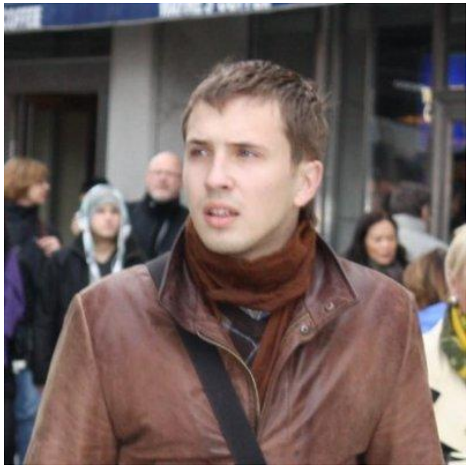
Самообразование в IT

BCS Andrey Toumilovich
BCS Dmitry Balabka

TSI
TRANSPORT AND
TELECOMMUNICATION
INSTITUTE

Andrey Toumilovich
Software Developer @ ingenico GROUP
https://lv.linkedin.com/in/atoumilovich
Andrey.Toumilovich@gmail.com



Dmitry Balabka
Software Architect @ INTEXSYS
http://lv.linkedin.com/in/dmitrybalabka
Dmitry.Balabka@gmail.com

План

1. Вступление
2. Самообразование в общем
3. Самообразование в области Веб разработки
4. Самообразование в области C/C++

