



WAYNE STATE UNIVERSITY

STEM Day Lesson Plan

Title: Got Code? An Introduction to Coding

Subject Area: Technology-Computer Coding

Learning Activity Description:

This hands-on activity will walk participants through an introduction to computational thinking and coding. Participants will also have an opportunity to explore coding applications and will leave with a list of free coding programs they can download on their own computer or mobile devices to continue their exploration of coding.

Lesson Activity Objective:

- Students will participate in a Computational Thinking exercise.
- A hands-on activity will walk participants through an introduction to coding using the Swift Playground app on an iPad.
- Participants will work independently and as teams practicing coding skills.
- Participants will leave with a list of free coding apps they can load on their own devices to continue their exploration of coding.

Lesson Activity Outcomes:

- Students will understand the basics of computational Thinking.
- Student will be able to use computational thinking to problem solve both in technology and everyday applications.
- Students will be excited to explore computational thinking and coding as beneficial tools in developing their careers.

Materials/Supplies Listed:

- Chalkboard or whiteboard & marking pens/chalk
- Computer or iPad with Swift Playgrounds
- List of coding recourses
- Activity 1 worksheet
- Pens or pencils

Teacher Procedures:

- Complete Activity 1
 - “Your students get to tell you what to do! As a class, they should come up with ideas for what you should do. Examples might be to draw a smiley face on the

board or do five jumping jacks. Let your students decide on the action without telling you what the action is. Once they've decided, they can shout out step-by-step directions to you. Were you successful? How could you improve the directions? Your students were just telling you commands within a sequence, which is what you need to do when you write code." Excerpt From: Apple Education. "Swift Playgrounds: Learn to Code 1 & 2." Apple Inc. - Education, 2016. iBooks. <https://itunes.apple.com/us/book/swift-playgrounds-learn-to-code-1-2/id1118578018?mt=11>

- Activity 2 (appended to this document)
 - Students work independently or with another classmate to work through the Playgrounds modules.

Preparation Time for Learning Activity:

- Set up technology

Room set-up:

- Set up to work in small groups (3-4) and independent work.

Group Strategies (example, group size, expected time for groups, etc.)

- Activity 1: Work in groups of 3-4 for about 5 minutes; whole group discussion about 10-15 minutes leading to discussion of computational thinking.
- Activity 2: Independent work or ask classmate for help working through coding puzzles.

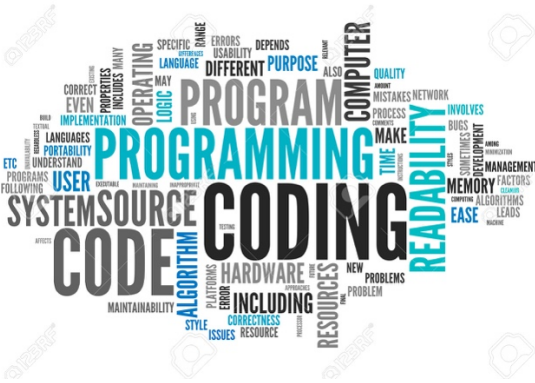
Student Products/Artifacts/work pages:

- Activity 1 worksheet indicating steps for directions for creating a smiley face on the whiteboard/chalkboard.
- Activity 2: Complete at least module 1 of Swift Playground.

Assessment Criteria/Rubric:

Closing/Transition to next activity:

- Recommendation to explore the coding and computational thinking resources provided.
- Think about how coding can help with problem-solving, creativity, communication, and real-life skills.



College of Education

"Got Code? An Introduction to Coding"

Dr. Mary Waker & Ms. Chelsea Smith

March 13, 2018

Activity 1 – Part 1

Computational thinking – What is it?

What is the purpose of computer coding?

How does computational thinking relate to coding?

Activity 1 – Part 2

In teams of 3-4 students: Create a list of “commands” for Dr. Waker to draw a smiley face on the whiteboard. Write your list of commands below. Number each command. Select a spokesperson from your table to share your list with the group.




