臺北科技大學TAIPEI TECH
互動設計系
Department of Interaction Design

National Taipei University of Technology, Tz-heng Fu (傅子恒)

Course Overview

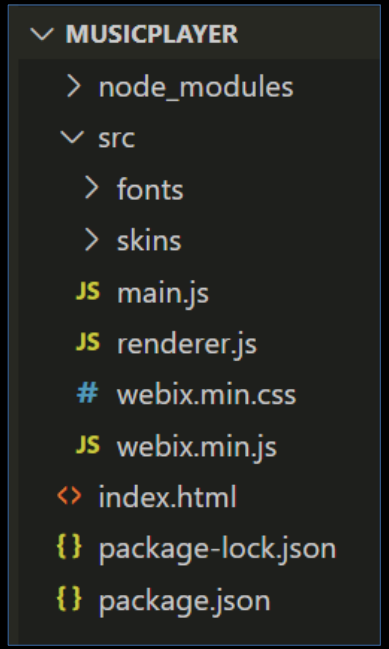
- Project-based learning
  - Implement Music Player logics
  - Remake everything in MVC pattern

Electron App Exercise #04

- Music Player
- MVC Music Player
  - MusicPlayerView
  - MusicPlayerController
  - MusicPlayerModel
- Discussions

Project Initialization

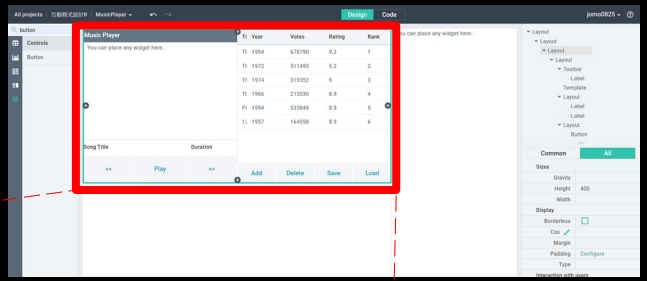
- Create a new project folder
  - Create a new folder called "MusicPlayer".
  - Open the folder in VSCode.
- Install packages
  - Dive into the folder and run the following commands in CMD:
    - npm init -y
    - npm install --save webix
    - npm install –save-dev electron
- Create folders and import assets
  - Copy the fonts/ and skins/ from node_modules/webix folder to src/
  - Copy the webix.min.css and webix.min.js from node_modules/webix folder to src/
  - Create the index.html, mystyle.css, main.js and renderer.js as empty files.
- Edit the package.json configuration file.




```
{
  "name": "musicplayer",
  "version": "1.0.0",
  "description": "Music player",
  "main": "src/main.js",
  ▷Debug
  "scripts": {
    "start": "electron ."
  },
  "keywords": [],
  "author": "Mr.Fu",
  "license": "ISC",
  "dependencies": {
    "webix": "^11.0.0"
  },
  "devDependencies": {
    "electron": "^34.1.1"
  }
}
```

UI Webix View Design

- Create an UI View in Webix Designer.



Toolbar

Music Player

You can place any widget here..

Reserved Area

Label

Song Title

Duration

<< Play >>

Button

Ti	Year	Votes	Rating	Rank
Tl	1994	678790	9.2	1
Tl	1972	511495	9.2	2
Tl	1974	319352	9	3
Tl	1966	213030	8.9	4
Pu	1994	533848	8.9	5
12	1957	164558	8.9	6

Datatable

Add Delete Save Load

Button

index.html

- index.html
  - Set a title.
  - Include css and js files.
    - Rememer to import render.js with type="module". This allows renderer.js to use ES6 import and export.
- Try to memory this boilerplate and type by yourself.

```
1   <html>
2
3   <head>
4       <title>Todo List App</title>
5       <link rel="stylesheet" href="src/webix.min.css">
6       <link rel="stylesheet" href="src/mystyle.css">
7   </head>
8
9   <body>
10      <script src="src/webix.min.js"></script>
11      <script src="src/renderer.js" type="module"></script>
12  </body>
13
14  </html>
15
```

mystyle.css

- Avoid user to select any text in body.

```css
body {
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
```

WebixUI.js

```
export function CreateWebixUI() {
    webix.ui(
        {
            "cols": [
                {
                    "rows": [
                        {
                            "height": 400,
                            "cols": [
                                {
                                    "rows": [
                                        {
                                            "css": "webix_dark",
                                            "view": "toolbar",
                                            "cols": [
                                                { "view": "label", "label": "Music Player", "height": 0 }
                                            ],
                                            "height": 36
                                        },
                                        {
                                            "view": "template",
                                            "template": "You can place any widget here..",
                                            "role": "placeholder",
                                            "id": "reserved",
                                            "height": 250
                                        },
                                        {
                                            "cols": [
                                                { "label": "Song Title", "view": "label", "height": 0, "id": "songTitle", "width": 0 },
                                                { "label": "Duration", "view": "label", "height": 0, "id": "songDuration", "width": 122 }
                                            ],
                                            "height": 40
                                        },
                                        {
                                            "cols": [
                                                { "label": "Prev", "view": "button", "height": 0, "id": "btnPrev" },
                                                { "value": "Play", "view": "button", "height": 0, "id": "btnPlay" },
                                                { "label": "Next", "view": "button", "height": 0, "id": "btnNext" }
                                            ]
                                        }
```

- Again, we export an CreateWebixUI function to create Webix UI.
  - Assign "id" for key components
    - songTitle, songDuration
    - btnPrev, btnPlay, btnNext
    - datatable
    - btnAdd, btnDelete
    - reserved
  - Explicitly add an id field in datatable. This helps to assign id as numbers instead of string.
  - Set select to true for datalist in order to select table items.
  - In order to change the label shown on btnPlay, we need to set its label value as "value" otherwise the label could not be changed by any means. This is a bug for Webix.

```
                                ],
                            },
                            {
                                "width": 450,
                                "rows": [
                                    {
                                        "columns": [
                                            { "id": "id", "header": "ID", "hidden": true, "type": "number"},
                                            { "id": "title", "header": "Title", "fillspace": true, "sort": "string" }
                                        ],
                                        "view": "datatable",
                                        "height": 0,
                                        "id": "datatable",
                                        "select": true
                                    },
                                    {
                                        "height": 53,
                                        "cols": [
                                            { "label": "Add Song", "view": "button", "height": 0, "id": "btnAdd"},
                                            { "label": "Delete Song", "view": "button", "height": 0, "id": "btnDelete" },
                                            { "label": "Save List", "view": "button", "height": 0, "id": "btnSave"},
                                            { "label": "Load List", "view": "button", "height": 0, "id": "btnLoad" },
                                        ]
                                    }
                                ]
                            }
                        ]
                    },
                    { "view": "template", "template": "You can place any widget here..", "role": "placeholder", "id": 1739416976455, "height": 0 }
                ],
                "width": 800
            },
            { "view": "template", "template": "You can place any widget here..", "role": "placeholder" }
        ]
    }
);
}
```

