App Authors Curriculum
(Last Revised: Spring 2017)

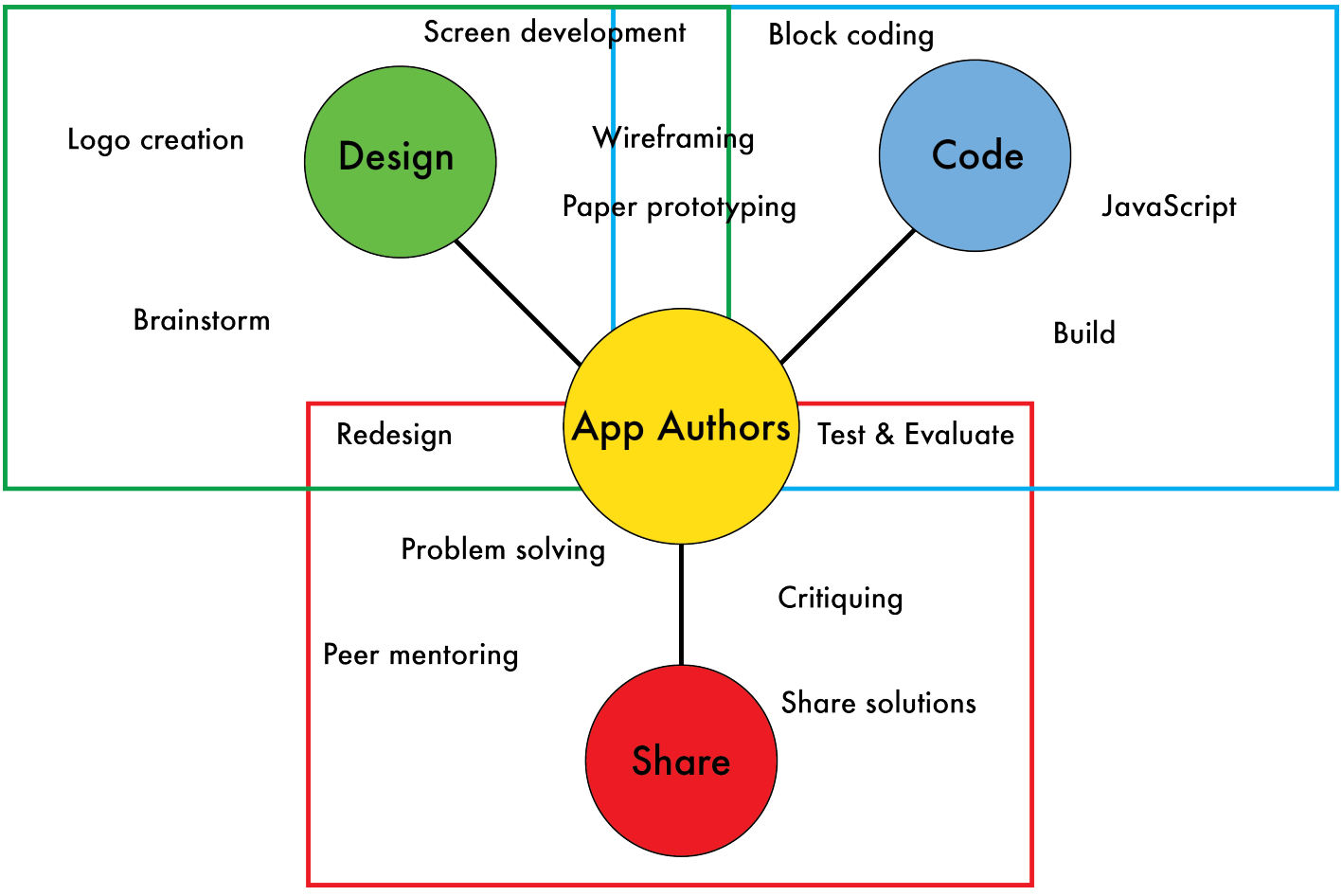
Introduction and Overview:

With STEM education a current national priority, public and school libraries can play an important role in creating STEM learning spaces for young people, especially those young people who have limited access to technology at home.

Apps provide an excellent entrance point for coding and STEM explorations, since even young people without their own tablets or smartphones are likely to possess basic competency with their interfaces.