slack

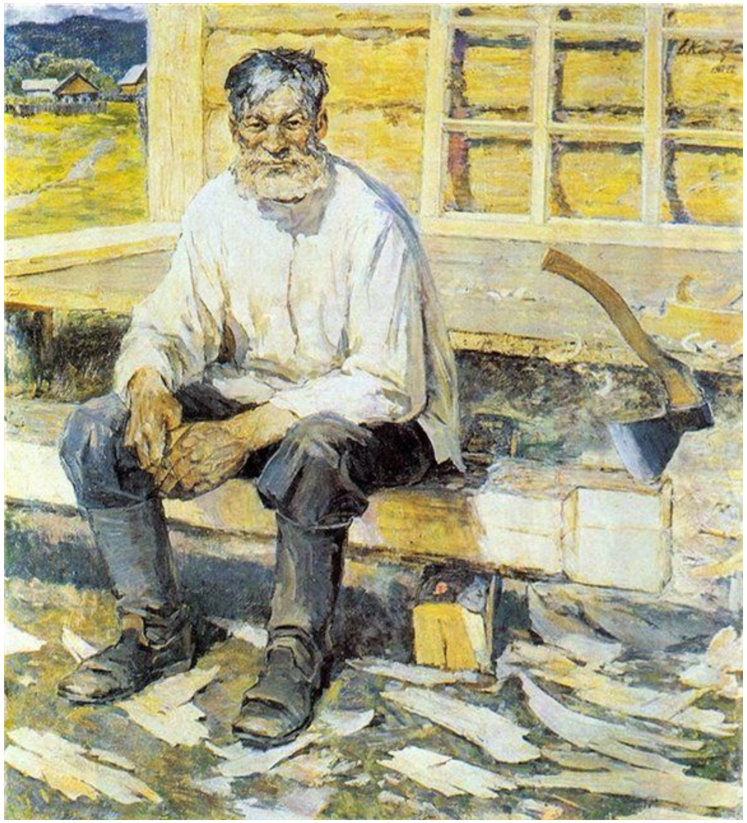
https://tsi-students.slack.com/

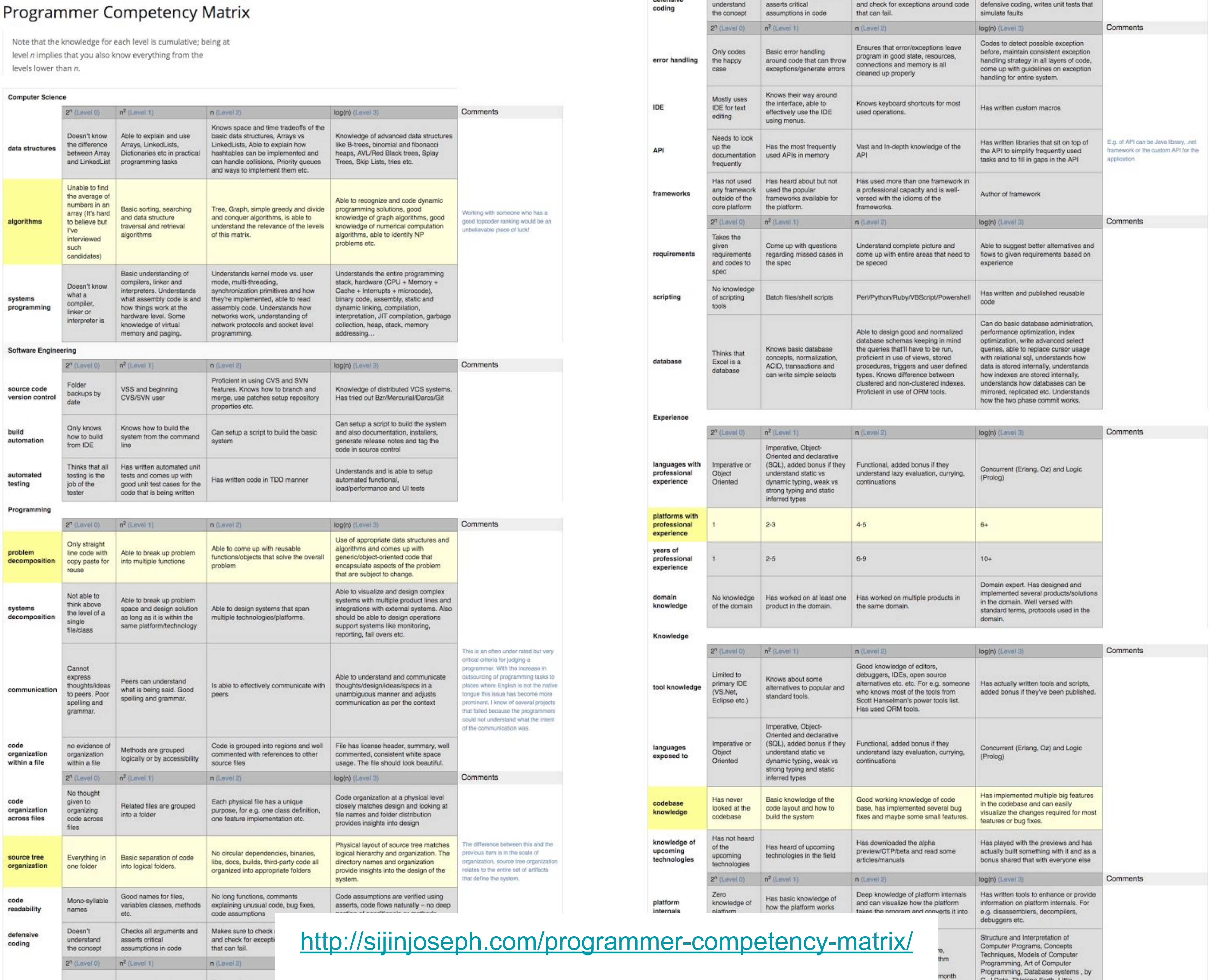
https://github.com/tsi-students/8-bit-panda-presentation

LET'S GET
STARTED

Как становиться ~~плотником~~ программистом

1. Осознание
2. Попробовали
3. Получили первый опыт
4. Развитие





Programmer Competency Matrix

Note that the knowledge for each level is cumulative; being at level *n* implies that you also know everything from the levels lower than *n*.

Computer Science

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
data structures	Doesn't know the difference between Array and LinkedList	Able to explain and use Arrays, LinkedLists, Dictionaries etc in practical programming tasks	Knows space and time tradeoffs of the basic data structures, Arrays vs LinkedLists, Able to explain how hashtables can be implemented and can handle collisions, Priority queues and ways to implement them etc.	Knowledge of advanced data structures like B-trees, binomial and fibonacci heaps, AVL/Red Black trees, Splay Trees, Skip Lists, tries etc.	
algorithms	Unable to find the average of numbers in an array (It's hard to believe but I've interviewed such candidates)	Basic sorting, searching and data structure traversal and retrieval algorithms	Tree, Graph, simple greedy and divide and conquer algorithms, is able to understand the relevance of the levels of this matrix.	Able to recognize and code dynamic programming solutions, good knowledge of graph algorithms, good knowledge of numerical computation algorithms, able to identify NP problems etc.	Working with someone who has a good topcoder ranking would be an unbelievable piece of luck!
systems programming	Doesn't know what a compiler, linker or interpreter is	Basic understanding of compilers, linker and interpreters. Understands what assembly code is and how things work at the hardware level. Some knowledge of virtual memory and paging.	Understands kernel mode vs. user mode, multi-threading, synchronization primitives and how they're implemented, able to read assembly code. Understands how networks work, understanding of network protocols and socket level programming.	Understands the entire programming stack, hardware (CPU + Memory + Cache + Interrupts + microcode), binary code, assembly, static and dynamic linking, compilation, interpretation, JIT compilation, garbage collection, heap, stack, memory addressing…	

Software Engineering

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
source code version control	Folder backups by date	VSS and beginning CVS/SVN user	Proficient in using CVS and SVN features. Knows how to branch and merge, use patches setup repository properties etc.	Knowledge of distributed VCS systems. Has tried out Bzr/Mercurial/Darcs/Git	
build automation	Only knows how to build from IDE	Knows how to build the system from the command line	Can setup a script to build the basic system	Can setup a script to build the system and also documentation, installers, generate release notes and tag the code in source control	
automated testing	Thinks that all testing is the job of the tester	Has written automated unit tests and comes up with good unit test cases for the code that is being written	Has written code in TDD manner	Understands and is able to setup automated functional, load/performance and UI tests	