MusicPlayerView.js

```
import { CreateWebixUI } from "./WebixUI.js";
import { MusicModel } from "./MusicPlayerModel.js";

export class MusicPlayerView {
    hiddenInput;
    audio
    constructor() {
        CreateWebixUI();

        // Attach and invisible file input element.
        this.hiddenInput = document.createElement('input');
        this.hiddenInput.id = "fileInput";
        this.hiddenInput.type = 'file';
        this.hiddenInput.accept = '.mp3';
        this.hiddenInput.multiple = true;
        this.hiddenInput.style.display = 'none';
        document.body.appendChild(this.hiddenInput);

        // Inject element into reserved area.
        $$("reserved").setHTML("<div>RESERVED.</div>");

        // Create an audio element.
        this.audio = new Audio();

        // Test resources and cases.
        let testItem1 = new MusicModel(111, "Party song I", "http://777", 123);
        let testItem2 = new MusicModel(222, "Party song II", "http://777", 230);
        let testItem3 = new MusicModel(333, "Party song III", "http://777", 178);
        this.AddItem(testItem1);
        this.AddItem(testItem2);
        this.AddItem(testItem3);
        this.SelectItem(333);
        console.log(this.GetNextId());
        this.DeleteSelectedItem();
        this.SetSongTitle(testItem1.title);
        this.SetSongDuration(Math.floor(testItem1.duration / 60), Math.floor(testItem1.duration % 60));
```

- Create an invisible file input button for file selection.
- Inject template content for reserved area.
- Create Audio instance to play music.
- Put your test mp3 file in src/assets folder

```
        $$("btnPlay").attachEvent("onItemClick", () => {
            const state = $$("btnPlay").$getValue();
            if (state === "Play") {
                $$("btnPlay").setValue("Pause");
                this.audio.src = "./src/assets/Koto G.mp3";
                this.PlayAudio();
            }
            else {
                $$("btnPlay").setValue("Play");
                this.PauseAudio();
            }
        });
    }
```

```
51      // Methods for list view.
52      AddItem(item) {
53          $$("datatable").add(item);
54      }
55
56      GetItem(id) {
57          return $$("datatable").getItem(id);
58      }
59
60      SelectItem(id) {
61          $$("datatable").select(id);
62      }
63
64      DeleteItem(item) {
65          $$("datatable").remove(item.id);
66      }
67
68      DeleteSelectedItem() {
69          return this.DeleteItem(this.GetSelectedId());
70      }
71
72      GetItemCount(){
73          return $$("datatable").count();
74      }
75
76      GetSelectedId() {
77          return $$("datatable").getSelectedId(false, true);
78      }
79
80      GetNextId() {
81          const selectedIndex = $$("datatable").getIndexById(this.GetSelectedId());
82          let nextIndex = selectedIndex + 1;
83          if (nextIndex === this.GetItemCount()) {
84              nextIndex = 0; // If it reaches the last row, loop back to the first row
85          }
86          return $$("datatable").getIdByIndex(nextIndex);
87      }
88
```

```
    GetPrevId() {
        const selectedIndex = $$("datatable").getIndexById(this.GetSelectedId());
        let prevIndex = selectedIndex - 1;
        if (prevIndex === -1) {
            prevIndex = $$("datatable").count() - 1; // If it reaches the last row, loop back to the first row
        }
        return $$("datatable").getIdByIndex(prevIndex);
    }

    // Methods for form view.
    SetSongTitle(title) {
        $$("songTitle").setValue(title);
    }

    SetSongDuration(duration, total) {
        const dur = `${Math.floor(duration/60).toString().padStart(2, 0)}:${Math.floor(duration%60).toString().padStart(2, 0)}`;
        const tol = `${Math.floor(total/60).toString().padStart(2, 0)}:${Math.floor(total%60).toString().padStart(2, 0)}`;
        $$("songDuration").setValue(`${dur} / ${tol}`);
    }

    SetPlayButtonLabel(value) {
        $$("btnPlay").setValue(value);
    }

    GetPlayButtonLabel() {
        return $$("btnPlay").getValue();
    }

    LoadAudioURL(url) {
        this.audio.src = url;
    }

    PlayAudio() {
        this.audio.play();
    }

    PauseAudio() {
        this.audio.pause();
    }
}
```

