Graph-Based Learning

Jason Gronn
jgronn@bgsu.edu

Follow this and additional works at: https://scholarworks.bgsu.edu/honorsprojects

Part of the Educational Technology Commons, and the Other Computer Sciences Commons

How does access to this work benefit you? Let us know!

Repository Citation
Gronn, Jason, "Graph-Based Learning" (2024). Honors Projects. 992.
https://scholarworks.bgsu.edu/honorsprojects/992

This work is brought to you for free and open access by the Student Scholarship at ScholarWorks@BGSU. It has been accepted for inclusion in Honors Projects by an authorized administrator of ScholarWorks@BGSU.
Graph-Based Learning

By Jason Gronn
I. Table of Contents

Sections

Graph-Based Learning................................................................................................................1

I. Table of Contents...................................................................................................................2

II. Introduction............................................................................................................................2

III. Project Motivation................................................................................................................4

IV. Literature Review................................................................................................................6

V. Methodology............................................................................................................................9
   A. Category Selection................................................................................................................9
   B. Website Construction.........................................................................................................12

VI. Built Prototype....................................................................................................................14

VII. Discussion of Results and Implications for Future Research........................................18

VIII. Conclusion.........................................................................................................................21

IX. References............................................................................................................................22

Figures

Figure 1 - Available Topics View.............................................................................................14
Figure 2 - Individual Topic View.............................................................................................15

Tables

Table 1 - Course Orders and Hypothetical Course Order.......................................................11
Table 2 - Example YAML Metadata File................................................................................16
Table 3 - Example Markdown Contents File............................................................................16

II. Introduction

With the advent of the internet, there is a wealth of convenient and accessible knowledge online about a variety of subjects [1][2]. These resources are oftentimes critical to people who would like to learn a new skill without taking a formal class. Many people are interested in a specific topic in a field - “How can I do X?”. However, despite how quickly these topics can be accessed now, sometimes these topics may be complex and require the learner to have knowledge about other topics already or have prerequisite skills. As such, upon finding and viewing a tutorial for their topic of interest, the learner may find that they lack the prerequisite
skills (i.e., a tutorial mentions a subject that they know nothing about), and as such, they are left more confused by the tutorial than when they started.

To further the problem, the user may not know what prior knowledge or skills they are missing. For example, if you were to try to learn something on YouTube [2], you might find something such as a short tutorial or a long talk. However, you might find that the tutorials may assume you know information that you do not already know.

On the other hand, there will be some knowledge or skills that the learner does already know, or that are not relevant to the learner, thus wasting the learner’s time. As an example of this, take CodeAcademy. Specifically, take the content about designing databases with PostgreSQL [1]. Say a learner has experience with another database already, such as SQLite. In this case, the learner may find some of the information redundant. Looking at the course syllabus page [1], one can see that the course includes some of the most basic information about SQL, such as defining what a database is and how to construct some basic queries. The learner, who is more interested in the PostgreSQL specific features related to queries, will not need the primer about databases because they already know the content. However, since concerns are mixed in with each other (e.g., PostgreSQL installation instructions and basic database definitions), the learner must parse through the undesired concern to get to the concern of interest. CodeAcademy estimates 30 hours for the course, which could be drastically cut down if the learner could focus only on PostgreSQL-specific aspects.

As such, the problem here appears to be an organization issue - the user does not know what they do not know, yet learning resources often make assumptions about presence or lack thereof prior knowledge.
The goal of this project is to develop a learning platform that addresses these issues. This project should enable a learner to learn a given topic despite the learner’s potential knowledge gaps. It also should be conducive towards effective learning, so that learners may be able to retain and apply the knowledge that they learn.

- **RQ1:** How can we allow people to learn a specific skill on the web, keeping in mind that there may be gaps in their prior knowledge?
- **RQ2:** How do we ensure that people remain engaged, retain knowledge, and are able to apply that knowledge?

Section I of this paper is the table of contents, and covers which pages each section, figure, or table is in. Section II is this introduction, which explains the basic background of this project, questions it poses, and briefly explains the rest of the paper. In Section III, this paper will explain why this project is necessary. In Section IV, this paper will observe relevant literature that relates to the field which could impact or improve the design of this project. In Section V, this paper will explain how this project was carried out. In Section VI, this paper will explain the final prototype used to address the posed research questions in a tangible manner. In Section VII, this paper will discuss some observations that have been drawn from the result and how the project would be further improved upon or affects other projects. Section VIII wraps up this paper, summarizing everything that has been covered. Finally, Section IX provides the source of each reference used within this paper.

