Programming Dojo

Samuel Grant Dawson Williams

May 14, 2010

1 Abstract

Computer programming languages are the primary platform for communicating computer science concepts and ideas. Learning a programming language gives you the ability to read and write code. Understanding different syntax and semantic behaviour gives a wide scope for understanding the possibilities when approaching a problem and considering its solutions. This report explores the development of a website which has been specifically designed to expose both students and teachers to a wide range of programming languages.

2 Introduction

Computer Science is not an easy subject to teach at any level because unlike many other formal subjects, Computer Science lacks a well understood rigorous definition[1]. Many teachers do not have a formal education of sufficient relevance and coverage. Because of this, students may not gain a wide level of exposure to correctly explained computer science concepts and in general be left with a very poor opinion of the field. If teachers lack confidence, so too will students.

3 Background

Many teachers do not know about the different programming languages that are available. Because of this, students are missing worthwhile learning opportunities and have to deal with the consequences of the teacher's choices. These include proprietary lock-in, poor interactivity, limited extendability and poor documentation, to name a few.

Understanding the differences between major programming languages and how they relate to each other is important for the development of a complete understanding of computer programming. Merleau-Ponty in his book 'Phenomenology of Perception'[2] discusses the 'Frontality of Experience' and how this applies to scientific understanding. In modern Computer Science education, students are often only taught that which is specifically required to advance to other areas. In general, this leads to a very shallow understanding of Computer Science concepts and how they relate to each other.

By exposing students to more than one language, especially when similar concepts are expressed in different semantic interpretation or syntactic declaration, we are empowering them to step back and construct a more complete understanding of the concept. This leads to a much deeper understanding of both specific and related concepts.

Teaching programming in an educational context involves many elements.

- The syntax must be easy for both writing new code and reading existing code.
- The conceptual load must be sufficiently low so that students can learn few concepts at a time.
- High quality and easily accessible documentation for language features and libraries.
- Easy to execute programs which encourage code reuse and structured source code.
- An easily accessible educated community of people who support the language and its use.
- Educationally relevant lesson plans and supporting resources designed for use in schools.

No one language on its own fulfils all of these criteria, but there are many languages which meet some or all of these criteria in different ways. Sometimes the teaching material or learning goals dictates the set of available languages or their relevance to in an educational context.

Because there is no one right answer, both teachers and students are expected to evaluate programming languages on a case by case basis. This can be as simple as choosing JavaScript for embedding in a webpage, or using C for game development. However, without the right background and knowledge about the available choices, this decision can be difficult.

There are many existing resources which list available programming languages. Wikipedia has a list[3] of programming languages that is highly comprehensive and fairly accurate, but its comprehensive coverage is also its weakness; it covers a large number of programming languages which are simply not relevant in todays educational environment. RosettaCode[4] is a wiki with many examples of programming languages for comparison purposes, but it does not offer any guidance or advice about said languages, and the implementations are often of poor quality. Although these sites are extensive in their scope, they do not provide sufficient support for helping teachers and students choose a programming language.

Many sites exist which advocate specific languages for use in education[5][6][7], and in general, many sites exist advocating specific languages for use by beginner programmers. There are even several online pages[8][9] which compare and contrast languages to varying levels. Because these sites generally only focus on one language, they don't provide balanced points of comparison with other programming languages.

I believe that none of the above sites are generally suitable for use in the classroom. An educational resource has to be conceptually focused, well organised and relevant.

4 Method

A good way to reach a wide audience is to publish a resource online. There are also a number of benefits to this kind of format: it can be updated and improved quickly and it is easily accessible to many people without the need for physical distribution. Online resources can also be easily linked to other online resources, and given the affordance of a modern web browser, can have various interactivity features.

However, in order to engage teachers more directly, a physical resource that can be used in the classroom is also valuable. There are many options: books, posters, stickers, games, activities. In many cases, it is possible to have these offline resources link through to an online resource which provides extended information.

4.1 Website

Initially content was integrated with the main Orion Transfer[10] website. Several pages were developed: a programming language comparison chart, an evaluation of Ruby as a programming language for use in education, a video discussing the use of Ruby in an educational context and a collection of links relevant to teaching and learning Ruby.

After receiving initial feedback and considering options, a separate website called 'Programming Dojo' was incrementally developed over two months[11]. Further feedback was continually solicited over this period through online mailing lists and offline discussions.

The title 'Programming Dojo' was chosen due to the nature of our goal: Programming is a pathway to a greater understanding of the universe (in a metaphysical sense). The word 'Dojo' is actually pronounced 'Dou-Jyou', and written in Japanese Kanji '道場'; a crude translation: 'Dou' means pathway, and 'Jyou' means place. 'Programming Dojo' simply means the place where one can start on the pathway of programming.

The website currently consists of several sections:

- A front page which has been designed to draw people into the site (see figure 1).
- A page for each major programming language which has a brief description, example code and external links (see figure 2).
- A page for each key terminology used for describing computer programming languages.
- A set of study resources relevant to computer programming (still under development).
- An educationally relevant programming language comparison chart (see figure 3).
- A page for teachers about the programming language posters which includes a freely downloadable version of the posters.

