Lab 0 JS OOP (Classes) , Coding Style, Simple Memory game (updated Jan 6th)

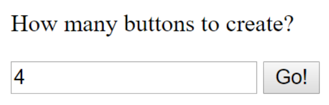
Hide Folder Information

Instructions

Create an interactive memory game using object-oriented JavaScript Classes that generates 3 to 7 buttons with random colors, scrambles their positions, and challenges the user to recall their original order. The game must read the browser window sizes before each time it scrambles the buttons, ensuring that they land within the new browser window dimensions

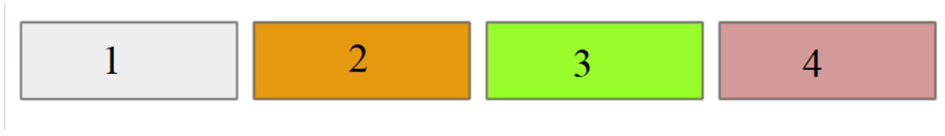
Details:

Display an input box with a message "How many buttons to create? " which gets a number, n, from user between 3 and 7(input validation is needed).


How many buttons to create?

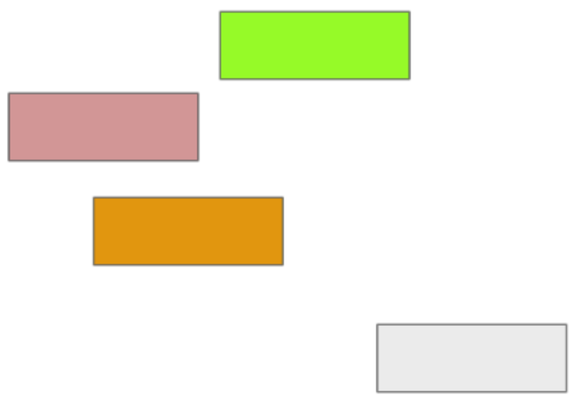
4 Go!

Pressing "Go" shall create n buttons with random colors on screen with height 5em and width 10em , next to each other in same row (unless the width of browser window does not allow)



after n seconds of pause (e.g. for 3 buttons, 3 seconds ... for 7 buttons 7 seconds etc), scramble the buttons to a new random location within browser window n times with time interval of two seconds in between . So for 3 buttons, after you display them in a row, you pause for 3 seconds. Then move them around within browser window every two second for three times. Make sure each time before moving them around utilize the current window size so buttons don't land outside window.

Then, hide the numbers from the buttons, make the buttons clickable now and and let the user test their memory to remember their orders.



Then the user has to remember the order in which they originally appeared and to click them in the same order. If the user remembered the order and clicked all of them in same order, display message "Excellent memory!"

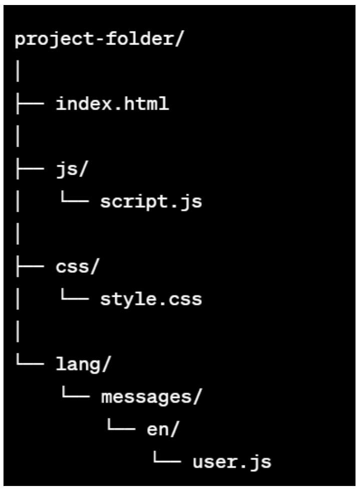
Each time the users clicks on a button in correct order, confirm the order of that button by revealing the number on it. Otherwise, the moment the user clicked a button in wrong order, display "Wrong order!" message and reveal the correct order of all buttons and game ends.

Note:

Store js file in \js\script.js

store the messages you display to user in a separate file

store the messages you display to user in a separate file \lang\messages\en\user.js



```
project-folder/
|
├── index.html
|
├── js/
|   └── script.js
|
├── css/
|   └── style.css
|
└── lang/
    └── messages/
        └── en/
            └── user.js
```

Pressing Go starts a new game. It shall remove any existing buttons from the screen and memory.

Unless the scrambling of buttons is not completed, do make the buttons clickable

For the message "How many buttons to create?" create a separate label.

When placing buttons at random locations, its ok to overlap

Follow object oriented practices and use JavaScript Object (like in image below) to create classes old style or use JS Classes (which is more modern and preferred).

Submit   Cancel

```javascript
/*Note: this code snippet is gvingn you some tips to start
you cannot simply copy this code.
Also your code has to be more modular create more functions
*/
let arrayButtons = [];
function Button(color, width, height, top, left, order) {
    this.order = order;
    this.btn = document.createElement("button");
    this.btn.style.backgroundColor = color;
    this.btn.style.width = width;
    this.btn.style.height = height;
    this.btn.style.position = "absolute";
    document.body.appendChild(this.btn);
    // A method to set location
    this.setLocation = function (top, left) {
        this.btn.style.top = top;
        this.btn.style.left = left;
    };
    /* we call this method to set original
    top,left during creation of the object*/
    this.setLocation(top, left);

}
arrayButtons.push(new Button("Red", "100px", "100px", "0px", "0px", 0));
arrayButtons.push(new Button("Blue", "200px", "100px", "200px", "200px", 1));
// this has to be in a separate function, e.g. moveRandom ... or something
setInterval(function () {
    arrayButtons[0].setLocation(
        Math.floor(Math.random() * 100) + "px",
        Math.floor(Math.random() * 100) + "px");
}, 500);
```

CODING STYLE for the ENTIRE COURSE: (updated jan 6th)

1. Use Object-Oriented Programming JavaScript (OOP JS)

Must write the entire assignment using **JavaScript classes**, not procedural code.