### III. Project Motivation

In physical settings, one way used to ensure that students have minimal gaps in their knowledge is via the usage of prerequisite courses. Cameranesi et al. aims to show the advantages of the usage of prerequisites before a course. They opted to experimentally
demonstrate this by means of tracking what order of courses students take, as well as their final grades and exam scores, and then reconciling the gathered data into a series of graphs [3]. They then provided visuals indicating the typical approach of students who performed well during the experiment, and students who did not perform well. The visuals demonstrate that poorly performing students attempted to learn many things concurrently, while well performing students did learn some things concurrently, but heavily followed a sequence, in order to make sure they did not lack prior understanding. It should be noted, however, that they also state that they do not distinguish between which factor (good performance or order of courses) was the causation, and which one was the effect.

Furthermore, Allan and Kolesar suggest that, for students learning computer science, directly beginning to learn programming without fully understanding the core computer science concepts behind it will lead to an increased chance of failure [4]. This seems to further indicate a need for clear prerequisites when attempting to learn a given topic of computer science. It indicates that the teaching of a concept should be separate from the teaching of a programming language construct relating to the concept, so that learners may first understand it at a more fundamental level.

Many people are learning because they have a usage for the content that they are attempting to learn. Rather than learning a large amount of material in advance, a learner may choose to just find relevant info as it is needed. This can be called Just-In-Time learning. Brandenburg and Ellinger published a paper including a section which mentions that people learn in response to need, and that they may ignore or not fully understand topics being taught should they not have an immediate need for that info [5]. On the other hand, people learn effectively and quickly when they do have an immediate need for info. Dale and Andrea also speculate that there
is demand for Just-In-Time learning. From this, it seems that some people may wish for a learning resource in which they learn exactly what they need to learn at the moment, and nothing more. Standard online courses and e-books, such as some of the ones that this paper will refer to, often interlace varying topics and as such do not fulfill this need.

The above demonstrates a need for a learning system that ensures students have learned all prior prerequisites of a course before taking a course, as to avoid confusion, but is also arranged optimally as to enable learners to perform Just-In-Time learning. This project attempts to tackle said need.

IV. Literature Review

As will be discussed in the methodology section, in order to base construction of the project with the knowledge of existing resources, five different existing courses and e-books - the BGSU CS 2010 class [6], the Khan Academy Javascript Tutorial [7], the “Learn Ruby the Hard Way” e-book [8], the “Golang Book” [9] and MDN’s Javascript Tutorial [10] were considered. This allows for a good reference on how existing high-quality learning resources are structured, which can be useful for ensuring the new system to be constructed will allow learners to learn the most important topics, and in an ideal order.

In addition to this, a variety of additional resources relating towards the fields of learning, retention, and organization of knowledge were used to inform this project.

As mentioned previously, it is possible to arrange prerequisites into a graph structure. When representing prerequisites this way, it could be ideal to arrange them in such a way that they are easy to navigate between. Palën has “conducted a systematic literature review on about 20 years of publications of two major human-computer interaction journals” [11] in order to discover what types of graph-based user interface designs work well. One of Palen’s findings
was that, using a combination of techniques called semantic zooming and hierarchically clustered graphs, it is possible to group many nodes into a single large node, which the user can then see in a “zoomed-out” large-scale view [11]. This approach could potentially be used to make a graph-based UI more structured, less clutter-some, and simpler for a user to navigate, which will be useful for making the prerequisite-graph concept more navigable.

Since this project will focus on teaching computer science concepts, it will also become necessary to ensure that computer science is taught effectively through these topics. Allan and Kolesar, as briefly mentioned in a prior section of this paper, suggest that jumping directly into programming can lead to an increased chance of failure when learning computer science. They also claim that teaching fundamental concepts before any syntax is discussed will ensure that learners understand the actual concepts behind computer science. Their research on the topic was done via observational analysis - they observed the success of students in a computer science course and how it relates to the amount of computer experience those students had prior to taking the course, and also observed the strategies typically used by more successful students. Based on these observations, they created a “CS0” course that taught concepts prior to programming courses, and observed that this course was effective in practice. They also noted that these prior activities resulted in students being very enthusiastic [4]. This will be used to inform the structuring of course content, ensuring that learners will learn the fundamental concepts before learning language-specific syntax.