I designed the site using relevant modern standards (HTML5, CSS, JavaScript, etc). Utopia[12] (a free and open source content management framework I developed) is used to simplify the construction and ongoing development of the site. To this end, Git[13] is used for version control. Due to the dynamic nature of a website with a lot of information, it is useful keep track of changes, and manage external dependencies.

The source code syntax highlighting on the site relies on jQuery.Syntax[14] (a free and open source JavaScript syntax highlighting algorithm which I also developed). Several additional syntax brushes were developed specifically for the site including: `pascal`, `perl5`, `haskell`, `smalltalk` and `java`.

The front page uses Quicksand[15] to provide an interactive list of programming languages. The logos for the languages were compiled from the respective languages, and when no logo was available I have chosen an image which I feel is relevant.

The initial set of source code examples were derived from RosettaCode[4]. They did not have comments explaining what was happening, nor did they use the best idiomatic approach in every case.

I have reviewed and corrected as many examples as possible and added relevant comments. This involved compiling/executing the example code and ensuring consistent output. Where further advice was needed, relevant sources were solicited for better examples, source code comments, explanations, descriptions and clarifications on existing text.

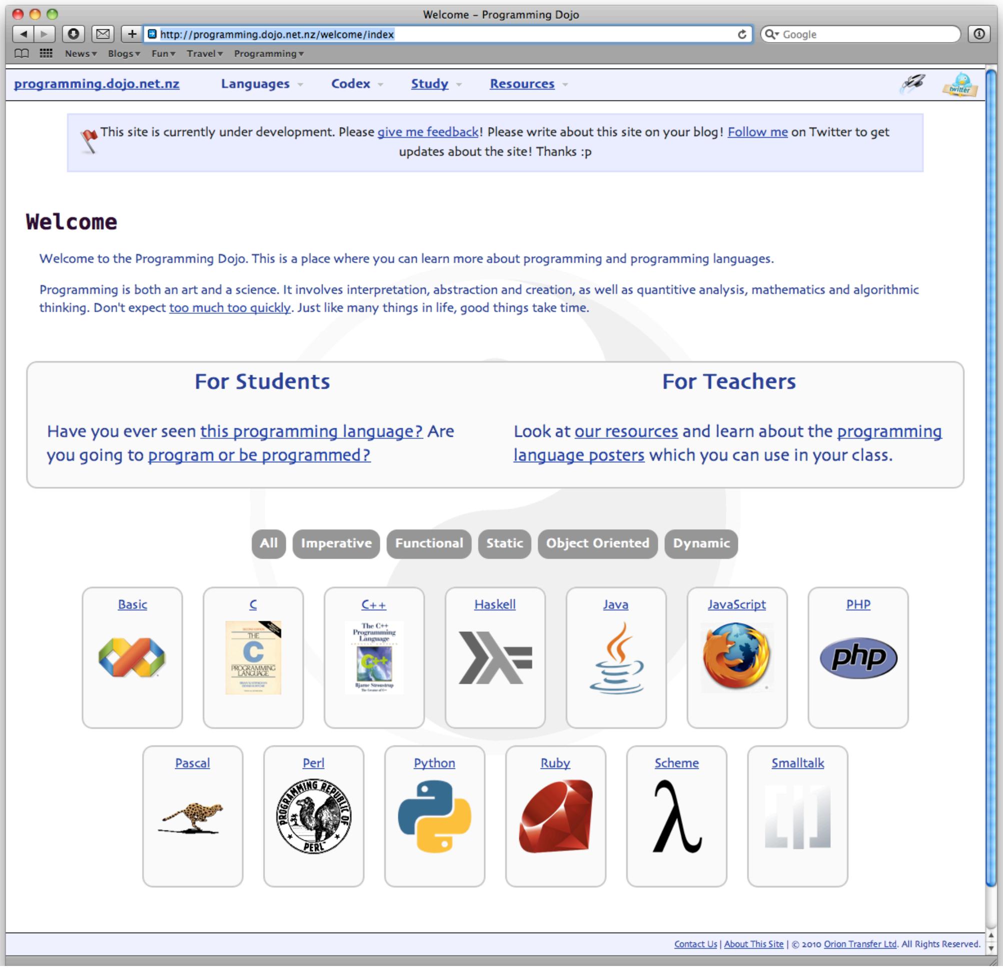
4.1.1 Programming Language Comparison

One page of particular interest on the website is the Programming Language Comparison. This page has a set of questions designed to raise awareness about the major differences between programming languages, and is focused on issues particularly affecting the use of programming languages in education.

Each {question language} pair can be rated as follows:

- Blue - This language is particularly good in this area.
- Green - This language suitably meets the requirements.
- Orange - There are potentially issues that need to be considered.
- Red - There are definitely issues that need to be considered.

The programming language comparison includes every major language discussed on the site.


