WEEK ONE: INTRODUCTION TO CODING
http://www.codecademy.com/tracks/afterschool-semester1

Go through what's going to happen during the semester: everyone will be coding websites and games. Introduce Codecademy and make sure everyone has an e-mail address and can register to the site. Emphasize that everyone learns at a different pace, experimentation is welcome, you will learn lots from catching and fixing your mistakes. This is also different from a normal class because you get to build real things while you learn.

Ask students what their goals are, what they already know about technology and the Internet, and what questions they want to get answered by participating in your club. If you have time, let students try the first exercises on the track.

Discussion:

- What is programming?
- What is a browser?
- What is HTML (structure), CSS (presentation), Javascript (behaviour/interaction)
- Where can you see HTML & CSS? View the source code of your school website.
- Where in the everyday world one can see programming?
- What knitting and mathematics have to do with programming?
- What would happen if computers disappeared?
- What will happen in the future as computers get smaller, faster, and cheaper?
- Look up the history of the printing press and how cheap paper manufacturing radically changed access to reading and writing across economic classes. Can you see any parallels to programming and our modern society?

WEEK TWO: HTML FUNDAMENTALS
http://www.codecademy.com/tracks/afterschool-semester1

HTML is the bones under every single web page on the web. You will learn the fundamentals of HTML to make your own basic website. You will include images, organize text, and add links to your page.

Have students start with the first lesson, which covers the following topics:
Structure of HTML - Basic tags - Hyperlinks - Images
Then they can go on to the first **Project:** Build Your First Webpage.

Discussion:

- Look at the source of a favorite website and see if you can find any <p> elements (use Ctrl-F). What other familiar tags can you spot?
- Show students how to use shortcuts to copy-paste code. For Windows it's ctrl + c and ctrl + v, for Mac cmd + c and cmd + v.
- Note that writing HTML and CSS is not technically "programming", but formating. HTML and CSS are called markup languages that the students need to know in order to create a canvas for web applications and programs that you will write in JavaScript.

WEEK THREE: MORE WITH HTML
http://www.codecademy.com/tracks/afterschool-semester1

Learn more about the fundamentals of HTML. You will further organize the material on a webpage using lists and tables, which will be extremely useful down the road. You also learn how to bold, align, and add color to your text!

Topics covered in the lesson: Lists - Styles - Tables

Project: Make a Recipe Card

Discussion: Ask everyone to look at http://w3schools.com/html and choose a tag from the left-hand column to research and present in the next session.

WEEK FOUR: HTML REVISION
http://www.codecademy.com/tracks/afterschool-semester1

Go through what you have learned so far and have students present what kind of sites they have built. Now is a good time to make sure they understand that indentation is very important, both so others can read their code and so they can make sure to catch any errors or unclosed tags.

Make sure students remember at least the following:
<body>.
<p> tags and <h1> .. <h6> headers.
<em>, <strong>.
<img src= ”” >

WEEK FIVE: STYLE WEBPAGES WITH CSS
http://www.codecademy.com/tracks/afterschool-semester1

If HTML is the bones of a webpage, CSS is the skin. It is a powerful tool that lets you style simple pages or complex websites. Integrate HTML and CSS to make a beautifully formatted page.

Topics covered: CSS - Selectors - Different ways to link CSS

Project: First Website Using HTML and CSS

Discussion:

- Learning CSS is a lot of fun: encourage students to experiment with colors, fonts and overall styling to make their pages ugly or beautiful.
- Teach them how to find CSS code, either with Firebug or by saving the website on their computer.
- Send them to CSS Zen Garden, a collection of pages that share exactly the same HTML. Only the CSS is different, but the pages look nothing alike. http://www.csszengarden.com/

WEEK SIX: ADVANCED CSS SELECTORS
http://www.codecademy.com/tracks/afterschool-semester1

You've seen a glimpse of the magic of CSS selectors. Now it's time to grasp the full power and make the Internet pretty once more!

Topics covered: Using selectors and IDs - Inheritance and Cascading - Pseudoselectors

Project: Build a Resume

Discussion: As they build their resume, have the students imagine they're applying for a real job when they are 22.

- If they are younger, have them invent their own personalities with fun and weird backgrounds.
- High school seniors could create a college application website with information about their work and activities.

WEEK SEVEN: INTRO TO CSS POSITIONING
http://www.codecademy.com/tracks/afterschool-semester1

Learn the basics of positioning elements on the page using CSS. Start with the box model. Then float and clear your way to lovely websites. You will learn the box model, which is fundamental to understanding positioning. Be patient as you learn it—it may require some practice to thoroughly learn it. Diagramming your page on paper before you build it can be extremely helpful.

Topics covered: Box model - relative positioning - absolute positioning
Project: Create a Personal Webpage
Discussion:

- Have a short look into the history of computer science: Who is Alan Turing? Ada Lovelace? Tim Berners-Lee? Steve Jobs? Linus Torvalds? James Gosling? Grace Hopper?

WEEK EIGHT: ADVANCED CSS POSITIONING

http://www.codecademy.com/tracks/afterschool-semester1

Build on the previous lesson by exploring absolute and fixed positioning, which are important tools for building any web page. Includes a review of relative and static positioning.

Topics covered: Absolute positioning - fixed positioning

Project: Pizza Time!

Discussion:

- Teach the students how to use Firebug if they are using Firefox. It's a tool that lets you inspect the structure of web pages.
- Adding a style that gives every div a border of 1px is ugly but can help you see what's going on.
- If you're using the Chrome browser, you can right-click and choose "Inspect Element." You will see the page's styles on the right. You can edit the styles right in the browser and see it change live on the page.
- Ask everyone to look at http://w3schools.com/css and choose an attribute to present in the next session.

