# 01 Getting Started With App Development

Content Area: **Math**
Course(s):
Time Period: **Semester**
Length: **4 - 5 Weeks**
Status: **Published**

## General Overview, Course Description or Course Philosophy

Introduction to App Development is a semester course that introduces students to fundamental concepts of programming and application design through exercises, labs, and app projects. Students will become familiar with the process of debugging and troubleshooting their code; develop new ways of thinking that can be applied outside the classroom. They will learn how to articulate their ideas, collaborate on large complicated projects and how to design with a purpose. Students will have the opportunity to foster their ability to develop and apply computational and design thinking to address real-world problems while design creative solutions. Students will engage not only independently but will work collaboratively to navigate the dynamic digital landscape.

In this unit students will learn that programming is about storing, manipulating and controlling the flow of data. While exploring Xcode and playground, students will learn how to name and describe data through types, and how to use these values to simulate the real world. Students will also start to develop debugging strategies and be introduced to how documentation can be utilize. The unit culminates with an application project that introduce them to Interface Builder.

## OBJECTIVES, ESSENTIAL QUESTIONS, ENDURING UNDERSTANDINGS

Objectives:

- Explain the importance of data in programming
- Understand how to apply simple mathematical equations in coding
- Explain how swift helps you write safe code
- Determine the outcome of algorithms
- Identify the most appropriate data type for different situation

Essential Questions

- Why is there a need to debug?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- Why is data type safely important?
- Why is a device necessary for real-world testing?

- What relationship does data and programming have?
- How can we store data to solve problems?

Enduring Understanding

- Programming uses mathematical and logical concepts.
- People write programs to execute algorithms.
- Critical thinking skills are used to collect and analyze data while debugging.
- Data can not only be stored but manipulated and compared
- Top down programming can be controlled with conditionals
- Programs cannot be written without the use of data
- A variable can only store one piece of data at a time
- An algorithm accomplished a specific task that is made up of a finite set of instructions
- Conditionals determine which part of the code executes based on whether the condition is true or false
- The way statements are sequenced and combined determine the outcome of an algorithm

## CONTENT AREA STANDARDS

| | |
|---|---|
| CS.9-12.8.1.12.AP.1 | Design algorithms to solve computational problems using a combination of original and existing algorithms. |
| CS.9-12.8.1.12.AP.2 | Create generalized computational solutions using collections instead of repeatedly using simple variables. |
| CS.9-12.8.1.12.AP.5 | Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. |
| CS.9-12.8.1.12.CS.2 | Model interactions between application software, system software, and hardware. |
| CS.9-12.8.1.12.CS.4 | Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. |

## RELATED STANDARDS (Technology, 21st Century Life & Careers, ELA Companion Standards are Required)

| | |
|---|---|
| LA.K-12.NJSLSA.L1 | Demonstrate command of the conventions of standard English grammar and usage when writing or speaking. |
| LA.K-12.NJSLSA.L6 | Acquire and use accurately a range of general academic and domain-specific words and phrases sufficient for reading, writing, speaking, and listening at the college and career readiness level; demonstrate independence in gathering vocabulary knowledge when encountering an unknown term important to comprehension or expression. |
| LA.K-12.NJSLSA.SL1 | Prepare for and participate effectively in a range of conversations and collaborations with diverse partners, building on others' ideas and expressing their own clearly and persuasively. |
| LA.K-12.NJSLSA.SL2 | Integrate and evaluate information presented in diverse media and formats, including |

| | |
|---|---|
| | visually, quantitatively, and orally. |
| LA.K-12.NJSLSA.SL3 | Evaluate a speaker's point of view, reasoning, and use of evidence and rhetoric. |
| LA.K-12.NJSLSA.SL5 | Make strategic use of digital media and visual displays of data to express information and enhance understanding of presentations. |
| MA.K-12.1 | Make sense of problems and persevere in solving them. |
| MA.K-12.2 | Reason abstractly and quantitatively. |
| MA.K-12.3 | Construct viable arguments and critique the reasoning of others. |
| MA.K-12.5 | Use appropriate tools strategically. |
| MA.K-12.6 | Attend to precision. |
| MA.K-12.7 | Look for and make use of structure. |
| MA.K-12.8 | Look for and express regularity in repeated reasoning. |
| WRK.K-12.P.4 | Demonstrate creativity and innovation. |
| WRK.K-12.P.5 | Utilize critical thinking to make sense of problems and persevere in solving them. |
| WRK.K-12.P.8 | Use technology to enhance productivity increase collaboration and communicate effectively. |
| WRK.K-12.P.9 | Work productively in teams while using cultural/global competence. |

