< CodeImpact Teens Program />

Curriculum Guide

Module: Scratch [Programming Language]

Overview

Scratch is a block-based visual programming language and website targeted primarily at children 8-16 to help learn code. With Scratch, you can program your own interactive stories, games, and animations — and share your creations with others in the online community.

Scratch helps young people learn to think creatively, reason systematically, and work collaboratively — which are certainly essential skills for life in the 21st century.

Prerequisites

Before starting this module, you don't need any previous knowledge on using Scratch or any other programming language knowledge, but you should have at least basic familiarity with using computers and using the web passively (i.e., just looking at it and consuming content). You should have a basic work environment set up. Here are some of the setup requirements
eb

1. Install the Scratch offline editor (Download link Scratch Offline Editor)
2. Sign up for a Free Scratch Account
3. A Personal functioning Computer

Section 1: Getting Started

Overview;

Ready to get started? This section is designed for those who are completely new to Scratch. From exploring inspiring projects, to creating a Scratch account, to having an initial experience playing with the Scratch project editor, each activity is designed to guide you and your students through the process of getting started with Scratch.

In this section, students are introduced to computational creation with the Scratch programming environment by viewing a collection of sample projects and engaging in an exploratory, hands on experience.

Learning Objectives;

The students will:

- understand the concept of computational creation, in the context of Scratch
- Be able to imagine possibilities for their own Scratch-based computational creation
- Become familiar with resources that support their computational creation

Session Activities:

- Introduce the concept of computational creation and the Scratch environment
- Show sample Scratch projects
- Review design processes
- Explore the Scratch interface

Resources

- Animate your Name
- Scratch Overview Video

Section 2: Exploring

Overview;

This section includes a mix of structured and open-ended activities that engage students in exploration of the key concept of sequence – identifying and specifying an ordered series of instructions. This is often a powerful moment for students: they're telling the computer what to do, by translating their ideas into blocks of computer code.

Students build on their initial explorations of the Scratch environment by creating an interactive project.

Learning Objectives;

The students will:

- become familiar with a wider range of Scratch blocks
- be able to create a Scratch project that is an interactive digital representation of their interests.

Session Activities:

- students explore and experiment with sprites, costumes, looks, backdrops, and sounds to create an interactive collage in Scratch.
- students will follow a tutorial in order to build up a program by experimenting and iterating.
- students will learn to express a complex activity using a sequence of simple instructions.

Resources

- Exploring Scratch
- Experimenting with Sprites

Module: HTML [HyperText Mark-Up Language]

Overview

At its heart, HTML is a fairly simple markup language for creating Web pages,HTML stands for HyperText Markup Language. It describes the structure of a Web page as it consists of a series of elements. HTML elements tell the browser how it should display content such as images, video, text etc. This module will introduce fundamental concepts and syntax you need to know in order to understand HTML.

Prerequisites

Before starting this module, you don't need any previous HTML knowledge, but you should have at least basic familiarity with using computers and using the web passively (i.e., just looking at it and consuming content). You should have a basic work environment set up. Here are some of the setup requirements
eb

4. Install a code editor (preferably Visual Studio Code)
5. Get a FreeCodeCamp Account
6. A Personal functioning Computer
7. Modern Web Browser
    - Google Chrome, FireFox

Section 1: Basic HTML web Structure

Overview;

In this section, we'll cover the whole range of HTML so you'll be completely comfortable with putting the right elements in the right places on a page.

Learning Objectives;

// objectives

To gain basic familiarity with HTML, and practice writing a few HTML elements.

To learn about the HTML head, its purpose, the most important items it can contain, and what effect it can have on the HTML document.

Learn how to mark up a basic page of text to give it structure and meaning – including paragraphs, headings, lists, emphasis, and quotations.