Programming

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
problem decomposition	Only straight line code with copy paste for reuse	Able to break up problem into multiple functions	Able to come up with reusable functions/objects that solve the overall problem	Use of appropriate data structures and algorithms and comes up with generic/object-oriented code that encapsulate aspects of the problem that are subject to change.	
systems decomposition	Not able to think above the level of a single file/class	Able to break up problem space and design solution as long as it is within the same platform/technology	Able to design systems that span multiple technologies/platforms.	Able to visualize and design complex systems with multiple product lines and integrations with external systems. Also should be able to design operations support systems like monitoring, reporting, fail overs etc.	
communication	Cannot express thoughts/ideas to peers. Poor spelling and grammar.	Peers can understand what is being said. Good spelling and grammar.	Is able to effectively communicate with peers	Able to understand and communicate thoughts/design/ideas/specs in a unambiguous manner and adjusts communication as per the context	This is an often under rated but very critical criteria for judging a programmer. With the increase in outsourcing of programming tasks to places where English is not the native tongue this issue has become more prominent. I know of several projects that failed because the programmers could not understand what the intent of the communication was.
code organization within a file	no evidence of organization within a file	Methods are grouped logically or by accessibility	Code is grouped into regions and well commented with references to other source files	File has license header, summary, well commented, consistent white space usage. The file should look beautiful.	

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
code organization across files	No thought given to organizing code across files	Related files are grouped into a folder	Each physical file has a unique purpose, for e.g. one class definition, one feature implementation etc.	Code organization at a physical level closely matches design and looking at file names and folder distribution provides insights into design	
source tree organization	Everything in one folder	Basic separation of code into logical folders.	No circular dependencies, binaries, libs, docs, builds, third-party code all organized into appropriate folders	Physical layout of source tree matches logical hierarchy and organization. The directory names and organization provide insights into the design of the system.	The difference between this and the previous item is in the scale of organization, source tree organization relates to the entire set of artifacts that define the system.
code readability	Mono-syllable names	Good names for files, variables classes, methods etc.	No long functions, comments explaining unusual code, bug fixes, code assumptions	Code assumptions are verified using asserts, code flows naturally – no deep	
defensive coding	Doesn't understand the concept	Checks all arguments and asserts critical assumptions in code	Makes sure to check and check for excepti that can fail.		

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)

coding	understand the concept	asserts critical assumptions in code	and check for exceptions around code that can fail.	defensive coding, writes unit tests that simulate faults	

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
error handling	Only codes the happy case	Basic error handling around code that can throw exceptions/generate errors	Ensures that error/exceptions leave program in good state, resources, connections and memory is all cleaned up properly	Codes to detect possible exception before, maintain consistent exception handling strategy in all layers of code, come up with guidelines on exception handling for entire system.	
IDE	Mostly uses IDE for text editing	Knows their way around the interface, able to effectively use the IDE using menus.	Knows keyboard shortcuts for most used operations.	Has written custom macros	
API	Needs to look up the documentation frequently	Has the most frequently used APIs in memory	Vast and In-depth knowledge of the API	Has written libraries that sit on top of the API to simplify frequently used tasks and to fill in gaps in the API	E.g. of API can be Java library, .net framework or the custom API for the application
frameworks	Has not used any framework outside of the core platform	Has heard about but not used the popular frameworks available for the platform.	Has used more than one framework in a professional capacity and is well-versed with the idioms of the frameworks.	Author of framework	

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
requirements	Takes the given requirements and codes to spec	Come up with questions regarding missed cases in the spec	Understand complete picture and come up with entire areas that need to be speced	Able to suggest better alternatives and flows to given requirements based on experience	
scripting	No knowledge of scripting tools	Batch files/shell scripts	Perl/Python/Ruby/VBScript/Powershell	Has written and published reusable code	
database	Thinks that Excel is a database	Knows basic database concepts, normalization, ACID, transactions and can write simple selects	Able to design good and normalized database schemas keeping in mind the queries that'll have to be run, proficient in use of views, stored procedures, triggers and user defined types. Knows difference between clustered and non-clustered indexes. Proficient in use of ORM tools.	Can do basic database administration, performance optimization, index optimization, write advanced select queries, able to replace cursor usage with relational sql, understands how data is stored internally, understands how indexes are stored internally, understands how databases can be mirrored, replicated etc. Understands how the two phase commit works.	

Experience

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
languages with professional experience	Imperative or Object Oriented	Imperative, Object-Oriented and declarative (SQL), added bonus if they understand static vs dynamic typing, weak vs strong typing and static inferred types	Functional, added bonus if they understand lazy evaluation, currying, continuations	Concurrent (Erlang, Oz) and Logic (Prolog)	
platforms with professional experience	1	2-3	4-5	6+	
years of professional experience	1	2-5	6-9	10+	
domain knowledge	No knowledge of the domain	Has worked on at least one product in the domain.	Has worked on multiple products in the same domain.	Domain expert. Has designed and implemented several products/solutions in the domain. Well versed with standard terms, protocols used in the domain.	