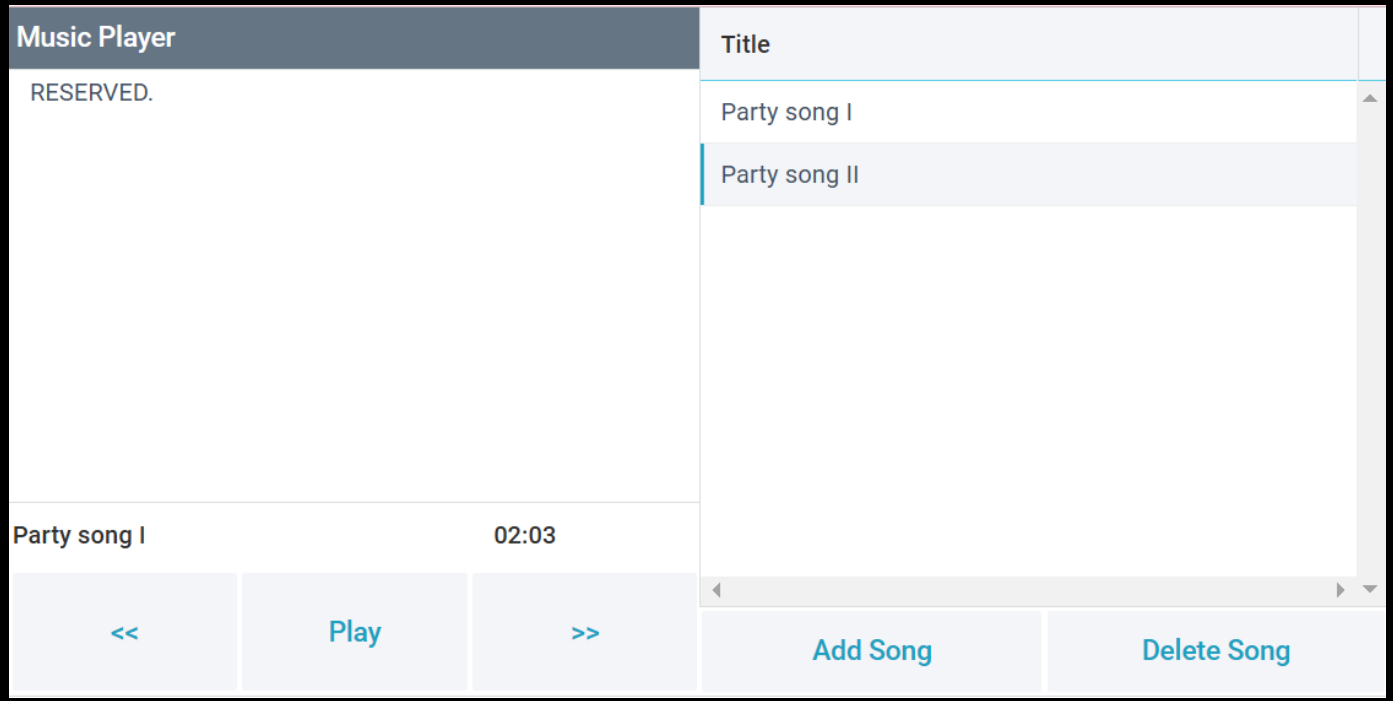
renderer.js

- Instantiate a MusicPlayerView for testing.

```
1  import { MusicPlayerView } from "./MusicPlayerView.js";
2
3  let view;
4
5  window.addEventListener("load", ()=>{
6          view = new MusicPlayerView();
7  });
```

Test

- Test. Press play to hear your mp3 audio.
- Remember to put you mp3 in src/assets folder.



Electron App Exercise #04

- Music Player
- MVC Music Player
  - MusicPlayerView
  - MusicPlayerController
  - MusicPlayerModel
- Discussions

MusicPlayerController.js

- a

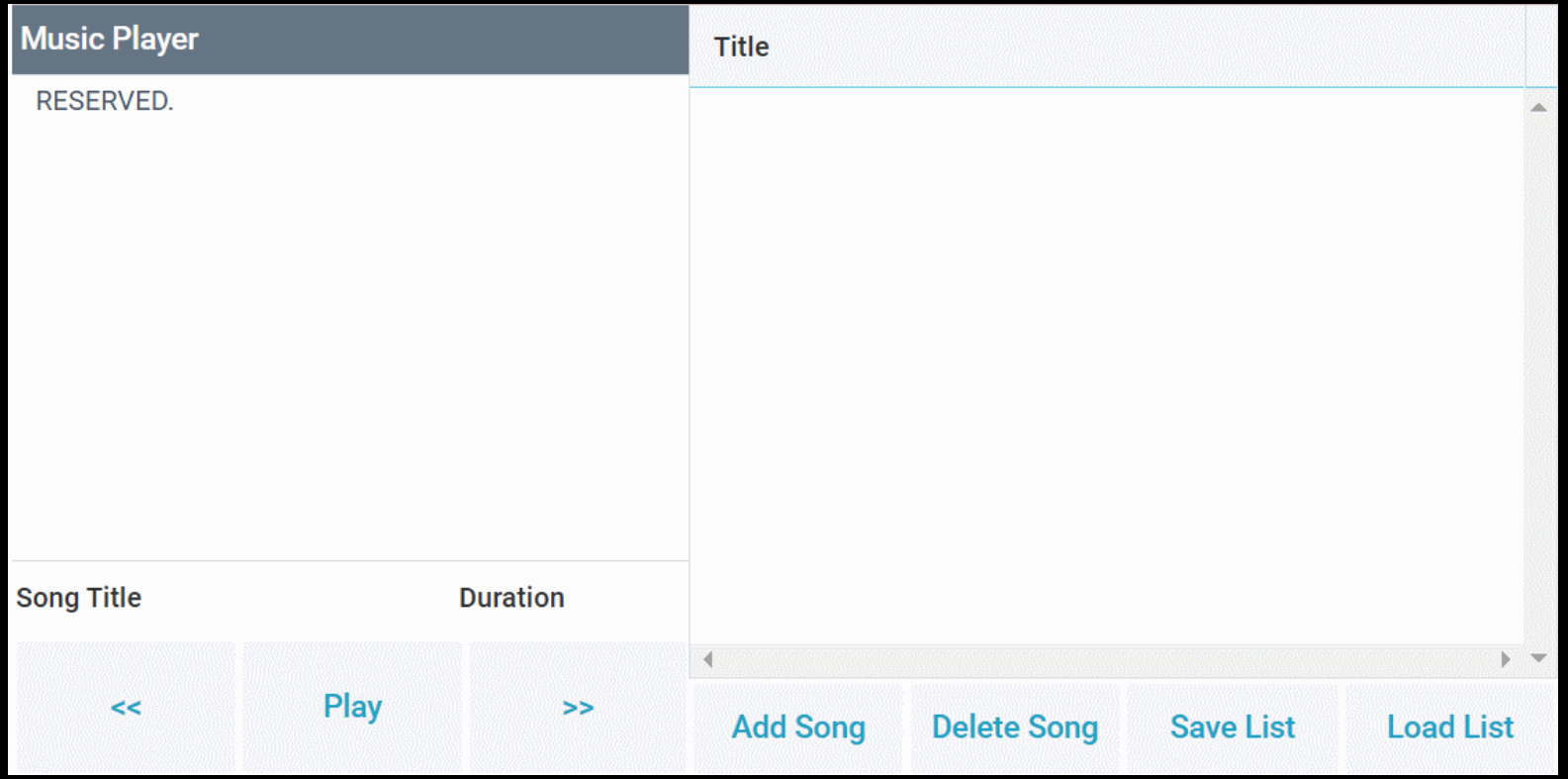
```
export class MusicPlayerController {
    view;
    database;
    constructor(view) {
        this.view = view;

        $$("btnAdd").attachEvent("onItemClick", () => {
            document.getElementById("fileInput").click();
        });
        document.getElementById("fileInput").addEventListener("change", (e) => {
            const jsonFiles = e.target.files;
            for(let i=0; i<jsonFiles.length;i++){
                console.log(jsonFiles[i])
                let audioUrl = URL.createObjectURL(jsonFiles[i]);
                let item = {
                    id: Date.now()+i,
                    title: jsonFiles[i].name,
                    url: audioUrl
                };
                this.AddItem(item);
            }
        });
        $$("btnDelete").attachEvent("onItemClick", () => {
            this.DeleteItem();
        });
        $$("btnPlay").attachEvent("onItemClick", () => {
            this.PlayAudio(false);
        });
        $$("datatable").attachEvent("onItemDblClick", (item) => {
            this.view.SelectItem(item.row);
            this.PlayAudio(true);
        });
        $$("btnNext").attachEvent("onItemClick", () => {
            this.Next();
        });
        $$("btnPrev").attachEvent("onItemClick", () => {
            this.Prev();
        });
        this.view.audio.addEventListener("timeupdate", () => {
            this.view.SetSongDuration(this.view.audio.currentTime, this.view.audio.duration);
        });
        this.view.audio.addEventListener("ended", () => {
            this.Next();
        });
    }