College of Education

"The Computer in Your Pocket"

Dr. Mary Waker & Ms. Chelsea Smith

March 13, 2018

Activity 2 – You Code!

1. Open the Swift Playgrounds app on the iPad 
2. Select the “Learn to Code 1” module
3. Read through the introduction
4. Start Coding!
5. Move through as many exercises as you can without using the hints. Work with a classmate if you run into problems.

Question:

What did you learn going through the levels?



WAYNE STATE

College of Education

“Got Code? An Introduction to Coding” Teacher Coding Resources STEM Day – March 13, 2018

Objectives

- Students will participate in a Computational Thinking exercise.
- A hands-on activity will walk participants through an introduction to coding using the Swift Playground app on an iPad.
- Participants will work independently and as teams practicing coding skills.
- Participants will leave with a list of free coding apps they can load on their own devices to continue their exploration of coding.

Resources:

- 1) How to do the Hour of Code: <https://www.youtube.com/watch?v=SrnvvWDm73k&feature=youtu.be>
- 2) Star Wars Hour of Code: <https://www.youtube.com/watch?v=vNjiHkQQI6A>
- 3) Building a Galaxy with Code: <https://studio.code.org/s/starwars/>
- 4) Hour of Code Activities: <https://code.org/learn>
- 5) Getting the Most Out of Hour of Code: https://www.edutopia.org/blog/calling-learners-teachers-hour-of-code-ashley-cronin?utm_source=Edutopia%20News&utm_campaign=1d2be19689-EMAIL_CAMPAIGN_112217_enews_classroomexercises_mc&utm_medium=email&utm_term=0_29295b4c8b-1d2be19689-47971007
- 6) 15 Ways of teaching every student to code (even without a computer): <https://www.edutopia.org/blog/15-ways-teaching-students-coding-vicki-davis>
- 7) Teach Computer Science: <https://studio.code.org/courses?view=teacher>
- 8) Hour of Code by grade level: <https://askatechteacher.com/2015/11/09/hour-of-code-3/>
- 9) Hour of Code for all ages: <https://code.org/hourofcode/overview>
- 10) 24 Hour of Code Resources for Teachers and Schools: <http://classtechtips.com/2016/12/01/24-hour-code-resources-teachers-schools/>
- 11) Swift Playgrounds Curriculum Guide: https://images.apple.com/education/docs/Swift_Playgrounds_Curriculum_Guide.pdf
- 12) Swift Playgrounds: Learn to Code 1 & 2-Teacher Guide <https://itunes.apple.com/us/book/swift-playgrounds-learn-to-code-1-2/id1118578018?mt=11>

6 Reasons why your child (and mine!) should learn to code

By Simon Julian

<https://www.sitepoint.com/6-reasons-why-your-child-and-mine-should-learn-to-code/>

Second in a pretty loose series based around why I think that coding is a necessary skill for kids to learn as they grow (ideally starting in Primary school).

In an era of increasingly fast technological innovation and change, there is a growing necessity for people not only to know how to use the connected world (being anything from computers, programmable devices, drones, the IoT, etc.), but also to understand how to utilize these things in new, innovative, untested and unusual ways. With this new necessity in mind, coding should become as basic as reading and writing, and be treated as such in school curriculums the world over. As a father of a daughter and a son I want to see them given the right tools, thought processes and education to thrive in an entrepreneurial and innovation-led environment.

Here are some reasons why you should teach your child to code.

Code is one of the world's most widely used languages

Computer science is the new language of the world, and it's also one of the fastest growing occupations. Almost every field of human endeavor relies more and more on software and software development for success. Other than coding and logic thinking being necessary skills, they'll also give your child a chance to both be well paid in the future and to do some of the more interesting and challenging jobs likely to be around. It has been predicted that by 2020 there are likely to be over 1 million unfilled jobs in the US alone that will be dependent on coding.

Connected devices are already reshaping our world

Kids also need to learn how to code because computers and connected devices are what will shape their world as they grow up, just as physics, Math, Chemistry and Biology shape our world right now. In about twenty years, an inability to code or at least to understand logic-based thinking to some level will be just as crippling as illiteracy and innumeracy is today. Better to start equipping your child with these skills as early as possible.