Knowledge

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
tool knowledge	Limited to primary IDE (VS.Net, Eclipse etc.)	Knows about some alternatives to popular and standard tools.	Good knowledge of editors, debuggers, IDEs, open source alternatives etc. etc. For e.g. someone who knows most of the tools from Scott Hanselman's power tools list. Has used ORM tools.	Has actually written tools and scripts, added bonus if they've been published.	
languages exposed to	Imperative or Object Oriented	Imperative, Object-Oriented and declarative (SQL), added bonus if they understand static vs dynamic typing, weak vs strong typing and static inferred types	Functional, added bonus if they understand lazy evaluation, currying, continuations	Concurrent (Erlang, Oz) and Logic (Prolog)	
codebase knowledge	Has never looked at the codebase	Basic knowledge of the code layout and how to build the system	Good working knowledge of code base, has implemented several bug fixes and maybe some small features.	Has implemented multiple big features in the codebase and can easily visualize the changes required for most features or bug fixes.	
knowledge of upcoming technologies	Has not heard of the upcoming technologies	Has heard of upcoming technologies in the field	Has downloaded the alpha preview/CTP/beta and read some articles/manuals	Has played with the previews and has actually built something with it and as a bonus shared that with everyone else	

	2^n (Level 0)	n^2 (Level 1)	n (Level 2)	log(n) (Level 3)	Comments
platform internals	Zero knowledge of platform	Has basic knowledge of how the platform works	Deep knowledge of platform internals and can visualize how the platform takes the program and converts it into	Has written tools to enhance or provide information on platform internals. For e.g. disassemblers, decompilers, debuggers etc.	
			re, thm month	Structure and Interpretation of Computer Programs, Concepts Techniques, Models of Computer Programming, Art of Computer Programming, Database systems , by C. J Date, Thinking Forth, Little	

http://sijinjoseph.com/programmer-competency-matrix/

Programmer Competency Matrix. Level 0

- Computer Science
  - Doesn't know the difference between Array and LinkedList
  - Unable to find the average of numbers in an array (It's hard to believe but I've interviewed such candidates)
  - Doesn't know what a compiler, linker or interpreter is
- Software Engineering
  - Folder backups by date (source code version control)
  - Only knows how to build from IDE
  - Thinks that all testing is the job of the tester
- Programming
  - Only straight line code with copy paste for reuse
  - Not able to think above the level of a single file/class
  - Cannot express thoughts/ideas to peers. Poor spelling and grammar.
- Programming
  - no evidence of organization within a file
  - No thought given to organizing code across files
  - Everything in one folder
  - Mono-syllable names
  - Only codes the happy case (error handling)
  - Mostly uses IDE for text editing
  - Needs to look up the documentation frequently
  - Has not used any framework
  - Takes the given requirements and codes to spec
  - Thinks that Excel is a database
  - Imperative or Object Oriented
  - No knowledge of the domain
  - Limited to primary IDE
  - Has not heard of the upcoming technologies
  - Zero knowledge of platform internals
  - Heard about blogs but never got the time.

How to become software developer

- Knowledge of the domain
- Programming language/s and standard library knowledge
- Framework and libraries knowledge
- Experience

How to learn new technology

- Course for dummies first (introduction course)
  - Online interactive tutorials
  - Video tutorials
  - others
- Intermediate or advanced level course
- Official documentation
  - “Get started”, “Quick start” etc.
  - Tutorials
  - Cookbook
  - Reference
  - API
- Read blogs and articles
- QnA and forums for questions