```

```
47      AddItem(item) {
48          this.view.AddItem(item);
49      }
50
51      DeleteItem() {
52          const id = this.view.GetSelectedId();
53          const nextId = this.view.GetNextId();
54          this.view.DeleteItem(this.view.GetItem(id));
55          this.view.SelectItem(nextId);
56      }
57
58      PlayAudio(forcePlay = false, _id = null) {
59          const id = (_id === null) ? this.view.GetSelectedId() : _id;
60          const item = this.view.GetItem(id);
61          if (this.view.GetPlayButtonLabel() === "Play" || forcePlay) {
62              this.view.LoadAudioURL(item.url);
63              this.view.PlayAudio();
64              this.view.SetSongTitle(item.title);
65              this.view.SetPlayButtonLabel("Pause");
66          }
67          else {
68              this.view.PauseAudio();
69              this.view.SetPlayButtonLabel("Play");
70          }
71      }
72
```

```
73      Next() {
74          const id = this.view.GetNextId();
75          const item = this.view.GetItem(id);
76          this.view.SelectItem(item.id);
77          this.PlayAudio(true, id);
78      }
79
80      Prev() {
81          const id = this.view.GetPrevId();
82          const item = this.view.GetItem(id);
83          this.view.SelectItem(item.id);
84          this.PlayAudio(true, id);
85      }
86  }
```

Test

- Test all the features.

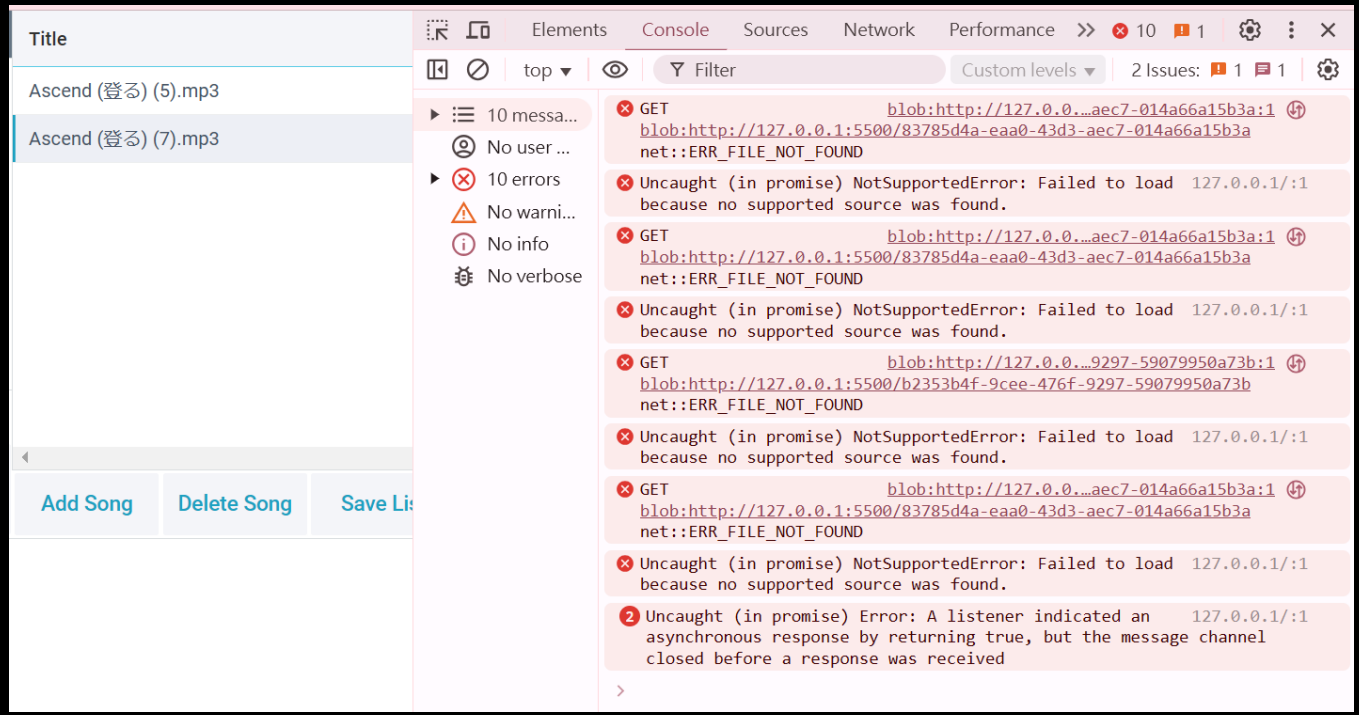


Electron App Exercise #04

- Music Player
- MVC Music Player
  - MusicPlayerView
  - MusicPlayerController
  - MusicPlayerModel
- Discussions