Kids learn better and faster when they're young

Teaching your child how to code when they are still young is important, because kids have an easier time learning skills than adults do – their minds are flexible and open, and learning code is like learning a language, far easier when you are young.

Given the sheer pace of innovation and the growing connectedness of our devices, houses, workplaces and potentially ultimately bodies, it makes sense to take advantage of the natural inclination that children have to learn faster and better when they are younger. I'm not just talking about coding in and of itself, but also logic-based thinking – 'procedural literacy', which is an ability to think about and understand processes in the world.

Coding is creativity unleashed

Another benefit of teaching kids how to code is that it builds their confidence and creativity and provides the tools to create a world of limitless possibilities, where they can build their own paths and solutions in their own way. Overall, coding is a very empowering skill.

Just like art and craft is a way to express creativity, coding can be a highly engaging, fun and empowering skill for kids today. There are some platforms, like Scratch, that teach kids how to code in a

way that doesn't make them feel like they are stuck in a classroom learning something they don't want to. They start out by playing games, and if the child doesn't like a particular aspect of the game, they can modify it to their preference by rewriting the code. They are also able to find games or apps that other kids have modified, and they can add their own spice to them if they want to. Finding a toolset that allows kids to utilize what I know to be their most amazing skill – virtually limitless imagination, is one of the best aspects of helping them learn coding.

Problem solving and critical thought

As discussed briefly above, learning coding techniques also helps children to develop critical thinking skills and problem solving processes that are not only important in computer science, but also in life. It teaches them how to look at the bigger picture, and to break down big challenges into smaller and more manageable tasks. Other than just relying on environmental perception, they are able to make logical connections which will help them learn how to properly analyze different situations. This type of skill is necessary for everyone to have, not just those who want to pursue software engineering in the future. Other than it being professionally applicable, it also helps one to achieve big and overwhelming goals in their lives. You get to think of the smaller steps you need to take in order to get you closer to your goals, which gives you more drive and focus.

The wonders of storytelling through code

Learning how to code is also going to help your children develop fluidity in their thinking. Coding is sequential – it's telling a story, where you need to know what to write and why one thing follows the other in a particular order. Most programming languages designed for kids usually use games to teach them how to code, and this requires them to follow (or even better create) a story line or sequence as they play and code.

Scratch is very popular with kids, as it offers a fair amount of control and is also based around storytelling and creativity, with a rich visual language that kids love.

Beyond platforms like Scratch there are lots of programmable devices (drones, robots, Arduino platforms etc) that allow kids the ability to both control and influence the world around them in creative and fun ways – all pretty amazing for teaching how amazing coding can be as a skill in the internet of everything.

As a final word, as a parent I don't want to be satisfied with my kids just knowing how to use a smartphone or a tablet – it's not enough anymore and they'll need more than that to thrive, innovate and create as they grow. Coding is an essential skill even if it hasn't yet been introduced to your local schools. In Estonia, the country that brought Skype to the world, public school first graders are learning how to code. Don't allow your child to be left behind (and I won't let mine either!)



WAYNE STATE

College of Education

“Got Code? An Introduction to Coding” STEM Day – March 13, 2018

Objectives

- This hands-on activity will walk participants through an introduction to coding using the Swift Playground app on an iPad.
- Participants will work independently and as teams practicing coding skills.
- Participants will leave with a list of free coding apps they can load on their own devices to continue their exploration of coding.