Online Courses

KHANACADEMY

https://www.khanacademy.org/

coursera

https://www.coursera.org/

edX

https://www.edx.org/

Hexlet university

https://ru.hexlet.io/

UDACITY

udemy

MITOPENCOURSEWARE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

codecademy

iTunes U

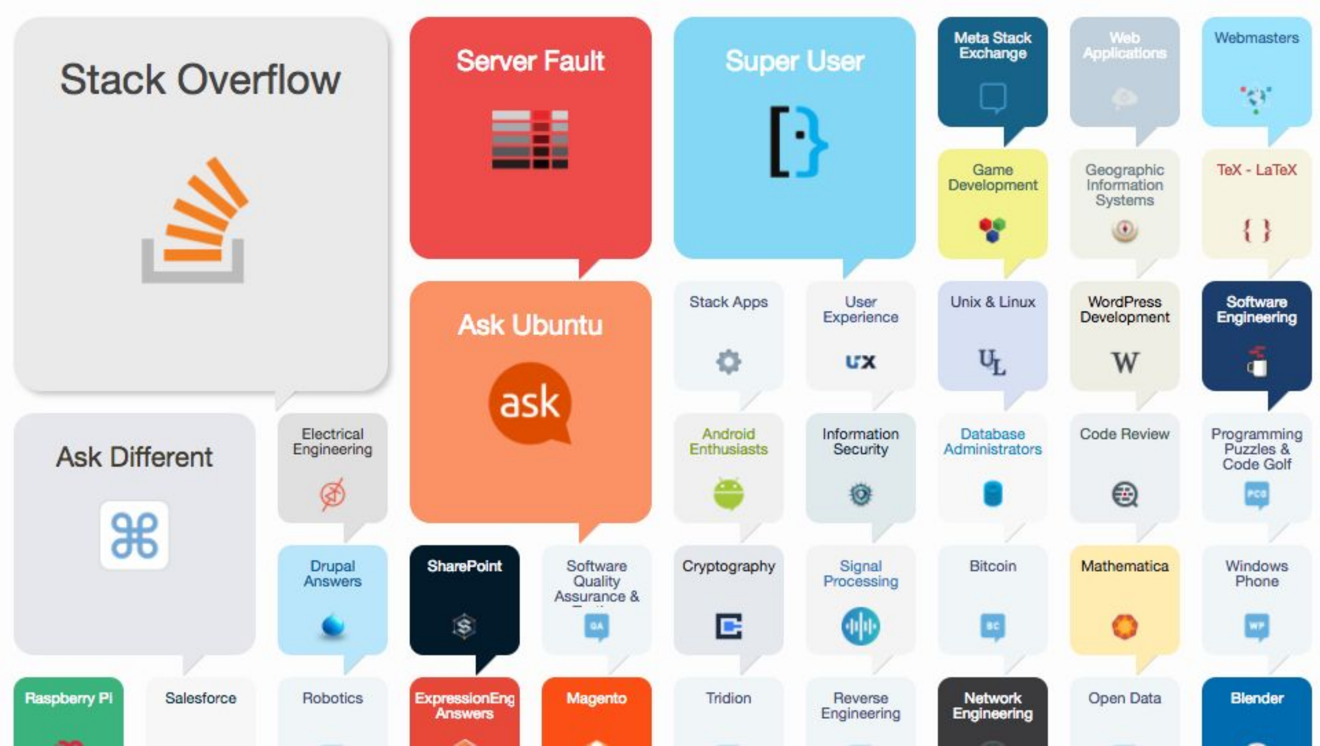
Code School

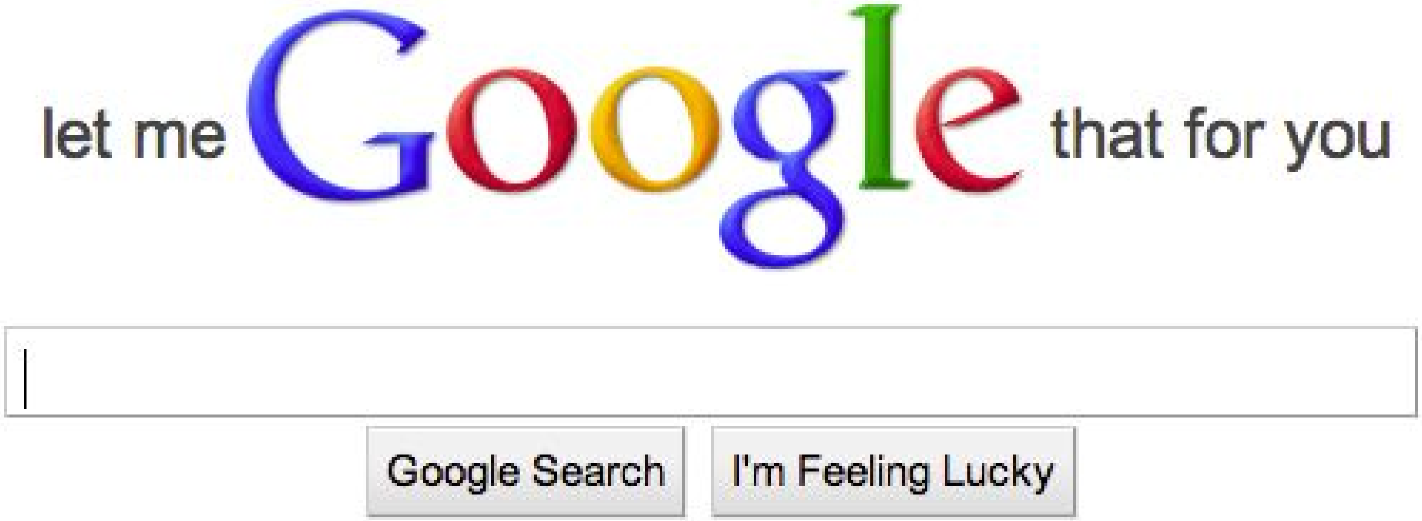
Where ask questions?