Mani and Mazumder performed a study in which both student confidence in a CS-related area as well as student performance in that area were related to each other, for a total of six CS-related areas. Notably, by doing this research, they produced a total of six tables that indicate both student performance (did the student struggle in this area) as well as student confidence vs
performance (did this student do as well in this area as they thought they did) [12]. This research is useful because it pinpoints areas of Computer Science that students should be encouraged to especially focus on. It could be used to determine what topics to put emphasis into and to allocate more resources to ensuring the quality of.

Since this project involves teaching students, it may also be worthwhile to look into the field of metacognition - which focuses on how the human mind works - and see what research in this field has been done towards improving learning. Hedberg et al. describes a tool that uses concepts from the field of metacognition to help learners understand and retain knowledge. They suggested incorporating the following concepts into learning-centered software tools: simulation-based learning (learning by doing), incidental learning, learning via reflection, case-based teaching, and learning by exploring. They have done observations of a tool utilizing such concepts, and found that it does have promise for improving student learning. However, they noted that it took time for students to become familiar with such a tool, as students were initially not familiar with the complexity of such a tool [13]. This seems to suggest that, while these approaches are useful for improving student learning, additional effort must be made towards keeping the approach intuitive to use. However, if it can be incorporated into a learning tool in an intuitive way, it could be very effective.

One subfield of metacognition is metamemory. In a paper written by Schwartz and Efklides, it is found that often learners expect that they will remember more than they actually remember and expect that they will forget less than they actually forget [14]. Because of this, it cannot be guaranteed that the learner has actually remembered information that they believe they have remembered, and at a later date they may find themselves struggling to recall said
information. This could indicate the need for encouraging a user to revisit a topic in the project that will be prototyped.

One topic of interest in the field of metamemory is a concept called overlearning, which suggests that a person can repeat a learning trial a certain number of extra times in order to better retain that knowledge. The exact number of extra trials is a percentage of however many trials of learning the topic it takes to initially get good at that topic. Driskell et al. suggest that one experiment found that overlearning could reduce mistakes made after an 8-week break from a topic by as much as 65% [15]. This seems to suggest that, in order to help students learn optimally, they should be encouraged to continue practicing a topic for an extended period of time after they feel that they have learned said topic.

V. Methodology

A. Category Selection

This project started out by looking into existing research into related fields. This research helped to inform the design of the project. After this, research went into determining what some of the most important topics to teach CS learners first are, and what order they are typically taught in. Then, a tangible prototype was constructed, and tangible content for that prototype was created.

To determine which topics are most important and in which order those topics would generally be taught in, an observation of existing online courses and e-books was conducted. In general, these courses and e-books were selected based on “authoritativeness” - would this look like something trustworthy to a user attempting to learn a programming language?

Exact reasons for the selection of each course/e-book selected for reference are listed below:
• BGSU CS 2010 [6] - Represents the topics that a learner would learn in a formal education setting.

• Khan Academy [7] and MDN [10] - These are high-quality introductory courses that a learner would likely find if they were to look up a tutorial. Their teaching is effective and they cover a satisfactory range of topics.

• Learn Ruby the Hard Way [8] and Golang Book [9] - These are easy-to-find resources that a learner would likely decide to use if they wished to learn a specific programming language (Ruby and Golang, respectively).