2. Minimum of Three Classes (for this assignment)

Each class must:

- Have its own **constructor**
- Define **properties**
- Include **methods**
- Be instantiated to form **visual components** or **logical modules**

Examples of valid classes:

- `AppController`
- `UserInterface`
- `TaskManager`
- `GameEngine`
- `ScoreBoard`
- `FormValidator`

3. No Hard-Coded User-Facing Strings

All strings visible to the user must be stored in a **separate JS file**, for example:

- `strings.js`

This includes:

- Button labels
- Error messages
- Titles
- Prompts
- UI text

4. Use the Strings File in Your Classes

Classes must import or reference the strings from the external file instead of embedding text directly (no user facing string literals = messages you display to users)

5. Code Must Be Modular and Organized

- Each class should handle **one responsibility**.
- Instances of classes should interact cleanly.
- No global variables except the imported strings.(cost, let and never var)

Short Example (Structure Only)

strings.js

```
export const STRINGS = {
    APP_TITLE: "My OOP App",
    BTN_START: "Start",
    ERROR_EMPTY: "Input cannot be empty"
};
```

AppController.js

```
import { STRINGS } from "./strings.js";

export class AppController {
    constructor(ui, manager) {
        this.ui = ui;
        this.manager = manager;
    }

    start() {
        this.ui.showMessage(STRINGS.APP_TITLE);
        this.manager.initialize();
    }
}
```

UserInterface.js

```
import { STRINGS } from "./strings.js";

export class UserInterface {
    constructor(rootElement) {
        this.root = rootElement;
    }

    showMessage(msg) {
        this.root.textContent = msg;
    }

    createStartButton() {
        const btn = document.createElement("button");
        btn.textContent = STRINGS.BTN_START;
        return btn;
    }
}
```

TaskManager.js

```
import { STRINGS } from "./strings.js";

export class TaskManager {
    constructor() {
        this.tasks = [];
    }

    addTask(text) {
        if (!text) {
            throw new Error(STRINGS.ERROR_EMPTY);
        }
        this.tasks.push(text);
    }

    initialize() {
        console.log("TaskManager ready");
    }
}
```

Deliverable:

1-zip your Lab 0 files from the root directory and put them all in one zip file: yourLastName0.zip e.g. GreenLab0.zip

2- Post the url (the root of lab0 url with https at comment section in similar format as
httpS://yourDomainName.xyz/COMP4537/labs/0/

Additional deductions

-1 if buttons leave browser window during the n*2 second juggling of buttons

-2 if you do not follow OOP and not use object constructor or JS Classes in your code to create buttons

* For this assignment you need to define at least three classes. Each class has to have its own properties and methods. Empty classes will get you 0 marks

(-1 to -3) Your JavaScript code has to be in separate js files (no JavaScript coding inside your HTML file), CSS(if applicable) also in its separate CSS file (only external styling is allowed), otherwise marks will be deducted (this line has been updated on Jan 11 for clarity)

(-1) for not following the naming convention/ directory structure for your deliverable;

(-1) for any extra file (e.g. none of these acceptable: this file was here from PHP, or node and I forgot to delete it, was put by IDE etc)

(-1) for any server side scripts (you don't need any server side script for this)

-10% for each day late

(-1 to -4) for not posting the url of your hosted assignment with http**s**// at the comments section before hitting submit, otherwise your assignment will be considered not hosted!

-1 if you use http instead of httpS

*if you are using http**s**! with 's'. No warning security messages must be observed by the visitor of your website otherwise you will lose 4 points for not hosting properly*

(-1) For wrong variable declaration. You cannot use var! use const if you don't need to change the variable's value later, otherwise use let for variable declaration.

(-1) If a variable is not supposed to be global, do not declare it global.

(-1) Do not use hard coded strings* all over the place in your code. Move them all to the top and declare them with const string variables and use those variables in your code instead .

Its ok to use chatGPT in assignments and term project provided
- You disclose using chatGPT inside your code (js file etc), otherwise -2 deduction
- and at the learning hub comment section when submitting your assignment otherwise -2 deduction
It is mandatory for you to comprehend each line of the code you have submitted. You will be questioned about it during marking which is done in-person, and for every incorrect answer, 1-3 marks will be deducted.

Due on Jan 13, 2026 11:59 PM

Available on Jan 6, 2026 12:01 AM. **Access restricted before availability starts.**

Submit Assignment

Files to submit

(0) file(s) to submit

After uploading, you must click Submit to complete the submission.

Add a File **Record Audio** **Record Video**

Comments