Resources:

Articles:

Coding outside the lines: CoderDojos get kids psyched about programming by turning them loose <http://hechingerreport.org/coding-outside-lines-coderdojos-get-kids-psyched-programming-turning-loose/>

Lifewire Article: https://www.lifewire.com/kids-programming-languages-4125938?utm_campaign=computersl&utm_medium=email&utm_source=cn_nl&utm_content=8818415&utm_term=bouncex8

Websites & Apps:

Common Sense Media: Cool Coding Apps & Websites for kids:

<https://www.commonsensemedia.org/lists/coding-apps-and-websites#>

Best Apps and Websites for Learning Programming and Coding:

<https://www.commonsense.org/education/top-picks/best-apps-and-websites-for-learning-programming-and-coding>

Code School: <https://www.codeschool.com/courses/try-ios> & <https://www.codeschool.com>

9 Places you can learn how to code for free: <http://www.inc.com/larry-kim/9-places-you-can-learn-how-to-code-for-free.html>

Learn to code for free: <http://mashable.com/2013/03/13/learn-to-code-free/#mUwW82qm6agc>

Hour of Code: <https://code.org/learn>

Swift Playgrounds: <http://www.apple.com/swift/playgrounds/>

Scratch: <https://scratch.mit.edu>

Blockly: <https://developers.google.com/blockly/>

Alice: <http://www.alice.org/index.php>

Swift Playgrounds: <http://www.apple.com/swift/playgrounds/>

Twine: <https://twinery.org>

Lego Mindstorms: <https://www.lego.com/en-us/mindstorms/downloads/download-software>

Kodu: <https://www.kodugamelab.com/about>

Cargo-Bot (app): Cargo-Bot is a puzzle game where students teach a robot how to move crates. <https://itunes.apple.com/us/app/cargo-bot/id519690804?mt=8>

Hopscotch: <http://www.gethopscotch.com>

Program characters to move, draw, and collide with each other, and use shaking, tilting, or even shouting at the iPad to control them. Hopscotch was inspired by MIT's Scratch and gives kids a creative way to learn the fundamentals of coding and computer programming.

Codecademy: <https://www.codecademy.com>

Students can learn how to build things online by programming with Codecademy. The app introduces users to the basic concepts behind the apps on their phone and the websites they visit. They'll learn to understand the basic structure of code when they see it.

Alice: <http://www.alice.org>

Alice is an innovative 3D programming environment that makes it easy to create an animation for telling a story, playing an interactive game, or a video to share on the web. Alice is a freely available teaching tool designed to be a student's first exposure to object-oriented programming. It allows students to learn fundamental coding concepts in the context of creating animated movies and simple video games. In Alice, 3-D objects (e.g., people, animals, and vehicles) populate a virtual world and students create a program to animate the objects.

Mozilla Thimble: <https://thimble.webmaker.org>

Thimble makes it ridiculously simple to create and share your own web pages. Write and edit HTML and CSS right in your browser, then instantly preview your work. Host and share your finished projects with a single click. Perfect for beginners and experts alike. web-based code editor, part of the company's recently unveiled "Webmakers" project. Thimble is designed to give novice webmakers an easy-to-use online tool to quickly build and share webpages.

Tynker (app): <https://itunes.apple.com/us/app/tynker-learn-programming-visual/id805869467?ls=1&mt=8>

Tynker helps children develop computational thinking and coding skills in a fun, visual, intuitive, and imaginative way. Tynker is used in over 8,000 schools to teach computer programming. More than 6 million kids have started coding with Tynker. Students can solve fun puzzles and learn to code. Simply drag & drop visual code blocks and program characters to beat the level. Additional adventures and puzzle levels are available as in-app purchases.

Scratch: <https://scratch.mit.edu>

With Scratch, students can program your own interactive stories, games, and animations. They can share their creations with others in the online community. Scratch helps young people learn to think creatively, reason systematically, and work collaboratively.

Daisy the Dinosaur (app): <https://itunes.apple.com/us/app/daisy-the-dinosaur/id490514278>

This free coding app has an easy drag and drop interface that kids of all ages can use to animate Daisy to dance across the screen. Students will intuitively grasp the basics of objects, sequencing, loops, and events by solving the app's challenges. After playing Daisy, kids can choose to download a kit to program their own computer game.

Code Monster & Others: <http://www.crunchzilla.com>

Code Monster, from Crunchzilla, is an interactive tutorial for kids that focuses on action. Code changes immediately yield visible results. Projects start with simple boxes and colors, rapidly progressing into exciting experiments with simple animation and fractals. Important programming concepts like variables, loops, conditionals, expressions, and functions are introduced by example.