After reading through each course, the next step was to arrange a list of general topics that they appeared to cover, in the chronological order of which topics were significantly discussed first. This is shown in the table below. After eliminating a sixth course that appeared to cover an outlier number of topics (not pictured in table), each topic covered was color-coded, to make repetitions of topics easier to view. From the table, topics which were covered in half the courses (three courses) or more were then brought over to the final result column as a topic that would be particularly good to focus on. The final results column is sorted by the order in which topics generally appeared within the courses first.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob. Solv</td>
<td>Pep Talk</td>
<td>env setup</td>
<td>env setup</td>
<td>language bas.</td>
<td>env setup</td>
</tr>
<tr>
<td>Alg. Design</td>
<td>Course Info</td>
<td>hello world</td>
<td>hello world</td>
<td>env setup</td>
<td>comments</td>
</tr>
<tr>
<td>flow charts</td>
<td>Live Demo</td>
<td>comments</td>
<td>comments</td>
<td>comments</td>
<td>types</td>
</tr>
<tr>
<td>pseudocode</td>
<td>Document.</td>
<td>math ops</td>
<td>types</td>
<td>prob. solving</td>
<td>variables/mut</td>
</tr>
<tr>
<td>types</td>
<td>variables/mut</td>
<td>variables/mut</td>
<td>string manip.</td>
<td>example</td>
<td>string manip.</td>
</tr>
<tr>
<td>variables/mut</td>
<td>string manip.</td>
<td>string manip.</td>
<td>variables/mut</td>
<td>variables/mut</td>
<td>logical ops</td>
</tr>
<tr>
<td>expressions</td>
<td>functions</td>
<td>print format</td>
<td>scope</td>
<td>types</td>
<td>conditionals</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>--------------</td>
<td>-------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td>order of prec</td>
<td>conditionals</td>
<td>multiline str</td>
<td>constants</td>
<td>constants</td>
<td>loops</td>
</tr>
<tr>
<td>casting</td>
<td>debugging</td>
<td>prompts</td>
<td>conditionals</td>
<td>subtypes</td>
<td>debugging</td>
</tr>
<tr>
<td>testing</td>
<td>loops</td>
<td>parameters</td>
<td>loops</td>
<td>casting</td>
<td>coding style</td>
</tr>
<tr>
<td>debugging</td>
<td>coding style</td>
<td>file IO</td>
<td>arrays</td>
<td>math ops</td>
<td>functions</td>
</tr>
<tr>
<td>coding style</td>
<td>arrays</td>
<td>functions</td>
<td>slices</td>
<td>order of prec.</td>
<td>arrays</td>
</tr>
<tr>
<td>logical ops</td>
<td>OOP</td>
<td>logical ops</td>
<td>maps</td>
<td>comp. ops</td>
<td>OOP</td>
</tr>
<tr>
<td>conditionals</td>
<td>pseudocode</td>
<td>conditionals</td>
<td>functions</td>
<td>string manip.</td>
<td></td>
</tr>
<tr>
<td>loops</td>
<td>arrays</td>
<td>recursion</td>
<td>arrays</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO</td>
<td>loops</td>
<td>panic</td>
<td>conditionals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>functions</td>
<td>coding style</td>
<td>pointers</td>
<td>logical ops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recursions</td>
<td>debugging</td>
<td>garbage col</td>
<td>loops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arrays</td>
<td>maps</td>
<td>OOP</td>
<td>map/filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sorting</td>
<td>OOP</td>
<td>concurrency</td>
<td>functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>composition</td>
<td>packages</td>
<td>events</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>testing</td>
<td>OOP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>json</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>async</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1 - Course Orders and Hypothetical Course Order**

This data was used in order to inform which topics should be focused on first, and to get a better idea of what order prerequisites would typically occur in. This allows for optimal structuring of the graph and ensuring the capability to learn basic topics is granted to the learner before more complex topics become available.
B. Website Construction

After this, a demonstration website was made. The purpose of this website is for evaluating how effective the chosen approach to RQ1 would be - what some merits and demerits to the approach might be, allowing for qualitative analysis through a tangible prototype. Because this website is a prototype, it was constructed using Ruby On Rails, technology commonly used for server-side programming and beneficial for both its ease of setup and its easy readability. In order to minimize the amount of redundant effort that went into building the prototype and to create the prototype in a reasonable timeframe, a variety of existing libraries were used to implement certain features.

To allow for simple but effective formatting of content in an easy-to-digest manner, lesson content is created using markdown (reference table 3 for an example of markdown). Markdown allows for writing styled text in the form of a file consisting largely of plain text. However, some text may be surrounded or prefixed by special symbols indicating formatting - for example, bolded text may be surrounded by an apostrophe at each side. This allows for adding formatting to text while being simple to integrate on the programming side (there is no need to develop a custom content formatting system). In order to integrate markdown with the website, a library called Redcarpet was used. Redcarpet parses markdown and converts it to a format that web browsers are capable of displaying.

