Simple Simon Game

-- Simon Introduction --

+

+ As a pair, develop a basic version, then a unique take, on the Simple Simon game (based on 80s Simon board game)

+

+ Examples

+

+ Up next...

+

+ 1) Set up enviroment

+ 2) Plan out all rules in great detail for basic version and pseudo code project

+ 3) Pair program minimum viable product (MVP)

+ 4) Pair program and style unique take on the Simple Simon game

+

+

+-- Expectations --

+

+ - Work together as a team (not as two individuals)

+ Together, plan the project

+ One person write code (driver)

+ One person should observe, strategize, give feedback/suggestions (navigator)

+

+ - Alternate frequently between driving/navigating about every 30 minutes

+

+ - Don't use eval()

+

+ - Commit frequently (entire project should be at least 30 commits)

+

+

+-- Suggestions --

+

+ - Use jQuery syntax where possible

+

+ - Work first to complete fully functional version before extra styling and features

+

+

+-- Project Set Up --

+

+ Pair up

+

+ BOTH team members do the following...

+

+ 1) Follow curriculum setup steps 1 - 3 (NOT INCLUDING STEP 4!)

+

+ First team member ONLY...

+ 1) Create organization on GitHub

+ 2) Add teammate to organization

+ 3) Complete step 4 from curriculum setup steps

+ 4) Add and commit index.html file

+ 5) Create remote repo on GitHub based on your local repo

+ 6) Push up all commits to the newly created remote repository on GitHub

+

+ Second team member ONLY...

+ 1) Go to your organizations page and simple-simon repo

+ 2) Click the link to clone the repo address

+ 3) cd into ~/vagrant-lamp/sites/simplesimon.dev folder

+ 4) Type "rm -rf public"

+ 5) Hit enter

+ 6) Type "git clone [paste address here] ." (no actual square brackets should be around the address and don't forget the dot)

+ 7) Type "subl .gitignore" and hit enter then save and close the empty file

+ 8) Add and commit .gitignore file

+ 9) cd into the public folder and use the "mkdir" to create folders for css, js, img, and sound

+ 10) Add and commit these folders

+ 11) Create main.css and main.js files

+ 12) Add and commit these files

+ 13) Once all additional files and folders are added and committed, type "git push origin master" and hit enter

+

+ First team member ONLY...

+ 1) Type "git pull origin master" and hit enter

+ 2) Make a small change to the index.html file like adding the basic tags of an HTML script

+ 3) Add and commit the change to index.html

+ 4) Type git push origin master

+

+ Second team member ONLY...

+ 1) Type "git pull origin master" and hit enter

+ 2) You are now free to begin planning the project

+

+

+

+-- Project Planning --

+

+ As a pair, write out the rules for Simple Simon in great detail.

+

+ Pseudo code as detailed as possible the logic of your project (remember to focus on the basic version first).

+

+ Aim to spend at least an hour planning the project.

+

+ Consider extra features and styling (a possible theme) for the unique version of the project.

+

+ Once the project is adequetly planned, begin pair programming, using the curriculum steps as a guide.

**Steps**

To build this game, follow these steps:

Modes:

1. Strict mode – if player makes error at any point in the game, they start over from the beginning
2. Regular mode – player has 3 chances to get a correct response at the previous level before restarting

Simple Simon Rules:

1. One color lights up with a sound
2. Player select the same color to move forward
3. The original color plus a new color lights up
4. The player has to match the color sequence
5. With each correct play, the game will repeat and add color sequence - up to 20 rounds before player is declared winner

Curriculum Steps

1. Create your HTML markup and CSS to position your square or shape. These will be the colors that are randomly selected for the game.
2. Randomly select a square and fade that color in then out.
3. Allow the user to click on the square that was selected.
4. Continue randomly selecting colored square/shapes adding the new random selection to be added to the previous selection. Eventually you will end up with a random sequence of selected colors.
5. Each time a new color is added to the sequence allow the user to enter (click) the sequence in the order as it was played.
6. If the user continues to get the order correct then proceed to adding another color to the collection until the user gets the order incorrectly.
7. Keep track of how many rounds the user is able to go.

Planning Notes:

HTML Markup & CSS:

* 4 squares with color & sound:
  + top left – red
  + bottom left – green
  + top right – blue
  + bottom right – yellow
    - Use divs & input boxes to start
    - Empty divs as placeholders for the sound file
* Start button to begin the game
* HTML tag to display the current round
* HTML tag to display instruction
* Add radio buttons to select strict or regular mode (extra styling)

JS Functionality

* Begin in regular mode,
* Click event button to start game
* Each box needs to light up & pause for user input
* Boxes light up randomly, beginning in iterations of 1 movement
* Will need a click function to capture the input & validate against the game’s random sequence
  + Function for random sequence animation
  + Memory Function: to keep track & store of the color sequence, user input & rounds
  + Validation function to compare the user input to the games auto generated sequence
    - Function will need to enable multiple color inputs but know when to stop for the validation (increment
* Sound event listener for each:
  + Color
  + Game won
  + Game lost
* Display message (If Else?)
  + Instructions
    - Click start to begin
    - Watch the sequence
    - Repeat the sequence you just saw
  + Round
    - (Label) Current
    - Round (number) – the number will reiterate as the player progresses through the game – it will need to be stored and returned as the game validates correct returned sequence
* User input to match the random output in same sequence

Conditions or considerations:

* + If error = stop the game
  + If correct = play previous sequence & add new color
  + Stop iteration at 20 plays

JS STRUCTURE PLANNING:

* Define the variables
  + var colors = [red, green, blue, yellow] (an array)
  + var rounds = 1-20
* Functions
  + Take array and randomly display with iteration
    - For (var i = 0; i < 21; i++)

color +=

7/12/2017 – PENDING TASKS

Button flash reiteration without requiring start button pushed each time

Allow user input

Validation/comparison of the user input

Add audio files

Alert for rounds – update “count” field

“Game over” status message on incorrect entry

Button light up on user input

Start disabled/enabled after game play start

Add limit of 20 rounds to declare “Winner”

Increase speed after 10 rounds

background image or color

button styling

add something visually fun to styling