Kodu: <https://www.microsoft.com/en-us/download/details.aspx?id=10056>

Kodu is a new visual programming language made specifically for creating games. It is designed to be accessible for children and enjoyable for anyone. The visual nature of the language allows for rapid design iteration using only an Xbox game controller for input (mouse/keyboard input is also supported).

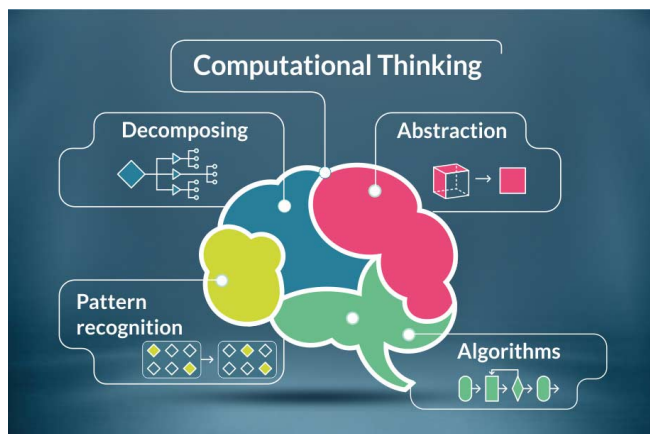


Computational Thinking Resources

“Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand.” (BBC: Introduction to computational thinking)

- Introduction to computational thinking (BBC)
 - <https://www.bbc.com/education/guides/zp92mp3/revision>
- Wikipedia: Computational thinking
 - https://en.wikipedia.org/wiki/Computational_thinking
- Google for Education-Exploring Computational Thinking:
 - <https://edu.google.com/resources/programs/exploring-computational-thinking/>
- What is Computational Thinking?
 - <https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1>
- Computational Thinking for Educators:
 - https://computationalthinkingcourse.withgoogle.com/course?use_last_location=true
- Developing Computational Thinking with Scratch Coding (video)
 - <http://www.speedofcreativity.org/2017/12/01/developing-computational-thinking-with-scratch-coding-webinar-video/>

Cornerstones to computational thinking (BBC):



- decomposition:** breaking down a complex problem or system into smaller, more manageable parts
- pattern recognition:** looking for similarities among and within problems
- abstraction:** focusing on the important information only, ignoring irrelevant detail
- algorithms:** developing a step-by-step solution to the problem, or the rules to follow to solve the problem

It is a way of thinking
It is making the impossible possible
It is creating solutions to problems in everyday life

Making mistakes

I can enjoy things that go wrong and learn from them.
I see mistakes as a normal part of solving problems.

Perseverance

I don't give up. I'm prepared to keep having a go to see what happens.
I keep going, even when things seem confusing. I'm determined to find solutions

Imagination

I can look at things in unusual ways.
I'm ready to consider the impossible. Sometimes I leave a problem for a while. A solution might come to me when I'm thinking about something else.

Collaboration

I can use other people's ideas.
I can share my ideas.
We can talk together to solve a problem.
I can teach my peers and they can teach me.

It is not thinking like a computer
It is not always using a computer as the solution
It is not limiting creativity

Pattern recognition

Is this similar to a problem I've already solved? How is it different?
How are the parts of the problem connected?

Decomposition

Can I explain the different parts of this problem and solution?

Algorithm design

What do I need to think about to make this happen?
What are the steps I will need to do to solve this problem?

Abstraction and generalisation

Which is the information I actually need?
What don't I need to know?
Have I made this more complicated than I need to?
Will this work for other things?

The Computational Thinker:

Attitudes and Skills



Problem solving
Designing solutions
Understanding behaviour

Attitudes

Skills



Briggs, J. Benefits of Programming (2013) https://doi.org/10.1007/978-1-4939-9100-0_10
Google. Exploring Computational Thinking (<https://www.google.com/edu/computational-thinking/index.html>)
Wing, J. Computation Thinking (2008) <http://www.cs.cmu.edu/~wing/notes/lectures/computational-thinking.pdf>