Since the focus of the topics in this project are computer-science related and largely involve programming samples, the Pygments Syntax Highlighter was used to colorize code and make code samples easier to digest. By default, the markdown formatter mentioned in the previous paragraph will emit code blocks that are a single color. Code is easier to read, however, whenever common keywords or data types are differently colored, allowing the reader to
distinguish the intent of parts of code at a glance. The Pygments Syntax Highlighter colorizes the code samples in order to achieve this benefit.

Because this project largely revolves around prerequisite graphs, it is also necessary to provide a way to show those graphs. For this use case, d3.js was an effective client side library for making the graphs. It allows for dynamic creation of graphs, and allows said graphs to be interactive, which is necessary for the user to be able to click on the graph and go to the topics on the graph quickly. The interactivity is part of the reason d3.js was chosen over a server-side graph library - the server serves static content, and it is necessary to use client-side code to allow interactivity. However, because d3.js still needs to know the layout of the topic graph, w

Overall, the stack was chosen to enable speedy development while still providing support for necessary features.

Another approach used to ensure speedy development of the prototype was to make all content static. In other words, it is not possible to add additional content through the usage of the prototype’s user interface itself. Instead, a feature of the stack that allows for creation of a database based off of elements on the host machine was used. This meant, instead of needing to set up, create necessary routes for, and maintain a database, it was possible to run a single command to update the contents of the website. The disadvantages of this approach are that adding new content requires direct access to the host machine, and that the website must be restarted whenever new content is available. However, for a proof-of-concept prototype, this appears to be permissible.
VI. Built Prototype

This project has led to the creation of a working website designed to teach topics to those wishing to learn the basics of programming in the domain of Computer Science. The website is split into two main pages - the available topics overview, and the individual topic view.

The project assumes that one topic may be related to another topic in one of two ways: it may be the “prerequisite” of that topic, or it may be the “next steps” of that topic. A “prerequisite” is considered to be a topic in which the learner should attempt to learn before the topic they wish to learn, lest the learner end up being confused. A “next steps” topic is a topic in which one of its prerequisites is the topic that it relates to. The intent of a “prerequisite” is to allow the learner to fill in knowledge gaps before learning the topic that they wish to learn, while the intent of “next steps” is to ensure a user knows what options for learning are still available upon finishing a topic of their choice, allowing them to stay engaged in the learning process.

The available topics view (see Figure 1) contains two main features: an unordered list of topics that are able to be learned, and a graph showing a visual representation of which topics may be considered the “prerequisite” or “next steps” of another topic. Both the unordered list of topics and the visual representation are interactive, allowing the user to click on a topic they wish to learn. In the visual representation, each topic is a node with a title. There are edges between some nodes, which indicate that the two nodes connected by the edge are in a “prerequisite” or “next steps” relationship with each other. There is no particular order in which nodes are placed on the page.
Figure 1 - Available Topics View

The individual topic view (Figure 2) shows four primary sections - a header including the topic title and a navigation link to the available topics overview, a sidebar tile consisting of links to each prerequisite topic, a sidebar tile consisting of links to each next step, and a content pane containing the information to be taught to the learner. The content pane includes support for syntax-highlighted code blocks and markdown-formatted lesson content.
Gronn 16

**Introduction to conditional operators**  
[Back to Topics List]

**Lesson**  
It is time for conditional operators to make the stage! Conditional operators allow for testing how values compare to each other.

In many languages, the set of conditional operators looks something like this:

**Logical Conditional Operators**

<table>
<thead>
<tr>
<th>Op.</th>
<th>Intent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
<td>True if both of two booleans are true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>equals</td>
<td>True if two values are equal to each other</td>
</tr>
<tr>
<td>!=</td>
<td>not equals</td>
<td>True if two values are not equal to each other</td>
</tr>
</tbody>
</table>

**Mathematical Conditional Operators**

<table>
<thead>
<tr>
<th>Op.</th>
<th>Intent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>True if the left number exceeds the right number</td>
</tr>
</tbody>
</table>

In the cases where these operators are not true, they are false.

