02 Introduction to UIKit

Content Area: **Math**
Course(s):
Time Period: **Semester**
Length: **6 - 8 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 deepen their skills in Xcode and Interface Builder. Students will explore how to create elements to a screen and connect those elements to code that responds to events generated by the user, Students will learn how to break down problems into smaller more manageable pieces and create algorithms for each piece using loops, conditionals and structures. Students will learn how functions can be created to reuse chucks of code, and explain their understanding of storing and utilizing data with collections and structures. The unit culminates with an application project.

OBJECTIVES, ESSENTIAL QUESTIONS, ENDURING UNDERSTANDINGS

Objectives:

- Choose the appropriate data structure and algorithm for a specific application
- Solve problems using algorithms and data structures
- Programmers use classes and structures to represent physical objects
- Students should be able to identify common features in multiple segments of code and substitute a single function, data structure, or class
- Name the fundamental data structures (strings, collections, structures, arrays, classes, and dictionaries) and when each one should be used
- Implementing programming solutions using some form of structured design with

multiple functions/procedures/modules

- Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
- Decompose complex problems into manageable sub problems that could potentially be solved with functions, classes, collections, structures or procedures that already exist.

Essential Questions

- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- What benefits and drawbacks to creating structures and classes?
- How does Interface Builder and UIKit group different actions together based on user interaction?

Enduring Understanding

- Models and simulations use abstraction to generate new understanding and knowledge.
- Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.
- Control structures should include conditional statements, loops, and event handlers

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.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
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.

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.
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:

- Strings can be both mutable and immutable
- Functions are used as an abstraction tool to simplify code
- Structures are used to make working with complex data and performing logic easier
- Classes can be formed with hierarchy using shared attributes and behaviors
- Structures are value types and classes are reference types
- Variables of different types can be stored in a structure, letting you define a named type
- Classes are used as a tool to manage the complexity of a program
- Collections come in different forms and are used to group and manage variables
- Loops are used to perform logic repeatedly
- Loops come in both indefinite and definite loops
- Interface builder is used to plan out the content for an app
- UIKit enables data entry on touch-based devices

Procedural Knowledge

Students will be able to

- Compare and contrast mutable and immutable strings
- Describe methods to manipulate strings
- List components of a function
- Describe how to return multiple values from a single function
- List the key components of a structure
- Compare and contrast an instance and a type
- Compare and contrast a structure and a class
- Describe how a subclass and a superclass properties and methods are shared
- Describe the difference between let and var
- Determine when to use a structure or a class in a given problem
- Describe how to insert a new value into an array and how to insert a new value into a dictionary
- Compare and contrast a set and an array
- Compare and contrast two looping methods
- Demonstrate how to break out of a loop or skip a loop iteration
- Describe the role UIKit plays in app development
- Demonstrate how to use developer documentation to find out more
- Show how to customize an image view and label
- Show how to use difference controls such as buttons, switches, segmented controls and sliders
- Show how to use stack view to help manage constraints

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:
    - Hangman App
    - Question Bot App
    - Stopwatch App
    - Finger Paint 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.