already know what you are looking for. This 'up and running' book promises a concise overview to quickly get your head around PostgreSQL's unique features if you already have experience with databases in general.

The first chapter covers the absolute basics, such as where to get PostgreSQL (thankfully without a 'step by step install' filler) and the three basic admin tools: command line, GUI and web client. The list of new features in the 9.x series is comprehensive and should prompt administrators to consider upgrading from 8.x, as well as giving users of other database products reasons to look at a possible migration. The next chapter continues on the topic of basic administration, including creating users and groups and assigning permissions. This process may seem long-winded at first, particularly if you are used to MySQL, but the authors provide a better explanation than I've seen in most other places.

Chapters three and four return to the basic administrative tools mentioned earlier, but in more detail. This is followed by a chapter devoted to data types – a sensible choice given how many there are in PostgreSQL. As well as the usual numeric, character and date/time types, PostgreSQL supports arrays, XML and any custom type which you want to create.

The sixth chapter covers constraints and indexes, probably two of the most important topics given how crucial they are, but also how often they are used inappropriately. Even a whole chapter doesn't do this topic justice, given the multiple types of indexes which PostgreSQL supports, but the authors have provided enough information for the reader to gain a broad understanding, from which you can fill in the gaps using the online documentation. The next chapter covers some SQL constructs which are part of the standards but often not implemented by other database software, such as window functions, plus some PostgreSQL-specific constructions which should be avoided if you want to write portable queries.

The final chapters cover performance tuning and replication, which are perhaps a bit too advanced for an 'up and running' book. Most of the performance tuning tips are generic though, and are worth a read even if you are currently using a different database server. Replication on the other hand is a complex topic and although the authors make a valiant effort, eight pages is insufficient to even begin to cover it.

Overall, the book serves as an excellent introduction to PostgreSQL. Despite my MySQL background and (possibly bad) habits, I was able to keep up with the authors most of the time, and feel that I could now start to use PostgreSQL without breaking things too badly. The short length (130 pages, excluding the appendix) means that many topics are skipped over rather than explored thoroughly, but for a practical 'up and running' guide this is not a problem. The authors have managed to strike the right balance between providing enough information for an overview and not overwhelming the reader in what may be their first experience with PostgreSQL. Well worth a read if you are experienced with another database server and want to see how PostgreSQL can meet your needs.

Intermediate Perl

Randal L. Schwartz, brian d foy and Tom Phoenix
O'Reilly Media
ISBN: 978-1-4493-9309-0
396pp.
£ 30.99
Published: August 2012

reviewed by Paul Waring

Having reviewed Learning Perl in the December 2011 newsletter, l seemed the next logical step. With the same authors, this book picks up where Learning Perl left off, and the second edition is possibly more up to date than your distribution's repository, covering Perl 5.14.

The first chapter is fairly brief and reiterates the basic knowledge of Perl which is assumed throughout – either that you have read the previous book or are familiar with control structures, subroutines, list operators and basic file manipulation. As with Learning Perl, all the code runs cleanly with warnings and strict mode enabled, although there is no mention of the `Modern::Perl` module which includes both modes and a few additional features.

The remainder of the first half of the book is largely devoted to references of all shapes and sizes. This is an area which often causes headaches, but is essential for most Perl programming beyond simple basic scripts, so it is good to see that the authors have spent plenty of time covering it. If you need to use a specific Perl environment, such as when deploying to a new server, then chapter 12 covers the topic of building your own Perl distribution, including the essential but often overlooked topic of documentation.

The second half of the book is dedicated to testing and objects, two topics which are important but rarely given the coverage they deserve. Objects in particular is a topic which took me some time to get my head around, but the authors have done a good job of explaining it. The only criticism I can make of the second half is that the authors jump between testing and objects, but the chapters are sufficiently self-contained that you can probably read all the chapters on objects first and then go back and tackle the ones covering testing.

Although short, I particularly enjoyed the final chapter on "Contributing to CPAN", as this is something I've wanted to do for some time but have been unable to find a well-written explanation.

Overall, this is an excellent book for Perl programmers who have got to grips with the basics and want to move on to the next level. There is a sensible balance between breadth and depth of coverage, the exercises are meaningful rather than contrived, and the text is as up to date as a printed edition can ever hope to be.

Mobile Javascript Applications Development

Adrian Kosmaczewski
O'Reilly Media
ISBN: 978-1-4493-2785-9
168pp.
£ 18.99
Published: June 2012

reviewed by Quentin Wright

This book provides an introduction to three frameworks for building cross-platform mobile applications, and iintroduces sample applications using JQuery Mobile, Sencha Touch and PhoneGap.

The first chapter discusses the new features in the HTML5 specification and focusses on aspects which have a bearing on mobile development. Mention is made of the HTML5 application cache, manifest files, geolocation, device orientation and motion and network connectivity. The canvas object is discussed along with CSS3 animations and transitions. Client side storage is the final topic referred to. All of these features are accompanied by snippets of illustrative Javascript code.