Welcome – Programming Dojo
http://programming.dojo.net.nz/welcome/index
Google
News Blogs Fun Travel Programming
programming.dojo.net.nz
Languages
Codex
Study
Resources

This site is currently under development. Please give me feedback! Please write about this site on your blog! Follow me on Twitter to get updates about the site! Thanks :p

Welcome

Welcome to the Programming Dojo. This is a place where you can learn more about programming and programming languages.

Programming is both an art and a science. It involves interpretation, abstraction and creation, as well as quantitive analysis, mathematics and algorithmic thinking. Don't expect too much too quickly. Just like many things in life, good things take time.

For Students

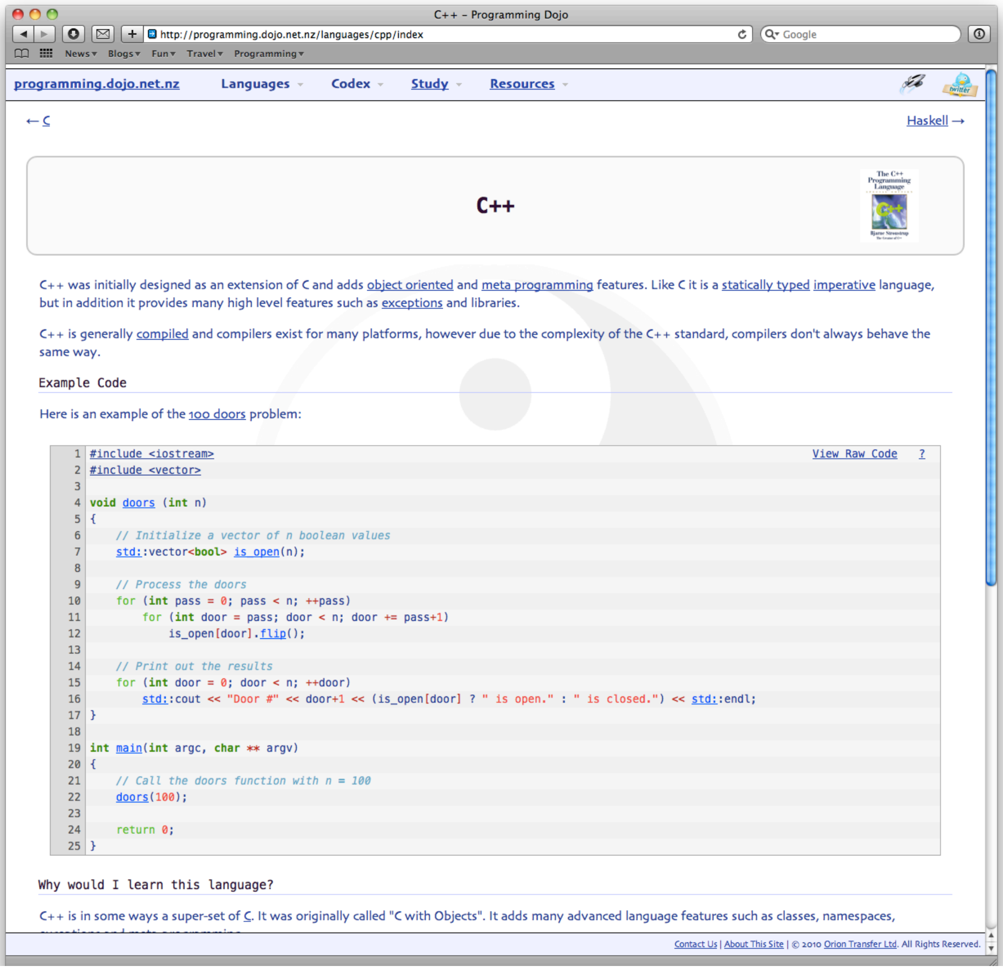
Have you ever seen this programming language? Are you going to program or be programmed?

For Teachers

Look at our resources and learn about the programming language posters which you can use in your class.

Contact Us | About This Site | © 2010 Orion Transfer Ltd. All Rights Reserved.

Figure 1: The front page of the Programming Dojo.



C++

C++ was initially designed as an extension of C and adds object oriented and meta programming features. Like C it is a statically typed imperative language, but in addition it provides many high level features such as exceptions and libraries.

C++ is generally compiled and compilers exist for many platforms, however due to the complexity of the C++ standard, compilers don't always behave the same way.