This curriculum is geared toward school and public libraries and intends to teach young people to create apps and share their achievements with their peers and communities. Students will develop fundamental and introductory computational thinking skills through unplugged activities and coding in Code.org's App Lab (and/or Play Lab).

App Authors programming is structured in three basic phases: designing, coding, and sharing. Collaboration, brainstorming, critiquing, and re-designing are incorporated throughout the curriculum. The diagram below illustrates the three major phases of this curriculum, along with the fundamental topics to be covered.



Essential Questions:

- What's an app?
- Who makes apps? Why do they make them?
- Why do people use apps? Why are apps important?
- How are apps made?
- What is coding?
- How do coders think of ideas for apps? How can I think of those ideas?
- How can I use coding to express my ideas or help my community?
- What goes into designing an app?
- What tools are available for making apps and coding?
- What is the design process? How can it be followed to create apps?
- How do I know what changes to make or when to make them?
- Who are users? Why do users matter?
- Why do I have to use paper and pencil to work on app design?
- What is computational thinking?
- Why do people share apps?
- How can I share my own app?

Standards:

International Society for Technology in Education (ISTE) Standards

1. Creativity and innovation: Students demonstrate creative thinking, construct knowledge, and develop innovative products and processes using technology.

    b. Create original works as a means of personal or group expression.

2. Communication and collaboration: Students use digital media and environments to communicate and work collaboratively, including at a distance, to support individual learning and contribute to the learning of others.

    c. Develop cultural understanding and global awareness by engaging with learners of other cultures.
    d. Contribute to project teams to produce original works or solve problems.

3. Research and information fluency: Students apply digital tools to gather, evaluate, and use information.

    b. Locate, organize, analyze, evaluate, synthesize, and ethically use information from a variety of sources and media.

4. Critical thinking, problem solving, and decision making: Students use critical thinking skills to plan and conduct research, manage projects, solve problems, and make informed decisions using appropriate digital tools and resources.

a. Identify and define authentic problems and significant questions for investigation
d. Use multiple processes and diverse perspectives to explore alternative solutions.

5. Digital citizenship: Students understand human, cultural, and societal issues related to technology and practice legal and ethical behavior.

b. Exhibit a positive attitude toward using technology that supports collaboration, learning, and productivity.
d. Exhibit leadership for digital citizenship.

6. Technology operations and concepts: Students demonstrate a sound understanding of technology concepts, systems, and operations.

a. Understand and use technology systems.
c. Troubleshoot systems and applications.
d. Transfer current knowledge to learning of new technologies.

American Association of School Librarians (AASL) Standards for the 21st-Century Learner

1. Inquire, think critically, and gain knowledge

1.1.9 Collaborate with others to broaden and deepen understanding.
1.3.4 Contribute to the exchange of ideas within the learning community.
1.4.2 Use interaction with and feedback from teachers and peers to guide own inquiry process.

2. Draw conclusions, make informed decisions, apply knowledge to new situations, and create new knowledge.

2.1.5 Collaborate with others to exchange ideas, develop new understandings, make decisions, and solve problems.
2.1.6 Use the writing process, media and visual literacy, and technology skills to create products that express new understandings.
2.2.4 Demonstrate personal productivity by completing products to express learning.
2.3.2 Consider diverse and global perspectives in drawing conclusions.
2.4.3 Recognize new knowledge and understanding.
2.4.4 Develop directions for future investigations.

3. Share knowledge and participate ethically and productively as members of our democratic society.

3.1.4 Use technology and other information tools to organize and display knowledge and understanding in ways that others can view, use, and assess.

3.1.5 Connect learning to community issues.
3.1.6 Use information and technology ethically and responsibly.
3.2.1 Demonstrate leadership and confidence by presenting ideas to others in both formal and informal situations.
3.2.2 Show social responsibility by participating actively with others in learning situations and by contributing questions and ideas during group discussions.
3.2.3 Demonstrate teamwork by working productively with others.
3.3.2 Respect the differing interests and experiences of others, and seek a variety of viewpoints.
3.3.4 Create products that apply to authentic, real-world contexts.
3.4.2 Assess the quality and effectiveness of the learning product.