Chapter 2 provides a short overview about techniques in Javascript. It starts with links to Douglas Crockford's 2001 article and Simon Willison's 2006 Re-introduction to Javascript tutorial. There then follow a series of tips about code style with examples. These include how to organize code into namespaces, how to create objects, arrays, singleton objects, how to schedule function execution, how to concatenate strings, iterate over arrays, how to achieve code reflection with console.log.

Chapter 3 describes JQuery Mobile and describes a ToDo list application. At the end of chaptera examples of two JQuery applications are given: Codiqa, which can be used to prototype mobile applications and ThemeRoller for developers to create stylesheets.

Chapter 4 describes Sencha Touch, which is the first HTML5 framework for mobile devices. Again there is brief coverage of a ToDo application.

Chapter 5 discusses PhoneGap. PhoneGap is the only hybrid mobile application framework with on open source license. It allows the development of mobile apps with standard HTML, CSS and Javascript which are wrapped in native app shell. Additionally it exposes a bridge that gives access to native device features. The book explains how to create an iOS application on a Mac, to create an Android application with Eclipse or IntelliJ and to create a Windows Phone application. The use of the Javascript bridge is covered with Javascript examples for each device.

Chapter 6 Introduces the use of browser web inspectors and Adobe Shadow for debugging. A mention of is made of Jasmine and Siesta for testing.

Chapter 7, Conclusion, is a single page with a table comparing the pros and cons of Sencha Touch and JQuery mobile.

The book has no index although there is a fairly comprehensive contents list.

This book is not for beginners. When I started to work through the example in Chapter 3 I realised that you need to have set up the Android SDK and be familiar with the use of the emulator. To run the examples in an iPhone environment you can use the iPhoney emulator – not mentioned in the book.

The simplest of examples are presented rather than being worked through. The book would have more valuable with richer examples and a step-by-step guide.

There is a lot of information presented in the book, but it doesn't give the "hands-on tour" the copy on the cover promises. I would have like to have seen a discussion of the different kinds of mobile applications, that's to say web, native and hybrid. Additionally there are other frameworks which could have been put into some context, as for example JQTouch which is like JQMobile but is a lighter webkit specific framework for small-screen devices. Also PhoneGap can be used to take a Sproutcore web application to produce a native wrapper which can be placed in the Apple App Store or Google Play store.

For a useful overview of the available frameworks take a look at the Wikipedia page "HTML 5 in Mobile Devices".
It's probably inevitable that in such a rapidly evolving area as mobile development it's difficult to present a collected view of the best techniques. This book has its place amongst others and might be helpful in getting some feel for the use of Javascript and the particular frameworks before looking elsewhere for more detail.

Learning Javascript Design Patterns

Addy Osmani
O'Reilly Media
ISBN: 1449331815
254pp.
£ 26.99
Published: August 2012

reviewed by Quentin Wright

Addy Osmani is well-known in the Javascript community. He works for Google and has a mere 96 projects on github. He wrote the first definitive on-line book for backbone.js called "Developing Backbone.js Applications" and many of us watched it evolve over several months. More recently he wrote an on-line version of "Learning Javascript Design Patterns". After an approach from O'Reilly this conventional paper version has appeared which contains the bulk of the on-line book with extra sections taken from the backbone.js book. In the pipe-line there's another O'Reilly book due out in March 2013 and perhaps unsurprisingly called "Developing Backbone.js Applications".

The book starts with some very short chapters of a page or two each setting the scene, with titles: “What is a Pattern?”, “Patternity Testing”, “Proto-Patterns and the Rule of Three”, “The Structure of a Design Pattern”, “Anti-Patterns” and “Design Pattern Categorization”. These will be familiar to those with a general knowledge of the use of design patterns in a general context.

The meat of the book starts with Chapter 9 in which Javascript implementations of many design patterns are laid out and discussed. In many cases the examples are fairly down-to-earth and practical. Depending upon one’s knowledge of patterns it could be advantageous to have the original “Gang of Four” book to hand. For most patterns there are discussions of the advantages and disadvantages of that particular pattern and suggestions as to alternatives to use. The section rounds off with use of the Flyweight Pattern to share data and in the DOM layer to act as an event manager.

Chapter 10 is entitled “Javascript MV* Patterns”. This discusses and compares the MVC (Model-View-Controller), MVP (Model-View-Presenter), MVVM (Model-View-ViewModel and MVC/MV* (as in backbone.js) with the classical MVC architecture. I found this very useful as in recent months I’ve been experimenting with backbone.js as well as looking at spine.js and KnockoutJS. Getting to understand the way these frameworks worked seemed to depend upon ploughing through many blog entries as well as looking at the code. The chapter in the book gives a more collected explanation and for me was particularly valuable.