Example Code

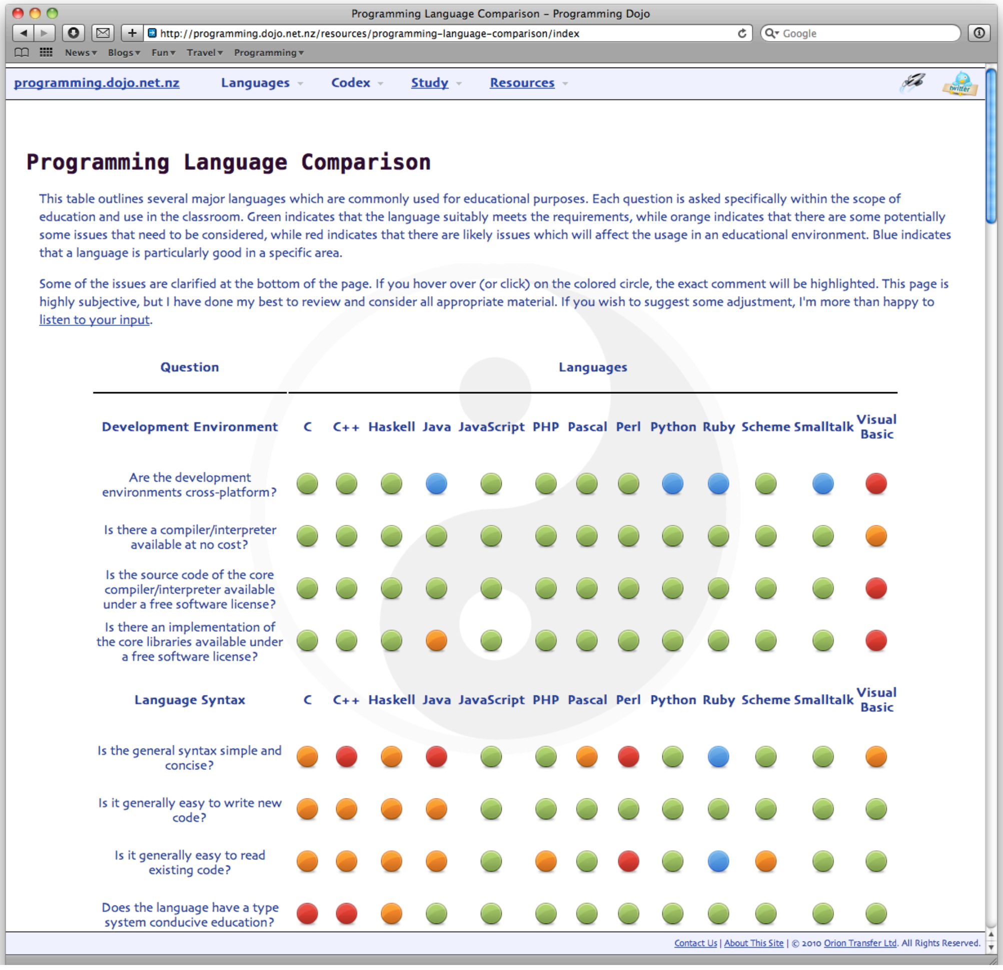
Here is an example of the 100 doors problem:

```
#include <iostream>
#include <vector>

void doors (int n)
{
    // Initialize a vector of n boolean values
    std::vector<bool> is_open(n);

    // Process the doors
    for (int pass = 0; pass < n; ++pass)
        for (int door = pass; door < n; door += pass+1)
            is_open[door].flip();

    // Print out the results
    for (int door = 0; door < n; ++door)
        std::cout << "Door #" << door+1 << (is_open[door] ? " is open." : " is closed.") << std::endl;
}

int main(int argc, char ** argv)
{
    // Call the doors function with n = 100
    doors(100);

    return 0;
}
```

Why would I learn this language?

C++ is in some ways a super-set of C. It was originally called "C with Objects". It adds many advanced language features such as classes, namespaces,

Figure 2: The language page for C++.


Programming Language Comparison – Programming Dojo
http://programming.dojo.net.nz/resources/programming-language-comparison/index
Google
News
Blogs
Fun
Travel
Programming
programming.dojo.net.nz
Languages
Codex
Study
Resources

Programming Language Comparison

This table outlines several major languages which are commonly used for educational purposes. Each question is asked specifically within the scope of education and use in the classroom. Green indicates that the language suitably meets the requirements, while orange indicates that there are some potentially some issues that need to be considered, while red indicates that there are likely issues which will affect the usage in an educational environment. Blue indicates that a language is particularly good in a specific area.

Some of the issues are clarified at the bottom of the page. If you hover over (or click) on the colored circle, the exact comment will be highlighted. This page is highly subjective, but I have done my best to review and consider all appropriate material. If you wish to suggest some adjustment, I'm more than happy to listen to your input.

Question	Languages												
Development Environment	**C**	**C++**	**Haskell**	**Java**	**JavaScript**	**PHP**	**Pascal**	**Perl**	**Python**	**Ruby**	**Scheme**	**Smalltalk**	**Visual Basic**
Are the development environments cross-platform?	🟢	🟢	🟢	🔵	🟢	🟢	🟢	🟢	🔵	🔵	🟢	🔵	🔴
Is there a compiler/interpreter available at no cost?	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟠
Is the source code of the core compiler/interpreter available under a free software license?	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🔴
Is there an implementation of the core libraries available under a free software license?	🟢	🟢	🟢	🟠	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🔴
Language Syntax	**C**	**C++**	**Haskell**	**Java**	**JavaScript**	**PHP**	**Pascal**	**Perl**	**Python**	**Ruby**	**Scheme**	**Smalltalk**	**Visual Basic**
Is the general syntax simple and concise?	🟠	🔴	🟠	🔴	🟢	🟢	🟠	🔴	🟢	🔵	🟢	🟢	🟠
Is it generally easy to write new code?	🟠	🟠	🟠	🟠	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Is it generally easy to read existing code?	🟠	🟠	🟠	🟠	🟢	🟠	🟢	🔴	🟢	🔵	🟠	🟢	🟢
Does the language have a type system conducive education?	🔴	🔴	🟠	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢

Contact Us | About This Site | © 2010 Orion Transfer Ltd. All Rights Reserved.