4. Pursue personal and aesthetic growth.

4.1.5 Connect ideas to own interests and previous knowledge and experience.
4.1.7 Use social networks and information tools to gather and share information.
4.1.8 Use creative and artistic formats to express personal learning.
4.2.3 Maintain openness to new ideas by considering divergent opinions, changing opinions or conclusions when evidence supports the change, and seeking information about new ideas encountered through academic or personal experiences.
4.3.1 Participate in the social exchange of ideas, both electronically and in person.
4.4.2 Recognize the limits of own personal knowledge
4.4.3 Recognize how to focus efforts in personal learning.

Next Generation Science Standards

3-5-ETS1-1. Define a simple design problem reflecting a need or a want that includes specified criteria for success and constraints on materials, time, or cost.

3-5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and restraints of the problem.

MS-ETS1-4. Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.

HS-ETS1-2. Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.

HS-ETS1-3. Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.

Common Core

ELA/Literacy

CCSS.ELA-LITERACY.RL.5.6
Describe how a narrator's or speaker's point of view influences how events are described.

CCSS.ELA-LITERACY.RI.5.7
Draw on information from multiple print or digital sources, demonstrating the ability to locate an answer to a question quickly or to solve a problem efficiently.

CCSS.ELA-LITERACY.RI.6.7
Integrate information presented in different media or formats (e.g., visually, quantitatively) as well as in words to develop a coherent understanding of a topic or issue.

CCSS.ELA-LITERACY.SL.5.1.A
Come to discussions prepared, having read or studied required material; explicitly draw on that preparation and other information known about the topic to explore ideas under discussion.

CCSS.ELA-LITERACY.SL.5.1.B

Follow agreed-upon rules for discussions and carry out assigned roles.

CCSS.ELA-LITERACY.SL.5.1.C

Pose and respond to specific questions by making comments that contribute to the discussion and elaborate on the remarks of others.

CCSS.ELA-LITERACY.SL.6.1
Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly.

CCSS.ELA-LITERACY.RST.6-8.4
Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to *grades 6-8 texts and topics*.

CCSS.ELA-LITERACY.WHST.6-8.2.D
Use precise language and domain-specific vocabulary to inform about or explain the topic.

Mathematics

CCSS.MATH.CONTENT.7.SP.A.1
Understand that statistics can be used to gain information about a population by examining a sample of the population; generalizations about a population from a sample are valid only if the sample is representative of that population. Understand that random sampling tends to produce representative samples and support valid inferences.

CCSS.MATH.CONTENT.7.SP.A.2
Use data from a random sample to draw inferences about a population with an unknown

characteristic of interest. Generate multiple samples (or simulated samples) of the same size to gauge the variation in estimates or predictions.

National Art Education Association (NAEA) Art Standards

Creating
1. Generate and conceptualize artistic ideas and work.
2. Organize and develop artistic ideas and work.

Presenting
5. Develop and refine artistic work for presentation.

Responding
7. Perceive and analyze artistic work.

Connecting
11. Relate artistic ideas and works with societal, cultural and historical context to deepen understanding.

Computer Science Teachers Association (CSTA) K-12 Computer Science Standards

3-6 (L1:6:CT)
- 3. Demonstrate how a string of bits can be used to represent alphanumeric information.
- 6. Understand the connections between computer science and other fields.

3-6 (L1:6:CL)
- 2. Use online resources (e.g., email, online discussions, collaborative web environments) to participate in collaborative problem-solving activities for the purpose of developing solutions or products.
- 3. Identify ways that teamwork and collaboration can support problem solving and innovation.

3-6 (L1:6:CPP)
- 5. Construct a program as a set of step-by-step instructions to be acted out (e.g., make a peanut butter and jelly sandwich activity).
- 6. Implement problem solutions using a block-based visual programming language.

3-6 (L1:6:CI)
- 1. Discuss basic issues related to responsible use of technology and information, and the consequences of inappropriate use.
- 4. Understand ethical issues that relate to computers and networks (e.g., equity of access, security, privacy, copyright, and intellectual property).

Objectives:

Students will be able to...