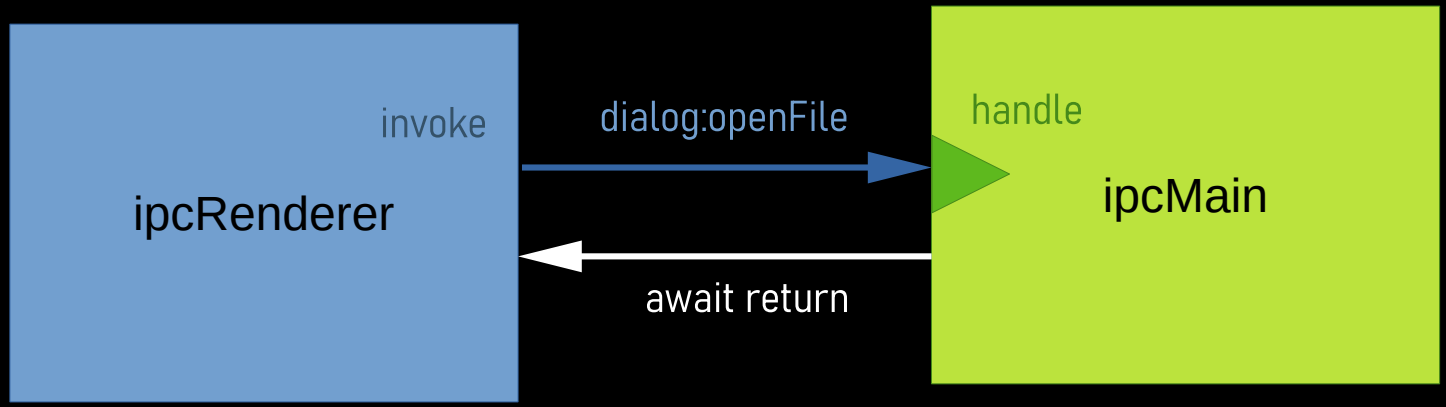
The Upload Cache of Webapp

- When you upload a file through a browser, the file is loaded into memory as a File object.
- If you use the URL.createObjectURL() method to generate a URL for this file, the resulting URL can be treated like a web link, allowing you to access and read the file.
- However, once you close/refresh the browser tab, both the File object and the generated URL are discarded and no longer accessible.
- The browser will never know the real address of any file for safety concern.



Use Electron File Dialog

- To access the actual file path, we must use the file dialog provided by Electron.
- The Electron file dialog can only be invoked from the main process (main.js). Therefore, the renderer process (renderer.js) must communicate with the main process via IPC (Inter-Process Communication).
- Both processes can initiate communication, and the response can be received either synchronously (via a normal return) or asynchronously (using await).
- If you do not need to use await and prefer a simple message without waiting for a response, you can use ipcRenderer.send() in the renderer process and ipcMain.on() in the main process.



Customized Callbacks

- A callback is a function passed as an argument to another function, and it is executed (called back) at a later time. Callbacks are commonly used in asynchronous operations, such as reading files, making API requests, or handling user interactions.
- We can use callbacks as delegates in C#. In our example, when the Database updates a large number of items (e.g., calling Load()), it triggers onDataUpdated. Instead of passing, we directly assign onDataUpdated in the Controller to manage the refreshing of the entire view.

```
64          this.database.onDataUpdated = () => {
65              this.RefreshView();
66          };
```

Assign callback in Controller

```
54      Load(){
55          let dataJSONArray = JSON.parse(localStorage.getItem("saveData"));
56          this.d = dataJSONArray;
57          if(this.onDataUpdated){
58              this.onDataUpdated(this.d);
59          }
60      }
```

Invoke callback in Database

MusicPlayerModel.js

-

```
export class MusicModel{
    id;
    title;
    url;
    duration;
    constructor(id, title, url){
        this.id = id;
        this.title = title;
        this.url = url;
    }
}

export class MusicPlayerDatabase{
    d;
    onDataUpdated;
    constructor(){
        this.d = [];
        this.onLoad = null;
    }

    AddItem(item){
        this.d.push(item);
    }

    GetItem(id){
        let result = null;
        this.d.forEach((item) => {
            if(item.id === id){
                result = item;
            }
        });
        return result;
    }

    GetAllItems(){
        return this.d;
    }

    DeleteItem(_item){
        let index = -1;
        this.d.forEach((item, ind) => {
            if(item.id === _item.id){
                index = ind;
            }
        });
        this.d.splice(index, 1);
    }

    Save(){
        let dataJSONArray = this.d
        localStorage.setItem("saveData", JSON.stringify(dataJSONArray));
    }

    Load(){
        let dataJSONArray = JSON.parse(localStorage.getItem("saveData"));
        this.d = dataJSONArray;
        if(this.onDataUpdated){
            this.onDataUpdated(this.d);
        }
    }

    PrintAll(){
        this.d.forEach((item)=>{
            console.log(item);
        });
    }
}
```

MusicPlayerController.js

```
1  const { ipcRenderer } = require('electron');
2  const { basename } = require('path');
3
4  export class MusicPlayerController {
5      view;
6      database;
7      constructor(view, database) {
8          this.view = view;
9          this.database = database;
10
11         $$("btnAdd").attachEvent("onItemClick", async () => {
12             //document.getElementById("fileInput").click();
13             const filePaths = await ipcRenderer.invoke('dialog:openFile');
14             for (let i = 0; i < filePaths.length; i++) {
15                 let audioUrl = filePaths[i];
16                 let item = {
17                     id: Date.now() + i,
18                     title: basename(filePaths[i]),
19                     url: audioUrl
20                 };
21                 this.AddItem(item);
22             }
23         });
24         // document.getElementById("fi
25         //     const jsonFiles = e.tar
26         //     for (let i = 0; i < jso
27         //         let audioUrl = URL.
28         //         let item = {
29         //             id: Date.now()
30         //             title: jsonFile
31         //             url: audioUrl
32         //         };
```

```
52         this.view.audio.addEventListener("timeupdate", () => {
53             this.view.SetSongDuration(this.view.audio.currentTime, this.view.audio.duration);
54         });
55         this.view.audio.addEventListener("ended", () => {
56             this.Next();
57         });
58         $$("btnSave").attachEvent("onItemClick", () => {
59             this.database.Save();
60         });
61         $$("btnLoad").attachEvent("onItemClick", () => {
62             this.database.Load();
63         });
```