Figure 3: The language comparison page.

4.2 Posters

Classrooms are an important environment for students to learn. Having a room that is stimulating and exciting encourages learning and ignites curiosity. Using different platforms to stimulate the senses also means that students with different learning needs are included.

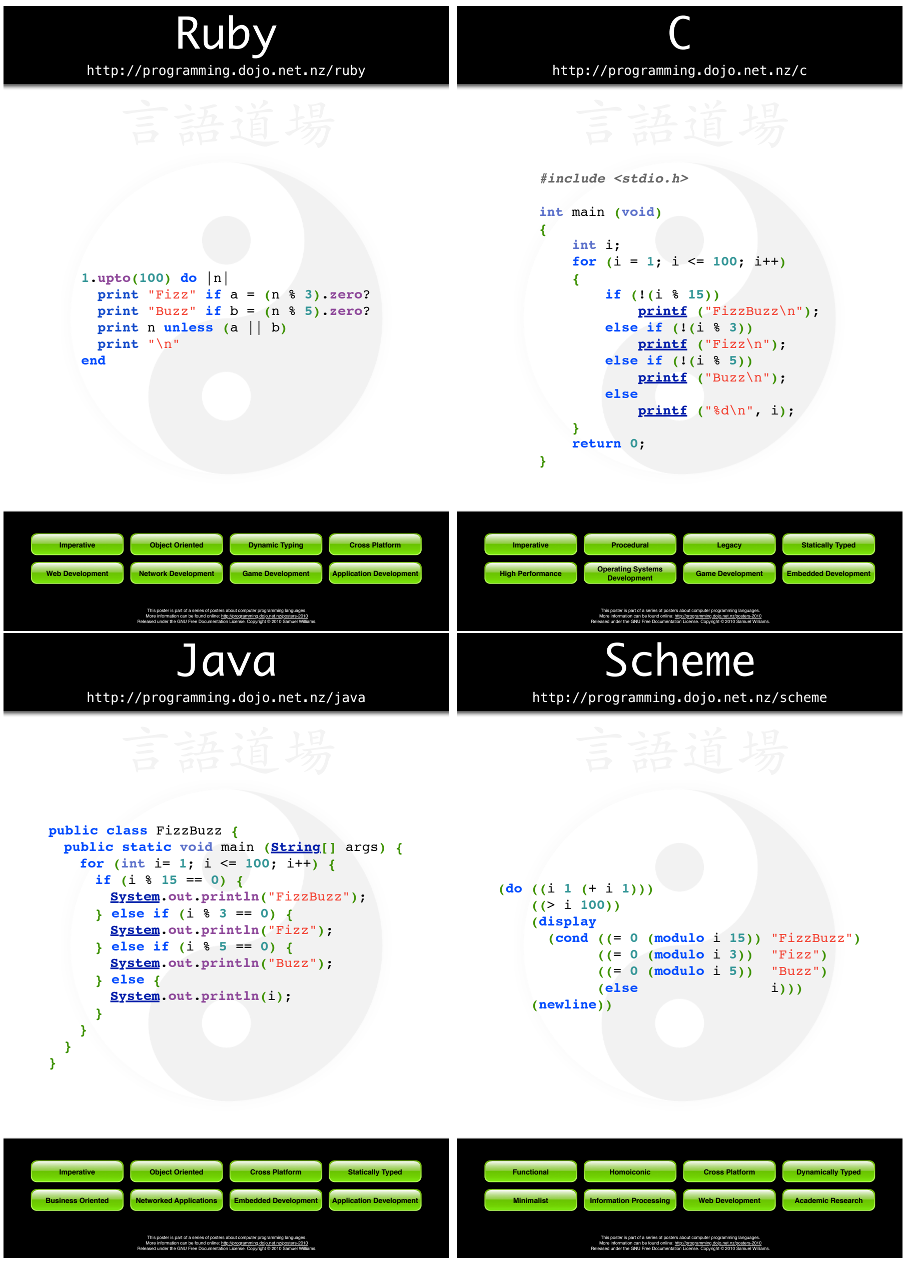
Many existing posters don't focus on computer science specific topics - rather they focus on the broad picture of information technology[16].

I have developed a set of posters (see figure 4) for the computer science class room that are directly related to programming languages. Each poster is an example of a specific programming language which includes a source code example and a list of topical keywords.