- **Ask** questions about apps and how they are used in their personal, school, and social lives.
- **Explain** what apps are, why they are important, and how they can be used to solve problems.
- **Identify** the kinds of apps they use and/or like and justify why these apps are "good."
- **Brainstorm** at least three fully developed ideas for apps, including an app title, a proposed solvable problem, and a target audience.
- **Select** one app idea and justify why this app is important to them before moving on to planning and constructing the app.
- **Participate** in unplugged activities such as paper prototyping, wire framing, and mind mapping in order to prepare for their app design and coding.
- **Practice** coding with a visual, block-based programming environment (such as App Lab) and will gain a basic understanding of the block capabilities and functions.
- **Design** a (beginning) maximum of 4 screens in App Lab that utilize buttons, images, and text in order to illustrate their app ideas.
    - **Summarize** what design features App Lab provides and hypothesize about how they might use them in their apps.
    - **Drag and drop** buttons, labels, text, and screens into their apps in order to enhance their designs.
    - **Add** text, use color, and upload images into their apps in order to personalize their designs.
    - **Label** design elements with an element ID in order to practice organizational skills during the app building process.
    - **Utilize** design features in App Lab that reflect their brainstorms and preliminary sketches.
- **Code** a basic mobile app that employs user interface blocks, control blocks, and function blocks in App Lab.
    - **Use** UI control blocks to show text, images, buttons, and other app interface features on the screen.
    - **Collect and save** data from App Lab apps.
    - **Control** the flow of their programming using control blocks, which can create loops.
    - **Draw** in an App Lab Canvas and experiment with App Lab Canvas functions.
- **Create** at least one functioning app that utilizes 4 screens. All apps should be functional, user-friendly, and have the capability to go back and return to a home screen.
- **Reflect** about their app development processes through individual writing, and small and large-group discussions.
- **Edit and refine** their apps after receiving verbal and/or written peer feedback in order to fully implement the design process.
- **Question** whether or not their apps are "done," and consider what edits could be made.

- **Present** their "beta" app with at least one peer and listen to their constructive feedback.
- **Evaluate** one peer's app and offer at least one compliment and one constructive suggestion.
- **Share** their apps with peers, friends, and/or family.

Supplies:

- Computers with Internet access—App Lab works best on computer interfaces. While it is available on tablets and iPads, it must be accessed in a web browser and touch functionality is not ideal for the program interface.
- Projector or SmartBoard—The teacher should have the technology available to work through tutorials, show videos, and present instructional slides.
- App Lab logins for each participant. App Lab gives teachers the ability to set up a class. Each student should have a login card with a passcode or secret picture.
- Headphones (optional)—In order to individually visit App Lab tutorials, or listen to music or white noise, students should either bring headphones from home or be able to access them in the library.
- Post-it notes
- Notecards
- Writing utensils
- Scissors
- Glue/Glue sticks
- Paper blocks for unplugged coding exercises
- One posterboard or large-scale paper per participant
- Student portfolio folders
- Assorted drawing/art materials (markers, crayons, colored pencils, etc.)

Unit Activities:

The unit activities during App Lab programming follow the steps in the design process. To begin, each student can have a posterboard or large paper to keep track of all of his or her brainstorming, design work, and notes. This is an optional addition to the curriculum, but serves as a nice visual framework for students to gauge their progress and track their ideas. As students fill up each bar of the poster template, they "level up" to the next step in the design process.

Students should also have portfolios to store their work throughout the program.

Pre-Design Investigation

1. Ask questions about apps and how they are used in their personal, school, and social lives.
2. Explain what apps are, why they are important, and how they can be used to solve problems.
3. Identify the kinds of apps they use and/or like and justify why these apps are "good."