stackoverflow

StackExchange

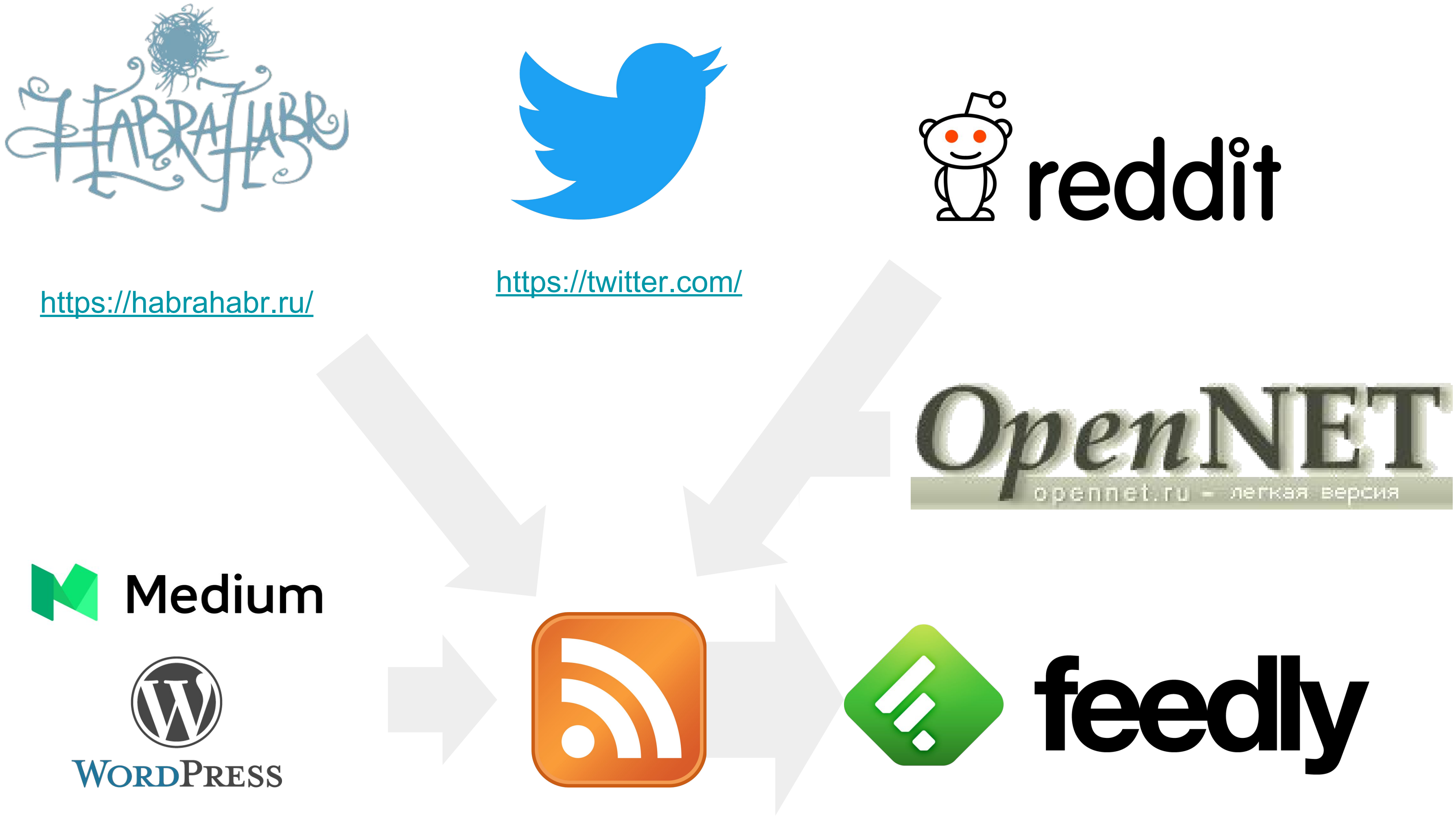
http://stackoverflow.com/
http://ru.stackoverflow.com/