These operators can vary per language. For example, in Lua `==` is used instead of `===`, and in JavaScript `===` is preferred over `==`. Additionally, some languages (such as Lua or Python) directly use the name of some operators (or, and, etc) instead of their symbols.

We can use some value tables to learn what each operator does. A value table tells you the proper output value given a logical conditional operator and the values that the operator is being used to compare.

```
& & (AND)
```

```
<table>
<thead>
<tr>
<th>x</th>
<th>false</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
```

Given two values (say `bool1` is `false` and `bool2` is `true`) and an operation (say `bool1 & bool2`), we can look at the table corresponding to that operation. Since `bool1` is `false`, we first navigate to the column labeled `false`. Next, since `bool2` is `true`, we go down that column until we reach the row labeled `true`. Then we see which cell we land on - `false` in this case. This indicates that, with our given values and our `bool1 & bool2` operation, the result is `false`.

**Figure 2 - Individual Topic View**

Each lesson is created using two files - a markdown file, and a yaml file. The markdown file is used for creating lesson content. Markdown was selected because it allows for easily creating an appealing layout of text that will help the reader understand what each part of the text represents. It supports various standard markdown features, including bold text, tables, and code block. Code blocks and tables are highlighted a distinct color in order to separate them from normal text. The yaml file contains metadata pertaining to each lesson - the title of the lesson,
and the prerequisites of the lesson. The “next steps” of a topic are able to be auto-generated given the prerequisites of each topic. Visual graphs and unordered lists/link trees will automatically be updated when lesson content is changed and the relevant command is run.

Shown below is an example yaml file used to hold lesson metadata.

```yaml
title: "Booleans and Conditional Operators in Java"
prereqs:
- "java_bootstrap"
- "conditional_operators"
```

**Table 2 - Example YAML Metadata File**

Shown below is an example markdown file used to hold lesson content. This example includes a table, a bolded header for that table, and a description of the table. It also contains escaped characters.

```
<table>
<thead>
<tr>
<th>Op.</th>
<th>Intent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
<td>True if both of two booleans are true</td>
</tr>
<tr>
<td>&amp;#124;&amp;#124;</td>
<td>OR</td>
<td>True if least one of two booleans are true</td>
</tr>
<tr>
<td>==</td>
<td>equals</td>
<td>True if two values are equal to each other</td>
</tr>
<tr>
<td>!=</td>
<td>not equals</td>
<td>True if two values are not equal to each other</td>
</tr>
</tbody>
</table>
```

**Table 3 - Example Markdown Contents File**

When creating content for the website, based on the research done by Allan and Kolesar, the topics were designed such that a general concept (such as the fundamentals of how logical operators work) was taught before an actual programming implementation (such as how to use logical operators within Java) was taught. This should ensure that students will have a solid grasp on why they are doing, as well as allow them to think independently of a specific programming language.
VII. Discussion of Results and Implications for Future Research

The construction of this site has presented varying pros and cons of this method of teaching, as well as the overall design of this approach. Based on these findings, there is a variety of future work that can be done to iterate and improve upon the solution.

One major issue revealed by this implementation of a solution to RQ1 was how unwieldy the graph is to follow. Even with as little as eleven topics, the graph already appeared rather large and cluttered. Many edges of the graph intersect with one another, and some edges that appear to connect two nodes are actually edges connecting two further nodes that end up passing under an unconnected node. The issue with this is that such a large and interweaving graph could potentially scare away potential learners. To further the issue, the current implementation of the graph does not distinguish between “prerequisites” and “next steps” in any meaningful way, which means that the user may not know which node they should click, and therefore end up confused.

Based on this, it will be necessary to further look into ways to ensure that the graph remains easy to understand and navigate. In its current form, there are some possible ways to improve this. For example, in the current form, one cannot distinguish between prerequisites and next steps when only observing the visual graph. Given more time, arrows indicating next steps and dots representing prerequisites would be added to the ends of each edge. To allow easier tracking of relationships between nodes, the edges should be color-coded. Furthermore, it is also necessary to improve the algorithms used for selecting node positions such that they would consider the relevance of topics to each other and minimize crossing of edges. Another improvement would be to minimize the impact of nodes that are not currently relevant to the user. One way to do this is to take note of Aksele Palén’s findings mentioned earlier, and group
together topic nodes into larger conglomerate nodes [11]. When zoomed out, the website will only show the name of the general subject of the conglomerate node. Upon zooming in, however, the user will be able to view all topics relating to a given subject.