Process:

- In a large group, the instructor should begin by introducing the App Authors program, any teachers/volunteers/librarians in the room, and the purpose of the research.
    - "We want you to have the chance to bring your ideas for apps to life, and to share those apps with your communities."
- Students should then work together to create a list of questions that will frame the program.
    - "What questions do you have about apps?" The list should be saved so that it can be displayed and used throughout the program to reinforce learning goals. Large paper, word documents, or a picture of written questions on a white board are all options. If students need prompting, consider the following follow-up questions:
        - What apps do you use? What would you like to know about how apps are made? Or why they exist? What do you know about coding? What would you like to learn about coding?
    - Instructors should use these questions as a basis for further instruction. Try to answer each of these questions throughout the program.
- Next, the students will work together to create a list of the class's favorite apps. This can be done using anonymous voting (on slips of paper), or through an online voting resource, such as **Socrative.com**. Together, the students should vote for their top 3 favorite apps, which the instructor will compile into a list. Once the list of all favorite apps is on the board, the class should discuss the apps with the most votes. This can be done in a large group, via activities like think-pair-share, or in small groups.
    - What do these apps do? Do they solve any problems? What problems?
    - Why are these apps important?
    - What do you think went into making these apps? Who do you think made these apps and why?
    - What makes some apps popular and others not so popular? What makes an app "good"?
    - Who are these apps meant for? Who is the audience the app creator had in mind? Who is the *actual* audience?
    - Who do these apps represent? Who do these apps leave out? Are there any problems with these apps?
- Finally, students should have at least 20-30 minutes dedicated to playing with their favorite apps. This is a time for exploration.
    - Remind students to pay attention to the details of the apps they use.
        - How many different screens do you use? Count them as you move through the app.
        - If you had to make a map about how to use the app, what would that look like? Try to draw one.
        - Is there a home menu? How is it organized?
        - How is the app designed? What colors did the app creator use? What about icons, pictures, and sounds?

Identify a Problem/Brainstorm

1. Brainstorm at least three fully developed ideas for apps, including an app title, a proposed solvable problem, and a target audience.
2. Select one app idea and justify why this app is important to them before moving on to planning and constructing the app.
3. Participate in unplugged activities such as paper prototyping, wire framing, and mind mapping in order to prepare for their app design and coding.

Process:

- After the students have done some thinking about the apps they use and have critically analyzed their favorite features of these apps, it is time for them to start thinking about their own creations.
- To begin, students should spend an entire program session/class brainstorming three app ideas. They can do this on paper and then tape/glue their ideas to their app development posters.
    - Students can use the Brainstorming video on the **App Authors Vimeo Channel** to get inspiration. Some suggested brainstorming tactics include:
        - Make a mind map
        - Use an online idea generator
        - Create a list of "problems" and possible app solutions that could solve those problems
        - Generate a list of apps that you *wish* existed. How might you make these come to life?
        - Free draw or write
- Students should develop a wealth of ideas, and should save everything that they brainstorm and glue or tape these brainstorming records to their posters.
- Students will then choose which app idea they would like to develop. Rather than just picking a favorite, encourage students to think about which app is the most important to them or to their communities.
- Once final ideas are selected, students will begin the process of planning their apps for design and coding.
    - Students should give their apps a name, identify the audience of the app, determine if the app solves a problem, and outline a specific purpose of their app.
        - The "Planning My App" handout can be used during this step in the planning.
    - Students should also learn to explain their app in only one sentence. Consider challenging students to "elevator talk" their app to each other or to the class. Can they explain their app in a concise sentence or a time limit of 30 seconds?
- Finally, students should engage in some unplugged activities to get them thinking about the coding, or the backbones of their app.
    - Questions to consider:
        - What is coding?