Content:

1. Structure of a web page
2. Page Titles
3. Paragraphs
4. Headings
5. HTML Lists (ordered and Unordered)
6. Emphasis Elements
7. Empty HTML Elements

Section 1 Challenge

// Resources: W3Schools, MDN, FreecodeCamp

Section 2: Links And Images

Overview;

The previous section covered some very important HTML elements, but we were only dealing with a single web page. Links and images are fundamentally different from those elements in that they deal with external resources. Links point the user to a different HTML document, and images pull another resource into the page.

Learning Objectives;

// objectives

To learn how to implement a hyperlink effectively, and link multiple files together.

Content:

1. Anchor Tags
2. Links (Absolute, Relative and Absolute-Relative links)
3. Link Targets
4. Image Tags
5. Image Formats
6. Text Alternatives
7. HTML Entities

Section 2 Challenge

// Resources: W3Schools, MDN, FreecodeCamp

Section 3: HTML Forms

Overview;

HTML form elements let you collect input from your website's visitors. Mailing lists, contact forms, and blog post comments are common examples for small websites, but in organizations that rely on their website for revenue, forms are sacred and revered. In This section we shall understand how to create them.

Learning Objectives;

// Objectives

To gain familiarity with what HTML web forms are, what they are used for, how to think about designing them, and the basic HTML elements you'll need for simple cases.

To understand how to structure HTML forms and give them semantics so they are usable and accessible

To understand in detail the original set of native form widgets available in browsers for collecting data, and how to implement them using HTML

Content:

1. Input Fields (Text and Email inputs)
2. Radio Buttons
3. Drop Down Menus
4. Text Areas
5. Checkboxes
6. Submit Button

Section 3 Challenge

// Resources: W3Schools, MDN, FreecodeCamp

Section 4: Semantic HTML

Overview;

HTML form elements let you collect input from your website's visitors. Mailing lists, contact forms, and blog post comments are common examples for small websites, but in organizations that rely on their website for revenue, forms are sacred and revered. In This section we shall understand how to create them.

Learning Objectives;

// objectives

Learn how to structure your document using semantic tags, and how to work out the structure of a simple website

Content:

1. Document Outline
2. Articles
3. Sections
4. Nav Elements
5. Headers and Footers
6. Divs For Layout
7. Asides
8. Figures and Captions

Section 4 Challenge

// Resources: W3Schools, MDN

< CodeImpact Teens Program />

Module: CSS [Cascading Style Sheets]

Overview

CSS (Cascading Style Sheets) is used to style and lay out web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module provides a gentle beginning to your path towards CSS mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML.

Prerequisites

Before starting this module, you should have at least basic familiarity with using computers and using the web passively (i.e., just looking at it and consuming content). You should have a basic work environment set up. Here are some of the setup requirements

1. Install a code editor (preferably Visual Studio Code)
2. Get a FreeCodeCamp Account
3. A Personal functioning Computer
4. Modern Web Browser
    - Google Chrome, FireFox

You should have basic familiarity with HTML, as discussed in the HTML module.

Section 1: Basic CSS

Overview;

The first Module focused exclusively on HTML. Now, it's time to make things pretty (sort of) with Cascading Style Sheets (CSS). You can think of CSS as defining the "design" of a web page. It determines things like font size, margins, and colors using a language entirely separate from HTML.

Learning Objectives;

// objectives

To understand the basics of linking a CSS document to an HTML file, and be able to do simple text formatting with CSS

Content:

1. CSS Style Sheet
2. Linking a Stylesheet
3. CSS Comments
4. Setting Multiple Properties and Elements
5. Selecting Different Elements
6. Units Of Measurement
7. List Styles
8. Reusable StyleSheets
9. Text Styles
10. The Cascade hierarchy

Section 1 Challenge

// Resources: W3Schools, MDN, FreeCodeCamp

Section 2: CSS Box Model

Overview;

Everything in CSS has a box around it, and understanding these boxes is key to being able to create layouts with CSS, or to align items with other items. In this section, we will take a proper look at the CSS Box Model so that you can build more complex layout tasks with an understanding of how it works and the terminology that relates to it.