These posters are designed with several goals in mind:

- Stimulate discussion about programming languages in the class room.
- Students can look at the posters and see examples of different programming languages.
- The posters can be used as a starting point for further research and inquiry learning.
- The depicted algorithm fizz buzz[17] can be used as a teaching tool for algorithmic thinking.
- Be eye-catching, fun, inspirational and aesthetically pleasing.

The posters are available online[18]. This page discusses how the posters can be used in the classroom as a tool for teaching about computer science and programming languages.



```
1.upto(100) do |n|
  print "Fizz" if a = (n % 3).zero?
  print "Buzz" if b = (n % 5).zero?
  print n unless (a || b)
  print "\n"
end
```

This poster is part of a series of posters about computer programming languages.
More information can be found online: http://programming.dojo.net.nz/posters-2010
Released under the GNU Free Documentation License. Copyright © 2010 Samuel Williams.

```
#include <stdio.h>

int main (void)
{
    int i;
    for (i = 1; i <= 100; i++)
    {
        if (!(i % 15))
            printf ("FizzBuzz\n");
        else if (!(i % 3))
            printf ("Fizz\n");
        else if (!(i % 5))
            printf ("Buzz\n");
        else
            printf ("%d\n", i);
    }
    return 0;
}
```

This poster is part of a series of posters about computer programming languages.
More information can be found online: http://programming.dojo.net.nz/posters-2010
Released under the GNU Free Documentation License. Copyright © 2010 Samuel Williams.

```
public class FizzBuzz {
  public static void main (String[] args) {
    for (int i= 1; i <= 100; i++) {
      if (i % 15 == 0) {
        System.out.println("FizzBuzz");
      } else if (i % 3 == 0) {
        System.out.println("Fizz");
      } else if (i % 5 == 0) {
        System.out.println("Buzz");
      } else {
        System.out.println(i);
      }
    }
  }
}
```

This poster is part of a series of posters about computer programming languages.
More information can be found online: http://programming.dojo.net.nz/posters-2010
Released under the GNU Free Documentation License. Copyright © 2010 Samuel Williams.

```
(do ((i 1 (+ i 1)))
    ((> i 100))
    (display
      (cond ((= 0 (modulo i 15)) "FizzBuzz")
            ((= 0 (modulo i 3))  "Fizz")
            ((= 0 (modulo i 5))  "Buzz")
            (else                i)))
    (newline))
```

This poster is part of a series of posters about computer programming languages.
More information can be found online: http://programming.dojo.net.nz/posters-2010
Released under the GNU Free Documentation License. Copyright © 2010 Samuel Williams.

Figure 4: Several examples of the programming language posters.

5 Results

The initial set of pages about Ruby were well received by the Ruby community with around 800 page views during March. However, there was little interest from the educational community to pursue a full set of learning resources centred around Ruby.

This lead to the development of the 'Programming Dojo' website, which was made public right from the very beginning (early April 2010). Throughout the ongoing design and development process, people have provided critical insight regarding the educational content and structure.

5.1 Raising Awareness

The 'Programming Dojo' was popular on the following sites:

Site	Popularity
Twitter	20+ retweets in 2 days
Reddit	60+ upvotes in 10 hours
DZone	10+ upvotes and hit the front page
Delicious	78 saved links by the 10th of May

The general response was very positive.

5.2 Analysis of Visitors

On May 2nd, 2010, the site was published on several news sites and mailing lists. During the following two weeks, there were over 6000 unique visitors and over 25,000 page views. On average, users viewed 4.13 pages which shows that they were consistently drawn into the site. The most popular languages in order:

Language	Page Views
Python	1611
Haskell	1348
Ruby	1309
C	1119
C++	1072
Java	1014
Scheme	937
PHP	825
JavaScript	822
Smalltalk	729
Perl	680
Basic	680
Pascal	474

During the last month, the posters have been downloaded 239 times. At least one person has written a blog post about the website publicly[19], and several local schools appear to be either reviewing or using the resource, including Christchurch Boys High School.

The initial popularity highlights the value of this resource.

6 Feedback

Almost every page on the 'Programming Dojo' has received useful feedback from multiple sources[1]. This has contributed to the overall quality of the resource, including the following major areas:

- Fixes in descriptive text, including grammar and accuracy of the description.
- Discussion about languages in the comparison chart continues to improve its accuracy.
- Source code examples have been checked for validity, and fixed accordingly.
- Additional links to external resources have been suggested.
- Changes in the high-level structure of the site based on constructive discussion.