Chapter 11 is called “Modern Modular Design Patterns” and is a brief discussion about the use of AMD and CommonJS to handle modules. AMD along with require.js has emerged as the preferred approach for web applications. CommonJS has wider application on the server side. There’s mention of the proposed Harmony ES standard along with suggestions as how one can experiment using transpilers like Traceur or Esprima.

Chapter 12, “Design Patterns in JQuery” looks at JQuery methods and discusses which pattern they correspond to and how they might be used in practice.

Chapter 13, “JQuery Plug-in Design Patterns”, is for advanced developers and discusses the way that patterns can be used to develop JQuery Plug-ins.

Chapter 14, “Conclusion” is very short, saying that we should be aware of design patterns in Javascript and recommending further reading in the form of Martin Fowler’s “Patterns of Enterprise Application Architecture” and Stoyan Stefanov’s “Javascript Patterns”.

Recently I passed the book round at our local Javascript group for the members to browse and offer their comments. It got a universal “thumbs down”! I’m not sure whether this was because the immediate application of design patterns to day-to-day work isn’t obvious.

The title “Learning Javascript Design Patterns” is something of a misnomer. This implies that by working through the book you will learn how to use design patterns with Javascript.

Really it's a reference work. Something to be studied over Christmas perhaps and then kept handy for reference. Chapter 9 with its Javascript rendering of common patterns and discussion is really the core of the book and in many ways on its own is a justification for buying a hard-copy version of the book. Chapter 10 on the MV* patterns is useful. Chapter 12 where the JQuery code is used to illustrate the presence of patterns helps to give practical examples. Ideally the book would go on to discuss examples of situations where a design pattern might be appropriate and then present or develop suitable code.

SciPy and NumPy: An Overview for Developers

Eli Bressert
O'Reilly Media
ISBN: 978-1-4493-0546-8
68pp.
£ 18.99
Published: November 2012

reviewed by Roger Whittaker

On opening the envelope in which this book arrived, I was surprised to see what it was. For this is a very slim volume indeed: only 68 pages (officially – actually only 57 numbered pages of the main text). So the official price of £18.99 sounds a little steep.

SciPy and NumPy provide fast scientific and numerical tools for Python. These include the use of large arrays and matrices, linear algebra, curve fitting, data modelling and plotting and statistical functions.

The book gives brief introductions and examples of all of these and an overview of ways in which one might want to use the tools available. I'm not sure whether the book really does fill a gap in the market. When I tried to use some of NumPy's capabilities a few years ago, I found that the documentation that I saw online seemed out of date with respect to the software itself. That is no longer the case, and the documentation online seems to be very complete and includes a rather nice tutorial and "cookbook".

So I'm not entirely convinced of the need for this book, but it could be useful to have by your desk if you don't baulk at paying exactly £1 for every three pages.

Contributors

Charlie Harvey lives in Oxford and for the last three years has been working for New Internationalist. Before that he worked for "People and Planet", a student organisation campaigning on world poverty and environmental issues. Charlie's online presence is at `http://charlieharvey.org.uk`.

Mark Keating is the Managing Director of Shadowcat Systems and a keen contributor to an array of community projects. He describes himself as an adminion organiser, cat herder,

writer and photographer. He lives in Lancaster with two children, a cat and tropical fish. Mark has heard of the concept of 'free time' but doesn't like to indulge in it.

Jane Morrison is Company Secretary and Administrator for UKUUG, and manages the UKUUG office at the Manor House in Buntingford. She has been involved with UKUUG administration since 1987. In addition to UKUUG, Jane is Company Secretary for a trade association (Fibreoptic Industry Association) that she also runs from the Manor House office.

Jess Robinson is a software developer and technical writer, who lives and works from Swindon in Wiltshire. She specialises in Perl development and writing technical documentation. She contributes to many software projects, helping new users, implementing extensions and mostly writing/improving documentation. When she is not programming, Jess' other interests include 3D printing, walking and the Swindon Hackspace.

Paul Waring is chairman of UKUUG and a director of a wholesale insurance broker. Outside of work he can usually be found filing documentation bugs against various open source and free software projects.

Roger Whittaker works for SUSE supporting SUSE Linux Enterprise Server for major customers in the UK. He is also the UKUUG Newsletter Editor, and co-author of three successive versions of a SUSE book published by Wiley.

Quentin Wright is a Director of Sunfield Technology working with Linux and Ruby and system integration projects. In his spare time in between playing with reluctant motor vehicles he is pre-occupied with Web and Javascript programming.

Contacts

Kimball Johnson
UKUUG Chairman
Preston

Paul Waring
Treasurer
Manchester

Jon Dowland
Council member
Newcastle

Martin Houston
Council member
Chelmsford

Holger Kraus
Council member
Leicester

Ian Norton
Council member
Rochdale

Stephen Quinney
Council member
Edinburgh

Quentin Wright
Council member
Warwick

Roger Whittaker
Newsletter Editor
London

Alain Williams
UKUUG System Administrator
Watford

Sam Smith
Events and Website
Cambridge

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273475
Fax: 01763 273255
office@ukuug.org