- Import ipcRenderer. Please be noted that for npm modules, we have to use the common.js style import pattern (using required()).
- Import basename to slice filename from path.
- Use ipcRenderer.invoke() to raise an ipc event. The name of the event is customized as 'dialog:openFile'
- Attach btnSave and btnLoad to correct actions.

```
64          this.database.onDataUpdated = () => {
65              this.RefreshView();
66          };
67      }
68
69      AddItem(item) {
70          //this.view.AddItem(item);
71          this.database.AddItem(item);
72          this.RefreshView();
73      }
74
75      DeleteItem() {
76          const id = this.view.GetSelectedId();
77          const nextId = this.view.GetNextId();
78          // this.view.DeleteItem(this.view.GetItem(id));
79          // this.view.SelectItem(nextId);
80          this.database.DeleteItem(this.view.GetItem(id));
81          this.RefreshView(nextId);
82      }
83
84      RefreshView(id=null) {
85          $$("datatable").clearAll();
86          this.database.GetAllItems().forEach((item) => {
87              this.view.AddItem(item);
88          });
89          if(id){
90              this.view.SelectItem(id);
91          }
92      }
93
94      PlayAudio(forcePlay = false, _id = null) {
95          const id = (_id === null) ? this.view.GetSelectedId() : _id;
96          const item = this.view.GetItem(id);
97          if (this.view.GetPlayButtonLabel() === "Play" || forcePlay) {
```

- AddItem()
  - Adds an item to the database and refreshes the view.
- DeleteItem()
  - Deletes an item from the database and refreshes the view.
- Database Load Completion
  - When the Load() operation on the database finishes:
    - The onDataUpdated callback is invoked, and we refresh the view.
- View Refreshing
  - Whenever the Controller operates on the Database or receives a notification about the database update, it triggers the view to refresh. (It's that simple!)

main.js

- Import ipcMain, dialog from electron module.
- Handle the 'dialog:openFile' event in ipcMain.handle().
  - Returns the user selected file paths as an array.

```
const { app, BrowserWindow, ipcMain, dialog } = require('electron');
const path = require('path');

function createWindow() {
    const win = new BrowserWindow({
        width: 810,
        height: 430,
        webPreferences: {
            nodeIntegration: true,
            contextIsolation: false,
            devTools: true,
        },
        autoHideMenuBar: true,
    });
    win.loadFile(path.join(__dirname, '..', 'index.html'));
}

app.whenReady().then(() => {
    createWindow();

    app.on('activate', () => {
        if (BrowserWindow.getAllWindows().length === 0) createWindow();
    });
});

app.on('window-all-closed', () => {
    if (process.platform !== 'darwin') app.quit();
});

ipcMain.handle('dialog:openFile', async () => {
    const result = await dialog.showOpenDialog({
        properties: ['openFile', 'multiSelections'],
        filters: [
            { name: 'Audio Files', extensions: ['mp3'] }
        ]
    });
    return result.filePaths;  // Return the full file path
});
```