        - Have you coded before? What did you do? What was it like? Why did you do it?
        - How many computers did you use today? What is a computer? How did you use that computer?
        - Do we see the coding in computers that we use? Why or why not? What parts do we see?
    - It is crucial at this point to emphasize the difference between coding and user interface.
        - The code *powers* the app. "Computer programming, or coding, is how we tell a computer to perform a task, and understanding how to code puts the power of computers at your fingertips." (Payne, 2015, p. 1)
        - The design is just what users can *see*. "The app's visible components are the ones you can see when the app is launched—things like buttons, text boxes, and labels. These are often referred to as the app's user interface." (**App Inventor Book**, Chapter 14, p. 219)
    - To gain basic understanding of computational thinking and the specificity of computer coding, students should participate in at least one unplugged coding exercise. For a full list of ideas with specific instructions, utilize Code.org's unplugged fundamentals: **https://code.org/curriculum/unplugged**. Even something as simple as having students give step-by-step instructions for making a sandwich can illustrate the importance of sequenced and specific algorithms. (See an example of this here: **https://www.youtube.com/watch?v=B1H8XsOxw28**)
    - Depending on the group, activities can be done individually, in small groups, or as a large group.
- Students should engage in *at least one* paper prototyping exercise. Using prepared blank phone templates, buttons, and printed out elements from App Lab, students should prepare what their apps will look like.
    - Remember: users should be able to navigate to the home screen once they are in the "meat" of the app.
    - Students can partner or group together and use each other's paper prototypes. This is an excellent time to do a small or large-scale critique and give students feedback about how to make their designs or app ideas more user-friendly.
    - For an example of how to paper prototype, check out the following videos:
        - Google has a great series of videos about prototyping. Try the first one here: **https://www.youtube.com/watch?v=JMjozqJS44M**
- If time allows, students should also create a wireframe for their apps. Once they have narrowed down what there app will be called, who it is for, and what problem it solves, they will need to focus on how users will interact with the app.
    - Paper prototypes show what the app will *look* like. Wireframes show *how* the app will work.
    - For more information about what wireframes are and why they are used, check out this video: **https://www.youtube.com/watch?v=8-vTd7GRk-w**
- Remember: all paper prototypes, wireframes, or any paper planning that students do can go onto their posterboard under the Brainstorm and Design levels.

Design

1. Practice coding with a visual, block-based programming environment (such as App Lab) in order to gain a basic understanding of the block capabilities and functions.
2. Design a beginning maximum of *4 screens* in App Lab that utilizes buttons, images, and text in order to illustrate their app idea.

- **Summarize** what design features App Lab provides and hypothesize about how they might use them in their apps.
- **Drag and drop** buttons, labels, text, and screens into their apps in order to enhance their designs.
- **Add** text, use color, and upload images into their apps in order to personalize their designs.
- **Label** design elements with an element ID in order to practice organizational skills during the app building process.
- **Utilize** design features in App Lab that reflect their brainstorms and preliminary sketches.

3. Code a basic mobile app that employs user interface blocks, control blocks, and function blocks in App Lab.

Apps may include the following features:

- **Use** UI control blocks to show text, images, buttons, and other app interface features on the screen.
- **Collect and save** data from App Lab apps.
- **Control** the flow of their programming using control blocks, which can create loops.
- **Draw** in an App Lab Canvas, and experiment with App Lab Canvas functions.

4. Create at least one functioning app that utilizes 4 screens. All apps should be functional, user-friendly, and have the capability to go back and return to a home screen.

Process:
Link to App Lab: **https://code.org/educate/applab**

- First, students should spend at least an hour experimenting with App Lab in order to gain familiarity with the tool. App Lab is similar to Scratch, but the functionalities are better suited for app making and sharing.
  - Note to educators: Be sure to make an account in Code.org/App Authors in advance. You will need to set up your class and prepare secret codes or words for your students to login.
- App Lab is created with older middle school and high school users in mind. For students who have a harder time typing or following along with video pacing, it can be useful to provide links to tutorials on notecards. Students can view App Lab How-to videos through **Code.org** or our **App Authors YouTube playlist** with headphones or ear buds. Giving students the chance to work through tutorials at their own pace will differentiate for many ability levels.