http://letmegooglethat.com/?q=php

News

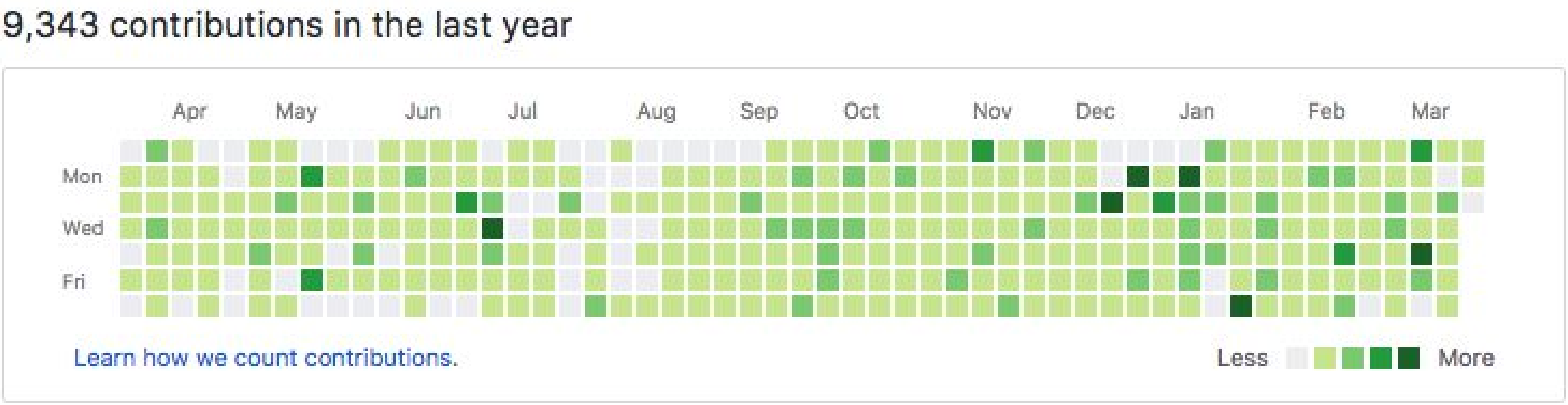


https://habrahabr.ru/

https://twitter.com/

github

SOCIAL CODING



9,343 contributions in the last year

Learn how we count contributions.

Jokes

bash.im — Цитатник Рунета

govnokod.ru

по колено
в коде

ithappens

Online Challenges

CodinGame

https://www.codingame.com/

Google Code-in

https://developers.google.com/open-source/gci/

CODEFORCES β
Sponsored by Telegram

http://codeforces.com/

code jam
printf("hello, world!\n");