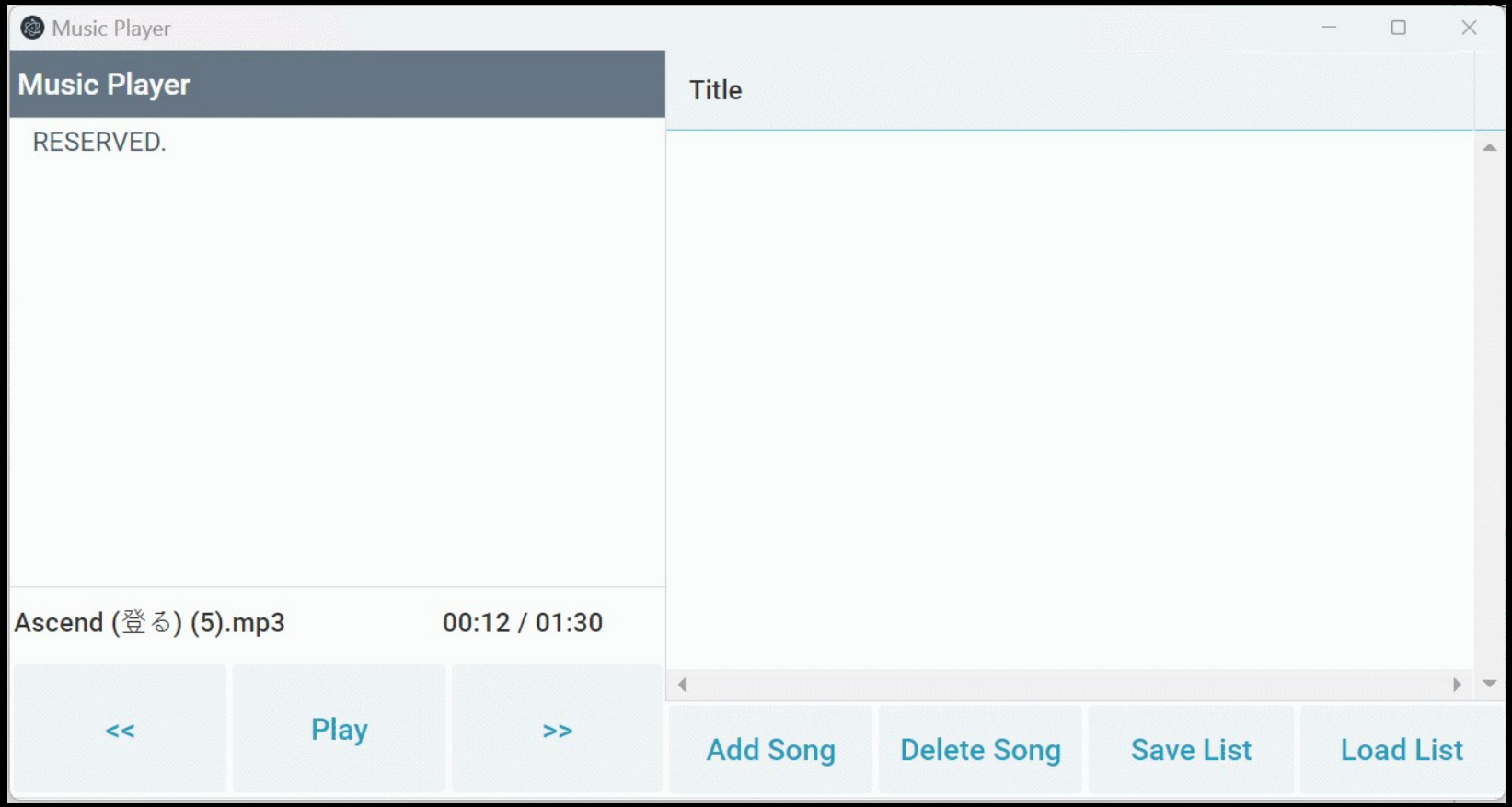
renderer.js

- Import MusicPlayerDatabase
- Create and pass an instance of Database to Controller.
- Set a debug function:
  - Press 1 to show all database in console.

```
1  import { MusicPlayerController } from "./MusicPlayerController.js";
2  import { MusicPlayerDatabase } from "./MusicPlayerModel.js";
3  import { MusicPlayerView } from "./MusicPlayerView.js";
4
5  let view;
6  let controller;
7  let database;
8
9  window.addEventListener("load", ()=>{
10         view = new MusicPlayerView();
11         database = new MusicPlayerDatabase();
12         controller = new MusicPlayerController(view, database);
13 });
14
15 window.addEventListener("keydown", (e)=>{
16     if(e.key === "1"){
17         database.PrintAll();
18     }
19 });
```

Test

- Test all the features in Electron.



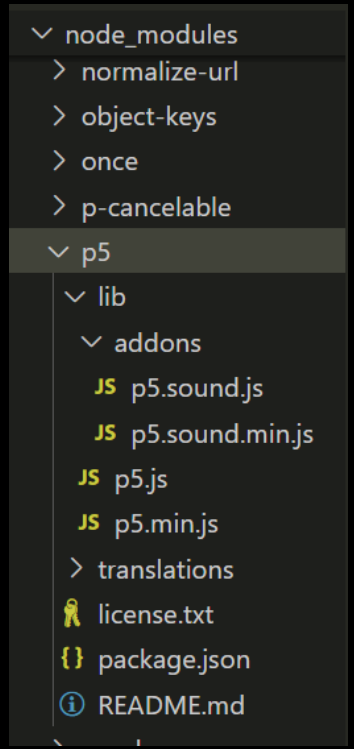
Electron App Exercise #04

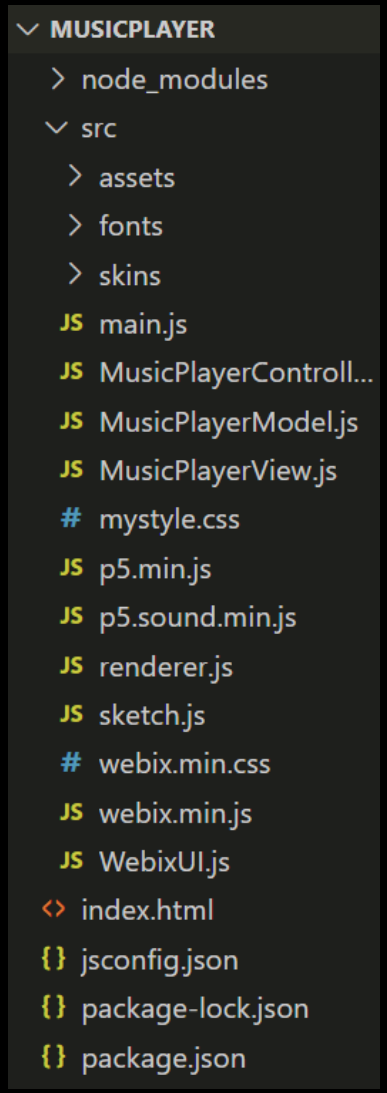
- Music Player
- MVC Music Player
  - MusicPlayerView
  - MusicPlayerController
  - MusicPlayerModel
- Discussions

Integrate with P5.js

- npm install p5
- Copy the p5.min.js and p5.sound.min.js to /src folder.
- Add a sketch.js in /src folder.
- Edit index.html:
  - Import p5.min.js, p5.sound.min.js, sketch.js

```
1   <html>
2       <head>
3           <title>Music Player</title>
4           <link rel="stylesheet" href="src/webix.min.css">
5           <link rel="stylesheet" href="src/mystyle.css">
6           <script src="src/p5.min.js"></script>
7           <script src="src/p5.sound.min.js"></script>
8       </head>
9       <body>
10          <script src="src/webix.min.js"></script>
11          <script src="src/renderer.js" type="module"></script>
12          <script src="src/sketch.js"></script>
13      </body>
14  </html>
```





- Remove the padding around webix template area.

```css
body {
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

.webix_template{
    padding: 0;
}
```

sketch.js boilerplate

- This create a div as container for the p5 canvas.
- Create p5 canvas then attach to container
- You may create your artwork starting from here.

```
function setup() {
    $$("reserved").setHTML(`<div id='p5Container'
        style="width: 100%; height: 100%; padding: 0;"></div>`);
    // Get the width and height of the div
    let canvasDiv = select('#p5Container');
    let canvasWidth = canvasDiv.width;
    let canvasHeight = canvasDiv.height;

    // Create the canvas and set its size to match the div
    let canvas = createCanvas(canvasWidth, canvasHeight);
    canvas.parent('p5Container');  // Attach the canvas to the div with id 'myCanvas'
}

function draw() {
    background(240);
}
```

FFT Example

- Go to View:
  - Add id for the audio element.
  - Append the audio element in html body.
- Setup p5 for an fft.