WEEK NINE: CSS REVISION
http://www.codecademy.com/tracks/afterschool-semester1

Go through what you have learned so far and have students present what kind of sites they have built. Now is a good time to make sure students know how to format CSS stylesheets properly, indenting blocks with two spaces and leaving blank lines in between blocks.

Make sure students remember at least the following:

selector { property : value;}, color, background-color, where to put your CSS and how to link to it from the HTML page

New activity: save your HTML to a file in a text editor that allows you to save the file as plain text (important—saving it as a **Microsoft Word or RTF file will not work)** with the extension .html. Do the same for the CSS. Then open your HTML file in a browser to view your web page!

WEEK TEN: GETTING STARTED WITH PROGRAMMING
http://www.codecademy.com/tracks/afterschool-semester2

An introduction to JavaScript, a friendly programming language. Note that this is their first exposure to programming, which is a completely different paradigm from HTML. Refer students to need to think slightly differently, and be patient as they learn a completely new skill. This lesson will introduce them to some fundamental building blocks, which will be revisited later. Programming is fun and rewarding once you get the hang of it!

Topics covered: Strings - Numbers - Variables - If/Else statements
Project: Make your own adventure game!
Discussion: Make your own flashcards: Collect a list of new terms from every session and make flashcards for students to practice with. You can find help from codecademy.com/glossary. Can you think of any good programming jokes?

WEEK ELEVEN: INTRODUCTION TO FUNCTIONS
http://www.codecademy.com/tracks/afterschool-semester2

Functions store blocks of code that can be called upon any time to avoid repetition. They are one of the most fundamental concepts in all of computer programming, but can be tricky at first.

Topics covered: Functions - Using variables in functions - Returning a value at the end of a function.

Project: Rock, Paper, Scissors

Discussion:

- Remember to review last week's topics: strings, numbers, variables and if/else statements in the beginning of the class.
- Why are functions so useful to programmers?
- What are some useful analogies that can help you understand and explain functions?

WEEK TWELVE: INTRODUCTION TO FOR LOOPS IN JS
http://www.codecademy.com/tracks/afterschool-semester2

Loops are a series of instructions that repeat until a condition is met. This is extremely useful for automating repetitive tasks.

Topics covered: For loops (one type of loop)

Discussion:

- Why are loops so useful?
- What else could you do with a for loop?
- If students are having trouble with this concept, consider acting it out.
- Stepping through a program line by line and seeing how the variables change can be useful for helping students understand how the programs work.

WEEK THIRTEEN: RECAP OF JAVASCRIPT

http://www.codecademy.com/tracks/afterschool-semester2

Go through what you have learned so far and have students present what kind of games they have built. Make sure students remember at least the following:

Topics covered:

Javascript Revision

var myName = “Ryan”;

Output: console.log(), alert(), confirm()

Operators: +*/-%

For loops: for(i=0;i<5;i++) { }

Comparison operators: >, <, >=, <=, ===, !==

Branching: if() { } else if() { } else { }

Tip: Ask a programming-savvy parent or volunteer to visit your classroom and explain what they do in their work. You can also reach out to local software companies their programmers are often willing to share their experience.

WEEK FOURTEEN: PUTTING IT ALL TOGETHER

http://www.codecademy.com/tracks/afterschool-semester1

Celebrate what you’ve learned. Have students complete their own personalized projects.

Project: Code’n’Tell

DEBUGGING YOUR CODE

Finding and fixing mistakes in code is an important skill for any programmer.

General tips

- Error messages can provide clues. If you can't decipher them, you can often Google them.
- A frequent source of mistakes for beginners is syntax—a wayward comma or a missing bracket. Software programs are very picky about every single character, so check them carefully. We run a program called a linter that checks your code for syntax errors, and will display either a red or yellow icon to the left of the problem line. If you hover over the icon, it will tell you with the problem is.
- Logic problems: trace through the program line by line. Speaking the program out loud can really help you spot errors more quickly. Programmers call this rubber duck debugging (http://en.wikipedia.org/wiki/Rubber_duck_debugging).
- Printing out values of your variables at various places in the program can help you reveal this sort of error. Once you get to be a more advanced programmer there are more sophisticated debugging techniques, but this one will really help with simpler programs.

HTML & CSS

- Did you include <body>,<head> and <html> tags?
- Make sure to link the correct CSS stylesheet so the HTML page can find your styles.
- Make sure you remember to close the tags in the right order. They should nest, so the most recent one opened should be the first one closed: {()} rather than {(}).
- Make sure your image titles have the correct name and extension (.jpg, .png).
- Classes and IDs are easy to confuse in CSS. Use a period for classes and a pound sign (#) for IDs.

JavaScript

- Variable names are case-sensitive and even changing one character will refer to a different variable.
- Make sure you used the right sorts of brackets (there are several: (), [], and {}) and that each open bracket is closed.
- Did you include enough semicolons? Are there too many semicolons? If you wrote a function but it's not running, did you call the function?

TROUBLESHOOTING

For fixing your code, see the Debugging section of this document, and for instructional help, see the teacher forum on our website.

Install the most up-to-date version of Firefox or Chrome. Codecademy works most reliably on these two browsers. Internet Explorer versions 6-8 are not supported—please use version 9. iPads are not currently supported but any kind of laptop or desktop computer should work fine.

Check that there are no firewalls that block Codecademy, and that cookies are enabled.

Clear your browser cache and restart the browser. You can clear the browser cache in your browser settings or preferences.

Check that all of your plugins are up-to-date. You don't need to have flash installed to use Codecademy. You can check the status of your plugins for Firefox, Chrome, and Safari here: http://www.mozilla.org/en-US/plugincheck/

For most common issues, see: http://help.codecademy.com/ You can also reach out to us via contact@codecademy.com.