- The best experiences when learning to use a new tool are spent exploring. Teachers can play with App Lab on a projected screen as well to show students how the tool works. App Lab is a tool that can work for absolute beginners and experienced coders because it allows users to switch between block and text coding. All instructors should spend time familiarizing themselves with the tool before programming begins. For specific tutorials, here are suggested links to a few:
    - App Lab Demo (general introduction): **https://www.youtube.com/watch?v=4DntNOOYmpo**
    - App Lab How-To Make a Simple App (general introduction): **https://www.youtube.com/watch?v=tDnoxkOSfQw**
    - App Lab How-To Introduction (general introduction): **https://www.youtube.com/watch?v=xIbRL5eOgkI**
    - App Lab How-To: Make a Whack-E-Moji Game (uses text coding for more advanced coders): **https://www.youtube.com/watch?v=WZVAA3ajZxc**
    - CS Principles: Intro to Design Mode in App Lab (introduction to how Design mode works, HTML and CSS): **https://www.youtube.com/watch?v=-EoTeD4mSNU**
    - App Lab How-To: Make a Choose Your Own Adventure App (Great for showing how to create design and code elements simultaneously): **https://www.youtube.com/watch?v=ZhgBSYKWL2A**
    - App Lab Tutorial: Build a Drum Machine in 45 Minutes (walks users through a complete app building process): **https://www.youtube.com/watch?v=jh5O85L5_0c**
- Students will work on creating an app with **4 total screens**. Many students with limited coding experience may want to devote most of their time to designing complex Screens in App Lab. While this can be useful (and a lot of fun), emphasize the importance of designing a limited amount of screens and focusing on the *code* first.
    - Present this as a challenge: "Focus your ideas and show us how you can make this work in only 4 screens."
    - If some students make it through 4 screens that are fully coded, functional, and well designed, they can move on to including more screens.
- App Lab works especially well for survey apps, quizzes, and question/answer structures. In fact, if students are inclined to collect data from app users, App Lab has the data functionality that allows them to do so.
- App Lab also gives coders the option to "remix" projects. On the App Lab homepage, scroll to "Explore these sample apps and take the challenge to make them even better" to see sample projects:
    - **https://code.org/educate/applab**
- Many students will want to create games. App Lab has limited capabilities to create characters or objects that move around the screen. For students who are more inclined to creating games, they might consider trying Code.org's Play Lab.
    - Play Lab: **https://code.org/playlab**
    - *Note: Play Lab does not give users the ability to share finished apps, like App Lab does. If coders would like to show off their Play Lab apps, they should take screenshots or pictures of their work as they go.*

- As students code and design their apps in App Lab, remind them to revisit their paper prototype plans and wireframe plans. Screens should have navigation options and users should be able to access the home screen.
- Because the work that students complete in App Lab/Play Lab will have little paper evidence to fill up their "Design" level of their posters, they should include wireframes, paper prototypes, any written peer feedback, or any notes that they take. If technology allows, students can also take screenshots of their in-progress App Lab designs and code and print them out for their posters.

Redesign

1. Reflect about their app development processes through individual writing, and small and large-group discussions.
2. Edit and refine their apps after receiving verbal and/or written peer feedback in order to fully implement the design process.

Process:

- Once students have had adequate time to create first "drafts" or beta versions of their apps, it is time to move on to critiquing and redesigning their work. At this point, some students may be further ahead than others. It is okay for students to move to the redesign phase without a complete beta version.
- There are many redesign/critique activities that can be done during this design step. The following activities will give students a chance to collaborate, provide and receive peer feedback, and reflect on their strengths and struggles:
    - First, try one session of **pair programming.**
        - Code.org's video about pair programming provides an outline of how this can be done as well as suggested guidelines for students: **https://www.youtube.com/watch?v=vgkahOzFH2Q**
        - Students can pair up for one program session, with one person designated as "driver" and the other as "navigator." Students should split the class time in two, and they will work through each partner's in-progress app. This way, students with more experience can provide expertise or assistance, while all students get the chance to work in a realistic programming style. This is a nice activity to give students informal, verbal feedback about the functionality and design of their apps.
    - Students can also participate in a **blind critique session**.
        - Have students set up their beta apps on their computers in an outward facing circle. Leave paper, post-its, or some type of note-taking device by each screen.
        - Each student should get 2-3 minutes at each app and leave at least one piece of constructive feedback and one compliment.
            - It might be a good idea to devote some time to talking about constructive criticism and the importance of taking it seriously.

Encourage students to frame all feedback with a compliment or a *reason why* they are providing the criticism.

- Students might also show their beta apps to small panels or the group as a whole.
  - Students can refer to their "elevator talk" or one-sentence description from the brainstorming phase and refine it for their beta app. They can then present their app by showing how it works and walking through the app name, intended audience, and problem that it solves.
  - Panel members or classmates can provide feedback verbally. Or, they might take notes and provide anonymous feedback to classmates at the end of the session.

- Once students have had the opportunity to give and receive feedback from peers and instructors, they should be given some program time to rework their apps. One program session for redesigning the app would be ideal.