6.1 General Feedback

Many positive comments were submitted in various mediums (via the contact page, via comments on the news articles, etc). Here are some specific quotes:

> "I like the way this is unfolding and wanted to suggest you start a serious effort to get people to join a mailing list for updates." - John (who has apparently been following the site for some time)
>
> "I'm a beginner Objective-C developer, I like desktop Mac apps :) Any plans for a ObjC Dojo? :-) Keep up the great work! Greetings from Oslo, Norway." - Storm
>
> "It seems that the C# language is missing from the site. I'd be happy to write it up for you if you'd like." - Charlie
>
> "It looks like you have left python off of the list of functional languages. Since Ruby and Python have similar functional programming features it seems Python should be included as well. Since you mention Python's functional features on the Python page I suspect this omission is just an error. I just thought I'd let you know. Nice site, by the way, it's very informative." - Curtis
>
> "How about OCaml? Functional and OO." - Anonymous
>
> "It looks great. Erlang might be a good addition?" - Chris
>
> "Great site, learned quite a few things from it." - Jeff
>
> "What, no FORTRAN?" - Anonymous
>
> "Thank you so much for this. I'm majoring in Computer Engineering with no programming background and I'm definitely adding this to my bookmarks. :)" - Anonymous
>
> "Thanks SO much - I will look at it this weekend, we need more of this! Will try and get feedback to you on Monday." - Sonya discussing the initial Ruby tutorial.

The following sections discuss particularly important areas of the website and relevant feedback.

[1]To improve privacy only first names have been used in this report, however the majority of these comments were made in public forums.

6.2 Front Page

Initially the front page did not have a list of programming languages. Tim Bell mentioned that the site needed to draw people in, and this lead to the development of the programming language list (with dynamic category selection), and the major ‘For Students’ and ‘For Teachers’ segments.

Several rounds of feedback from various people have improved the content and text on the front page.

6.3 Languages : C

I reviewed the source code on the C page and solicited feedback from FreeNode IRC in channel ‘C’. I received helpful suggestions about improving the source code example.

6.4 Languages : C++

I reviewed the source code on the C++ page and solicited feedback from FreeNode IRC in channel ‘C++’. We had an interesting discussion about using C++ as a programming language in education.

6.5 Languages : Haskell

The original source code example for Haskell did not have comments and was not easily understandable. I also did not have a good understanding of why someone would want to use Haskell.

I contacted the Haskell mailing list for assistance and it was moderately helpful. Kyle discussed the features of Haskell and why it is a useful programming language; he also provided excellent comments for existing Haskell code. Stephen discussed the use of Haskell in an educational context. Several useful links were suggested.

A single point about static typing in the initial discussion formed a new thread about typing and compile time errors.

6.6 Languages : Java

I reviewed the source code on the Java page and refactored it to use more idiomatic structure. I also added comments.

While discussing Ruby, Robert mentioned that Java is not a pure object oriented system, and this was used to improve the content of the Java page.

6.7 Languages : JavaScript

I wrote the code for the JavaScript example. Leon provided several improvements.

6.8 Languages : PHP

Michael wrote the original PHP source code on request, and I subsequently improved it.

6.9 Languages : Pascal

I formatted the source code with comments and checked that it worked. Vilna mentioned that she used to use Pascal successfully, tried Visual Basic which wasn't so successful, and now uses Python.

6.10 Languages : Perl

The initial Perl source code was not well structured so I solicited improvements from the Perl mailing list. The replies were very concise and helpful. Shlomi provided several source code examples which were used on the Perl page, suggested several useful links, and also discussed the features and benefits of Perl.

6.11 Languages : Python

I was interested to know why people would want to use Python.

The Python mailing list was generally very helpful and provided useful contributions. Chris provided a useful list of reasons supporting the use of Python, and also provided a comprehensive review of the programming language comparison. André provided a very useful link to the Python EDU-SIG page[5].

One thread which asked for general feedback eventually became a discussion about indentation.

6.12 Languages : Ruby

The ruby-lang mailing list was asked to provide feedback on the website. They checked code and provided enhancements. They commented on the language comparison and offered useful feedback. Josh in particular provided a comprehensive review of several languages in the comparison chart.

Robert corrected several omissions in the language descriptions (notably Java) and offered suggestions about how they could be fixed. Rick discussed the nature of compiled vs interpreted languages and how this classification is a feature of the implementation rather than the language.

The Rails Bridge mailing list provided lots of positive feedback including several suggestions for improvements to the code. Jeff raised the issue that the language pages themselves did not really answer important questions such as 'Why would I pick this language?' or 'Is it popular?'. He also mentioned that the terminology might be too complex for beginners. This feedback prompted the addition of a 'Why would I learn this language?' section to every language page.

The thread was eventually hijacked by someone talking about another site about Ruby.

6.13 Languages : Scheme

The Scheme mailing provided fairly useful feedback. The majority of discussion was highly academic, with several people contributing somewhat offensive feedback.

Matthias asserted that teaching any real programming language to students is not suitable, and provided links to several research papers. Shriram provided some useful ideas relating to terminology.

Stephen was incredibly helpful and provided examples in both Haskell and Scheme.

6.14 Languages : Smalltalk

The Smalltalk community was generally very helpful and provided many useful links. David and John both provided some example code. Lawson mentioned that one of the main benefits of Smalltalk is the ‘liveness’ of its objects, and provided some interesting comments on Smalltalk’s major features.

6.15 Resources : Programming Language Posters

No feedback has been received about the posters. I’m not aware of their use at this time.