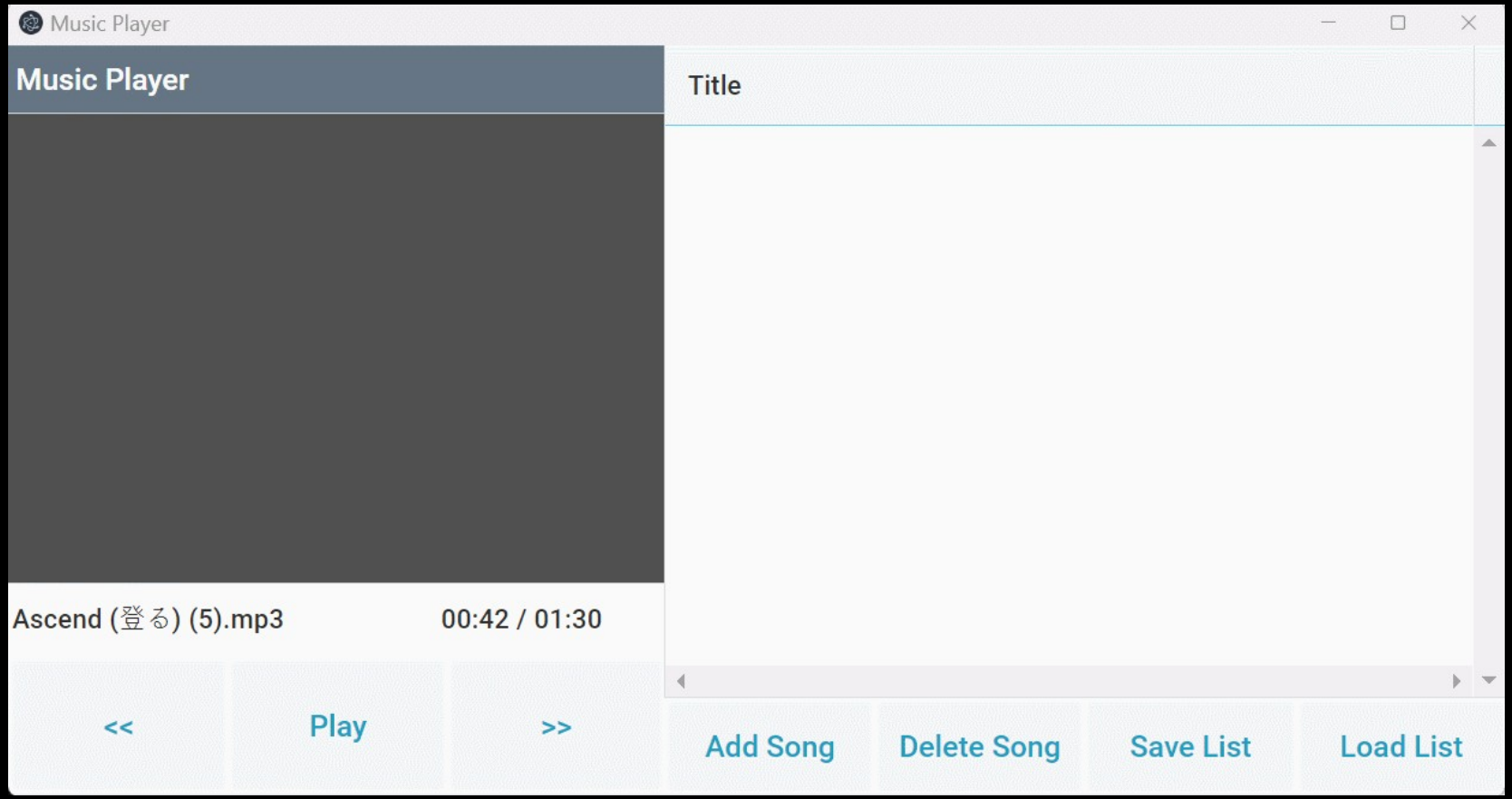
```
22      // Create an audio element.
23      this.audio = new Audio();
24      this.audio.id = "webixAudio";
25      document.body.appendChild(this.audio);
```

```
5   function setup() {
6     $$("reserved").setHTML(`
7       <div id='p5Container' style="width: 100%; height: 100%; padding: 0;"></div>
8     `);
9
10    // Get the container's dimensions
11    let canvasDiv = select('#p5Container');
12    let canvasWidth = canvasDiv.elt.offsetWidth;
13    let canvasHeight = canvasDiv.elt.offsetHeight;
14
15    // Create the canvas inside the container div
16    let canvas = createCanvas(canvasWidth, canvasHeight);
17    canvas.parent('p5Container');
18
19    // Get the HTML audio element and initialize FFT
20    audioElement = select('#webixAudio').elt;
21    console.log(audioElement);
22
23    fft = new p5.FFT();
24    let mediaElem = new p5.MediaElement(audioElement);
25    fft.setInput(mediaElem);
26  }
```

```
27
28  class Particle {
29    constructor(x, y, clr) {
30      this.pos = createVector(x, y);
31      // Upward velocity for a subtle rise
32      this.vel = new p5.Vector(0, 0.0);
33      this.lifespan = 200; // Determines fade-out time
34      this.clr = clr;
35    }
36
37    update() {
38      this.pos.add(this.vel);
39      this.vel.add(new p5.Vector(0, 0.2));
40      this.lifespan -= 5; // Fade-out speed
41    }
42
43    display() {
44      noStroke();
45      fill(200, 200, 0, this.lifespan);
46      // Very fine particle: 1px by 5px rectangle
47      rect(this.pos.x, this.pos.y+5, 1, 5);
48    }
49
50    isDead() {
51      return this.lifespan < 0;
52    }
53  }
54
```

```
55  function draw() {
56    background(80);
57
58    // 1. Update and display particles first
59    for (let i = particles.length - 1; i >= 0; i--) {
60      particles[i].update();
61      particles[i].display();
62      if (particles[i].isDead()) {
63        particles.splice(i, 1);
64      }
65    }
66
67    // 2. Then perform FFT analysis and draw the FFT bars
68    let spectrum = fft.analyze();
69    let barWidth = (width + 100) / spectrum.length;
70
71    for (let i = 0; i < spectrum.length; i++) {
72      let amp = spectrum[i];
73      let barHeight = map(amp, 0, 255, 0, height);
74      let x = i * barWidth;
75      let y = height - barHeight;
76
77      // Use a consistent color for both the FFT bar and the particle
78      let barColor = color(230, random(100, 150), 0);
79      fill(barColor);
80      noStroke();
81      rect(x, y, barWidth+0.2, barHeight);
82
83      // Emit a particle from the tip of the bar (with 10% chance)
84      if (amp > 10 && random(1) < 0.1) {
85        particles.push(new Particle(x + barWidth / 2, y - 20, barColor));
86      }
87    }
88  }
```

- Test all the features in Electron.



Q&A