Test and Evaluate

1. Question whether or not their apps are "done," and consider what edits could be made.
2. Present their "beta" app with at least one peer and listen to their constructive feedback.
3. Evaluate one peer's app and offer at least one compliment and one constructive suggestion.

Process:

- The Test and Evaluate step in the design process is for perfecting apps and adding final details and edits. Now that students have redesigned their apps and have moved to a second beta version, they should push themselves to question if their apps are done.
  - Students should ask themselves the following questions:
    - What's missing? If I could add anything to my app, what would I add?
    - If I could take anything away from my app, what would I remove? Why would I want to do that?
    - If I gave this app to a stranger, would it make sense to them?
    - Can I navigate in this app? Can I get back to home? Can I quit?
  - These questions can be answered in the format of a reflection worksheet, free writing, or verbally. Any written reflections should be added to the Test and Evaluate level of their posters, along with any available screenshots or additional pictures.
  - Then, students should participate in one final blind test activity.
    - In groups of 2-3, students will share their app (via the shareable link or on their computers). Students should not provide any explanation about the app.
    - Students will use their partner's app for 2-3 minutes. Then, they will conference with their partner about what they thought and if there was any confusion in this process. The critiquing skills that students built

during the Redesign phase should be utilized. (At least one compliment, at least one constructive criticism.)

- By the end of the Test & Evaluate phase, apps should be complete and ready to share.

Share

1. Share their apps with peers, friends, and/or family.

Process:

- During the Share phase, students will retrieve the sharable link for their app in App Lab. This can be copied onto their posters, as well as a screenshot or picture of their final app's screens or homepage.
    - If technology allows, students or teachers can also make QR codes linking users to the finished apps. Printed QR codes on the posters allow anyone with a QR reader to easily access the coder's app.
- Students should be encouraged to share this link with their communities, friends, and/or families.
- And remember: the design process is never truly over! There will always be changes, updates, and edits to be made. This is still version one, and there is plenty of room for improvement and further work.

Unit Conclusion:

Upon the conclusion of App Authors, educators may consider hosting a family session or fair where participants can formally present their apps. Consider creating QR codes of each app so that attendees can easily access the apps. The posters created through this process also serve as nice visuals for the final App Authors presentation. Families can also be asked to fill out short surveys at the final app sharing event so that librarians and teachers can get a better idea about student experiences, what can be improved in the future, and what worked well.

Students will receive a certificate showing their App Builder/Coder status, as well as a portfolio of information for further coding and app building.

Resources for further exploration:

App Authors YouTube Playlist:
https://www.youtube.com/playlist?list=PL6tcFBCFPA7IvAYt9ZuYMc8eiPmTtTvnG

Google Drive with spring 2017 curriculum materials:
https://drive.google.com/open?id=0B6zJ9bSuahYTUU9ET0RBYWFET2s

Code.org's CS Fundamentals Unplugged:
https://code.org/curriculum/unplugged

References:

AASL. 2007. *American association of school librarians: Standards for the 21st-century learner*. Retrieved from http://www.ala.org/aasl/sites/ala.org.aasl/files/content/guidelinesandstandards/learningstandards/AASL_Learning_Standards_2007.pdf.

Code.org. (2017). *App lab*. Retrieved from: https://code.org/educate/applab.

Code.org. (2015). *CS fundamentals unplugged*. Retrieved from https://code.org/curriculum/unplugged.

Code.org (2017). Video library. Retrieved from https://code.org/educate/resources/videos.

Common core state standards initiative. 2017. *English language arts standards*. Retrieved f rom http://www.corestandards.org/ELA-Literacy/.

ISTE. (2017). *ISTE standards for students*. Retrieved from: http://www.iste.org/standards/standards/standards-for-students.

National Core Arts Standards. 2014. *Dance, media arts, music, theatre and visual arts*. Retrieved from: http://nationalartsstandards.org/.

Next Generation Science Standards. 2017. *Standards by topic*. Retrieved from: http://www.nextgenscience.org/overview-topics.

PBS Kids. (2017). *Design squad global*. Retrieved from: http://pbskids.org/designsquad/.

Wolber, David. 2014. *App inventor 2: Create your own android apps*. O'Reilly Media. Retrieved from http://www.appinventor.org/bookChapters/chapter14.pdf .