6.16 Resources : Programming Language Comparison

Throughout the construction of the site, feedback has been received about the programming language comparison chart. In every case, this feedback has been reviewed seriously with the goal of providing a balanced comparison.

Sean (helpfully) complained that the language comparison chart did not include all the languages that were available at the site. This prompted the addition of PHP and JavaScript

Several people mentioned that Python was a functional programming language, however after discussing this on the python mailing list, the general consensus about Python was that it is not a functional language, but did have several functional features such as lambdas, list comprehensions, etc. This information was used to update the programming language comparison.

After incorporating feedback from several groups, I decided to add the ‘Blue’ classification, which indicates a particularly remarkable feature about a language.

7 Discussion

Overall the site has been well received by the online community. Programming languages are a relatively niche topic, but the results show that many people are interested.

The front page was highly successful at drawing people into the site, so developing that part of the site was time well spent. The structure of the site has meant that many people find it a useful starting point, and thus it has been bookmarked by many people as a resource they might refer to in the future.

The level of information seems to be about right. No one has complained that there wasn't enough information for individual languages. Conversely, no one has complained that there was too much. This seems to indicate that the right balance between technical detail and informal prose has been found.

Several people have requested additional languages. Care needs to be taken to ensure that the site remains relevant, and one of the initial goals was to provide a broad scope for existing methodologies and paradigms. Languages which are similar to existing languages potentially enlarge the site without adding much value.

Due to the inherent nature of the posters being physical objects, more time and feedback is required to assess the usefulness of this component of the resource.

I have enjoyed the project and look forward to developing it further.

8 Conclusion

The website and posters were produced and provide a unique resource for use by both teachers and students for learning more about programming languages. Initial feedback and usage has been positive. Constructive comments were taken seriously and used to modify and improve the site. Over time, increased awareness of programming languages will hopefully enhance computer science education.

8.1 Future Plans

There are several things that will continue to be incrementally improved:

- Additional languages that fit within the goals of the site.
- A new section per page which discusses similar/derived programming languages.
- Additional examples of source code could be added for better coverage of language features.
- Additional external links and resources as they are found/recommended/created.
- A larger set of posters with improved design/structure.

There are several diversifications which are potential avenues for development:

- Programming language competitions where the focus is on a wide coverage of solutions in different programming languages. This could be done by having a major award category for each programming language.

- A discussion forum for people interested in discussing programming languages and asking questions about high-level features of programming languages.
- Some kind of computer programming club where the focus is not on any specific language or platform, but inclusive of many different approaches and environments.

References

[1] Peter J. Denning. Is computer science science? *Commun. ACM*, 48(4):27–31, 2005. Available from: `http://cs.gmu.edu/cne/pjd/PUBS/CACMcols/cacmApr05.pdf`.

[2] Maurice Merleau-Ponty. *Phenomenology of Perception.* 1945.

[3] Wikipedia: List of programming languages. [Online; accessed Apr-2010]. Available from: `http://en.wikipedia.org/wiki/List_of_programming_languages`.

[4] RosettaCode. [Online; accessed Apr-2010]. Available from: `http://rosettacode.org`.

[5] EDU-SIG: Python in education. [Online; accessed May-2010]. Available from: `http://www.python.org/community/sigs/current/edu-sig/`.

[6] The TeachScheme / ReachJava project. [Online; accessed May-2010]. Available from: `http://www.teach-scheme.org/`.

[7] squeakland : home of squeak etoys. [Online; accessed May-2010]. Available from: `http://www.squeakland.org/`.

[8] Popular programming languages. [Online; accessed May-2010]. Available from: `http://www.scriptol.com/programming/choose.php`.

[9] Programming language comparison. [Online; accessed May-2010]. Available from: `http://www.jvoegele.com/software/langcomp.html`.

[10] Orion Transfer Ltd. [Online; accessed May-2010]. Available from: `http://www.oriontransfer.co.nz`.

[11] Programming Dojo. [Online; accessed May-2010]. Available from: `http://programming.dojo.net.nz/`.

[12] Utopia. [Online; accessed May-2010]. Available from: `http://www.oriontransfer.co.nz/software/utopia`.

[13] Git - fast version control system. [Online; accessed May-2010]. Available from: `http://git-scm.com/`.

[14] jQuery.Syntax. [Online; accessed May-2010]. Available from: `http://www.oriontransfer.co.nz/software/jquery-syntax`.

[15] Quicksand. [Online; accessed May-2010]. Available from: `http://razorjack.net/quicksand/`.

[16] CS Education Week: Posters, brochures and kits. Available from: `http://www.csedweek.org/resources/posters-and-brochures/`.

[17] Fizz buzz. [Online; accessed Apr-2010]. Available from: `http://en.wikipedia.org/wiki/Bizz_buzz`.

[18] Programming Dojo: Posters. [Online; accessed May-2010]. Available from: `http://programming.dojo.net.nz/posters`.

[19] Programming dojo: Programmiersprachen im blick. [Online; accessed May-2010]. Available from: `http://matthiasschuetz.com/programming-dojo-programmiersprachen-im-blick`.