## STUDENT LEARNING TARGETS

### Declarative Knowledge

Students will understand that:

- Data can be stored and access using variables
- Data comes in different types
- Mathematical operations can be performed on data and variables
- Programs cannot be written without data
- Conditional logic is used to control what code is executed
- The simulator is necessary for real-world testing
- Interface builder is used to creating iteration with the use

### Procedural Knowledge

Students will be able to

- Investigate the three features that make Swift a modern language
- Describe why swift is a safe language
- Demonstrate how to use playgrounds to run Swift code
- Demonstrate how to declare variables and constants

- State the limitations on naming constants and variables
- Describe why most variables should be declared as constants
- State the result of assigning a constant or variable to another constant or variable
- List two numeric Swift types (Int and Double)
- Describe type inference
- Demonstrate how to specify the type for a variable or constant
- Demonstrate how to format integer and float values for easier reading”
- Identify the assignment operator (=)
- List the four mathematical operators that Swift supports (+ – * /)
- List the four mathematical compound assignment operators that Swift supports (+= – = *= /=)
- Identify the remainder operator (%)
- Demonstrate how to add two numbers from different types
- Identify and describe the order of operations in a Swift expression)
- Define six common comparison operators in Swift (==, >, >=, <, <=, !=)
- Demonstrate how to compare values to control application flow
- Demonstrate how to use the logical operators (!, &&, ||)
- Describe when to use a switch statement
- Describe how to evaluate a range of numbers in a switch statement
  Demonstrate how to use the ternary conditional operator (?:)
- Describe asset catalogs, capabilities, and deployment target
- Use the filter bar in the project navigator to quickly find files
- Describe the following content areas: navigator, debug, assistant editor, version editor
- List three common keyboard shortcuts in Xcode
- Simulate portrait and landscape orientations
- Use keyboard shortcuts to zoom in and out
- Use basic code signing to run an app on an actual device
- Insert a breakpoint in an app running on an actual device
- Open the documentation viewer for overviews, symbol descriptions, and discussion
- Use resources on the Apple Developer website to find sample code and framework guides
- Describe the advantages of using Interface Builder versus programmatic layout
- Describe the purpose of the initial view controller in a storyboard
- Describe the role of the document outline in Interface Builder
- Demonstrate how to create an IBOutlet and an IBAction using the assistant editor
- Explain the differences between the Identity inspector, Attributes inspector, Size inspector, and Connections inspector
- Demonstrate how to add objects from the Object library to a scene
- Use the Identity inspector, Attributes inspector, Size inspector, and Object library to build basic interfaces
- Create variables of different data types.
- Use variables and operations to control the flow of a program.

## EVIDENCE OF LEARNING

### Formative Assessments

- Guided practice
- Independent practice
- Check Lists
- Exit Ticket
- Class Discussion
- Teacher Observation
- Exit/Entrance Tickets
- Classwork
- Labs

### Summative Assessments

- End of Unit Test
- Performance Task Examples:
    - Flashlight App
    - Picture Frame App
    - WordArt App

## RESOURCES (Instructional, Supplemental, Intervention Materials)

- Developing in Swift Fundamentals  - Teachers Guide
- Developing in Swift Fundamentals - Student Book
- Teacher Resources (slides and answer key to labs and classwork)
- Student Labs and Classwork
- Quick Reference Guide

## INTERDISCIPLINARY CONNECTIONS

Interdisciplinary connections are frequently addressed through modeling existing applications and labs whereby creating solutions and analyzing situations from business, geography,

health/fitness, statistics and numerous other fields.  Examples can be found in topics specific textbook examples, lab practice, guided projects and digital resources.

## ACCOMMODATIONS & MODIFICATIONS FOR SUBGROUPS

See link to Accommodations & Modifications document in course folder.