Another issue that became apparent is the resource cost of maintaining quality information. Making a new topic takes a fair amount of time, and requires that the maintainer has expertise in that topic. Furthermore, it is hard to verify the quality of the writing about the topic as it is being written. This is expected to only become harder as the features of the website are extended (e.g. if support for quizzes or simulations were to be added, this would also entail writing quality quizzes and simulations). Many other resources may already have higher-quality writing that other learners may prefer. As such, breadth, depth, and quality of content are all issues that would need to be resolved when attempting to use this approach.

The above suggests that making content from scratch is not necessarily optimal. Despite the cost of not having full control over content and how content integrates with other content, it should be attempted to instead serve a resource that organizes references to other contents. What this means is that selecting a node on the graph, a next step, or a prerequisite would lead to a general description of a topic and a link to a third-party resource covering said topic, as opposed to self-hosted content related to that topic. As such, the project would serve more as a pure organization hub for learning resources and less as a standalone learning resource. It should be observed what advantages and disadvantages said approach would have.

One thing that did go well was the sidebar, indicating next steps and prerequisites. Notably, one of the topics had a total of 5 prerequisites if you recurse down the tree, meaning a learner would need to learn a total of 6 topics. This is notably smaller than the 11 topics total covered at time of writing. A more linear course would require completing all 11 topics, while a
free-format course might not make you aware of the 5 prerequisites, so this is notably more helpful in understanding content effectively while still remaining efficient.

Furthermore, the sidebar appears to be rather intuitive to use. Some UI designs require manually finding or looking up prerequisites, if they even tell you the prerequisites. With this approach, however, navigating to a prerequisite becomes as simple as clicking a single link. This can greatly reduce the barrier to finding prerequisite knowledge, helping prevent learners from getting confused by the things they don’t know. Furthermore, if the reader wanted to, they could scan through content anyways, and only reference the sidebar upon finding something they do not understand. This approach would be very convenient given this layout.

The prototype does not yet support any features related to enabling RQ2. As such, future iterations upon this concept should utilize prior research (as mentioned in the literature review section) to enhance the user’s ability to retain, understand, and engage with knowledge.

In order to enable better retention of knowledge, inspired by Hedberg et al.’s research discussed earlier, it should be attempted to incorporate the elements they discussed into the project. For example, simulation-based learning can be achieved by interactive simulations (such as allowing learners to toggle between logical operators to see their effects, or to run small snippets of code to see their effects). Learning by reflection can be achieved by giving learners an area where they can write down notes, and asking them to complete said notes after they have finished the topic.

Another feature that should be implemented to allow better retention of knowledge is a quizzing system. This system would track how many attempts the user takes in order to get a high accuracy rate on questions related to the content, and then would still ask another number of questions as determined by a percentage of the amount of attempts it originally took to reach the
high accuracy rate. This would incorporate elements of the research on overlearning that Driskell et al. performed. In order to also incorporate the research on metamemory that Schwartz and Efklides performed, the system should also give users a notification to take the quiz again after a set period of time.

VIII. Conclusion

In summary, this project outlined a potential solution to an issue in which existing learning resources either do not support Just-In-Time learning, or leave users confused because lesson content assumes that the user has prior knowledge which that user does not have. This project began with an investigation of existing research in the computer science and education fields, in order to improve upon the potential solution using already established and verified concepts. It also included an investigation of what topics would be useful to focus on teaching, and what order those topics are generally taught in. After this, this project involved creating a prototype of a potential solution, allowing for evaluating the solution in practice and gathering qualitative properties. Using these qualitative properties, both merits and demerits of the particular implementation were discussed, and potential future work towards improving the approach was suggested based on this. The overall findings were that using a graph-based learning structure can be effective, but that the effectiveness depends on how this information is presented to the user. Additionally, there is a large resource cost when attempting to build a graph-based learning structure.
IX. References


[7] “Computer programming - javascript and the web | computing,” Khan Academy,

[8] Z. Shaw, “Table of contents,” Learn Ruby the Hard Way,