Learning Objectives;

// objectives

To learn about the CSS Box Model, what makes up the box model and how to switch to the alternate model.

Content:

1. Block and Inline Elements
2. Changing Box Behaviour
3. The Box Model
4. Borders
5. Paddings
6. Margins
7. Vertical Margin Collapse
8. Generic Boxes
9. Content vs Border Boxes
10. The Cascade hierarchy

// Resources: W3Schools, MDN

Section 3: CSS Selectors

Overview;

In CSS, selectors are used to target the HTML elements on our web pages that we want to style. There are a wide variety of CSS selectors available, allowing for fine-grained precision when selecting elements to style. In this module and its sub-topics we'll run through the different types of selectors in great detail, seeing how they work.

Learning Objectives;

// objectives

To learn how CSS selectors work in detail.

Content:

1. Class Selectors
2. Container Divs
3. Reusing Class Styles
4. Modifying Class Styles
5. Descendant Selectors
6. Pseudo-Classes for links
7. Pseudo-Classes for Buttons
8. Pseudo-Classes for Structure
9. ID Selectors
10. CSS Specificity

// Resources: W3Schools, MDN

Section 4: CSS Layouts [Floats]

Overview;

Originally for floating images inside blocks of text, the float property became one of the most commonly used tools for creating multiple column layouts on webpages. With the advent of flexbox and grid it has now returned to its original purpose, as this section explains.

Learning Objectives;

// objectives

To learn how to create floated features on webpages, and to use the clear property and other methods of clearing floats.

Content:

1. Default Layout Behaviour
2. Floating an Element
3. Floating Inside of Parents
4. Multiple Floats
5. After a Float
6. Full-Bleed Layouts
7. Equal Width Columns
8. Floats For Grids
9. Floats for Content

// Resources: W3Schools, MDN

Section 5: CSS Layouts [Flexbox]

Overview;

The “Flexible Box” or “Flexbox” layout mode offers an alternative to Flexbox for defining the overall appearance of a web page. Whereas floats only let us horizontally position our boxes, flexbox gives us complete control over the alignment, direction, order, and size of our boxes.

Learning Objectives;

// objectives

To learn how to use the Flexbox layout system to create web layouts.

Content:

1. Flexbox Overview
2. Flex Containers
3. Aligning a Item
4. Distributing And Grouping Flex Items
5. Cross-axis Alignment
6. Wrapping Flex Items
7. Flex Container direction and Order
8. Flex item order and Alignment
9. Flexible Items
10. Auto-Margins

// Resources: W3Schools, MDN

Section 6: CSS Layouts [Advanced Positioning]

Overview;

Positioning allows you to take elements out of the normal document layout flow, and make them behave differently; for example sitting on top of one another, or always remaining in the same place inside the browser viewport. This section explains the different position values, and how to use them.

Learning Objectives;

// objectives

To learn how CSS positioning works

Content:

1. Positioned Elements
2. Relative Positioning
3. Absolute Positioning
4. Fixed Positioning
5. Positioning For Menus
6. Inline Menu Items and Submenus
7. Absolute Submenus
8. Z-Index

// Resources: W3Schools, MDN

Section 7: CSS Responsive-Design

Overview;

“Responsive design” refers to the idea that your website should display equally well in everything from widescreen monitors to mobile phones. It’s an approach to web design and development that eliminates the distinction between the mobile-friendly version of your website and its desktop counterpart. With responsive design, they’re the same thing.

Learning Objectives;

// objectives

To understand the fundamental concepts and history of responsive design.

Content:

1. CSS Media Queries
2. Understanding Design
3. Mobile-First Development
4. Tablet layout
5. Desktop Layout
6. Modern Layout Technologies (CSS grid, Flexbox)
7. Responsive Images
8. Responsive Typography
9. The Viewport meta tag

// Resources: W3Schools, MDN