https://code.google.com/codejam/

topcoder™

http://www.topcoder.com/

stripe

Google
SUMMER of CODE
2015

https://summerofcode.withgoogle.com/

coderbyte

kaggle

Conferences in Riga

devclub.lv

LatCraft

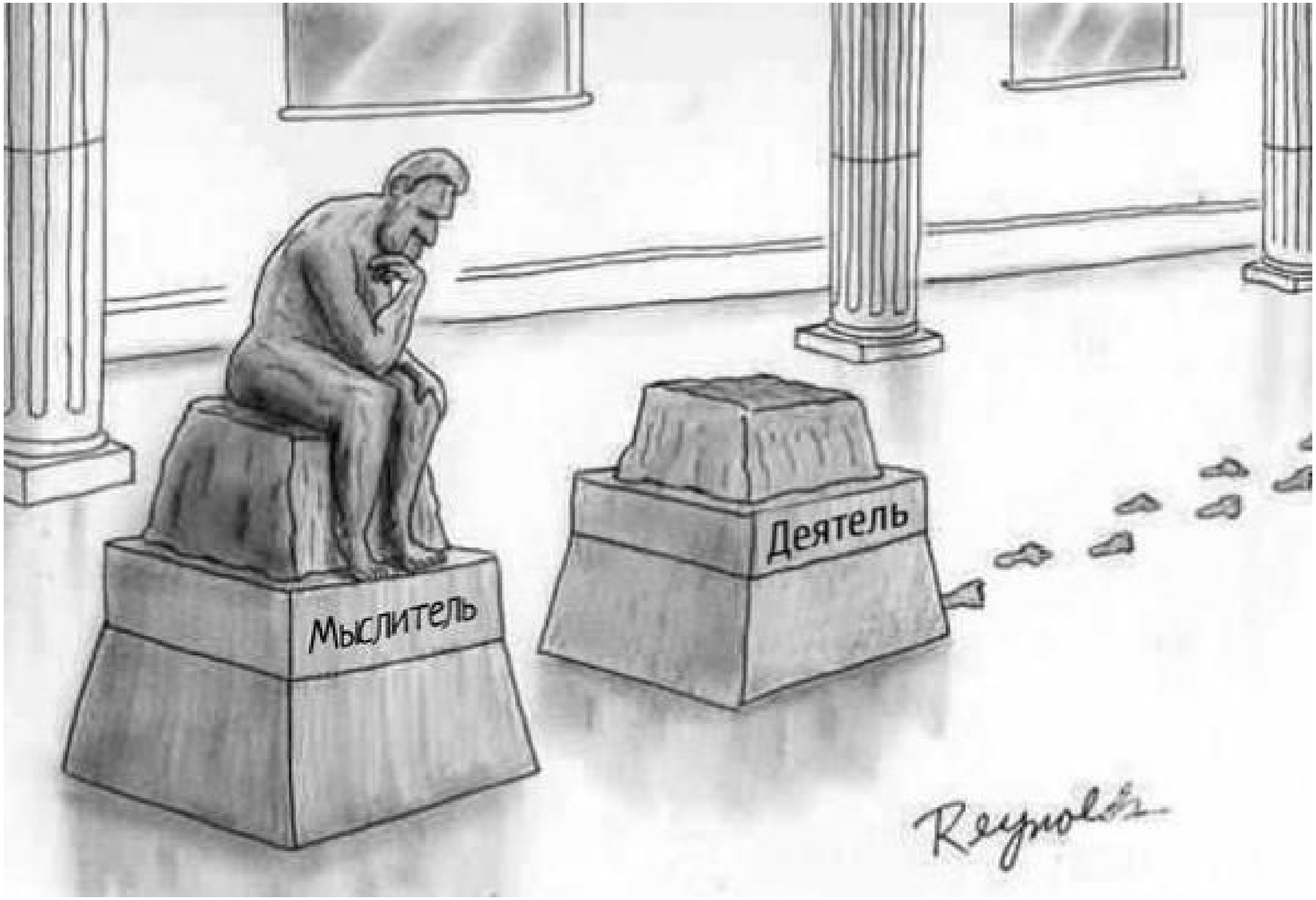
tech hub

GDG Riga

Google Developers Group Riga

DevTernity

meetup



“Обучение стало главной тенденцией двадцать первого века. К сожалению, это также стало новой формой прокрастинации. ”

https://habrahabr.ru/company/everydaytools/blog/322980/

Самообразование в области C/C++

Самообразование в области C/C++

- Изучение с позиции новичка
  - Изучение основ
  - Критическое мышление (незакостенелость)
- Изучение с позиции опытного программиста
  - Изучение особенностей
  - Переучивание

Источники

- Книги
- Справочники
- Интернет ресурсы
- Ментор

Библиотеки

- Пользовательский интерфейс (UI)
  - Microsoft WinApi
  - Qt
  - X11
  - GTK
- Сетевое взаимодействие
  - Berkley Sockets
  - Windows Sockets
- Параллелизм
  - Pthreads
  - Windows threads
  - OpenCL
- Графика
  - DirectX
  - OpenGL

Интернет ресурсы

- Edx.org
- SlideShare.net
- TutorialsPoint.com
- habrahabr.ru

Самообразование в области Web технологий

Front-end developer. Javascript

- Базовые знания о том как правильно писать JS(прочитать часть 2 - Essentials, остальное не нужно):
  - https://www.dropbox.com/s/nl0yyylekiuzvcn/JavaScript%20Patterns.pdf?dl=0
- Желательно почитать, вот тут про то как не нужно делать (appendix A и B):
  - https://www.dropbox.com/s/9k7s4u2cw45146v/JavaScript%20The%20Good%20Parts.pdf?dl=0
- Здесь описано как можно писать ОО код в JS(читать первую часть, остальное не нужно, но можно):
  - https://www.dropbox.com/s/dmoneb70hsexvvk/Pro%20Javascript%20Patterns.pdf?dl=0
- Тут почитать как правильно оформлять JS(привожу два источника, чтоб не было впечатления, есть какой-то один стандарт):
  - https://google.github.io/styleguide/javascriptguide.xml
  - https://github.com/airbnb/javascript
- А это просто хороший сборник всяких Pattern с ссылками на источники. Тут бы я посоветовал посмотреть как правильно работать с jQuery:
  - http://shichuan.github.io/javascript-patterns/
- Потом бы посоветовал почитать про Memory Leak в JS как работает Grabage collector и что это такое, как оптимизировать JS и т.п.. Можно начать тут:
  - https://developers.google.com/speed/articles/optimizing-javascript
- Про работу с DOM в JS. Еще нужно почитать про то как работать в JS с DOM. Очень важные момент про то как изменять DOM(это как правило будешь делать в jQuery) и как работают Dom Events:
  - http://www.smashingmagazine.com/2013/11/an-introduction-to-dom-events/

Front-end developer. HTML/CSS and slicing

- Курсы
  - https://developer.mozilla.org/en-US/docs/Learn
- Визуальному форматированию текста:
  - http://css-live.ru/articles/obzor-inlajnovyj-kontekst-formatirovaniya.html
- Framework
  - http://getbootstrap.com/
- Methodology
  - https://en.bem.info/methodology/

https://github.com/sindresorhus/awesome#front-end-development

Back-end developer. PHP

- Online course
  - Javaguru курс “Веб разработка с PHP”
    - http://www.javaguru.lv/php-web-development.html
  - https://knpuniversity.com/
  - https://www.codecademy.com/learn/php
- Frameworks
  - Symfony
    - http://symfony.com/doc/current/index.html
    - https://knpuniversity.com/

Awesome programming

- Awesome Javascript
  - https://github.com/sorrycc/awesome-javascript
- Awesome CSS
  - https://github.com/sotayamashita/awesome-css
- Awesome PHP
  - https://github.com/ziadoz/awesome-php
- And other Awesome technologies
  - https://github.com/sindresorhus/awesome

Q&A