Project Introduction

ZENVA

Project Introduction

- Setting up our project
- Kontra.js basics
- Auto scaling our game
- Building the Match 3 Game Class
- Building the Match 3 Game with Kontra.js

Project Setup

Project Setup

- Setting up our project in VS Code
- Installing the Live Server Plugin in VS Code
- Using the Beautify Plugin in VS Code
- Using the Live Server Plugin to start a server

Summary

Local Web Server

- In order to run our Match 3 game locally, we require a local web server. The reason for this is that when we go to load in local files, it is a security risk to load any files using file://.

- By using a local web server, these files are now served via http, which allows us to load these files in our game.

ZENVA

JavaScript Modules

ZENVA

JavaScript Modules

- Add Kontra Library
- How JavaScript Modules work
- Setup our code to use JavaScript Modules

ZENVA

Summary

JavaScript Modules

- In order to use JavaScript Modules, you need to tell the browser that your JavaScript file is a module. To do this, you need to add the type attribute to your script tag.

```
4    <script type="module" src="src/main.js"></script>
```

ZENVA

Initialize Kontra

Initialize Kontra

- Accessing the global Kontra instance
- Initialize Kontra

Summary

Initialize Kontra

- In order to initialize Kontra, you need to call init(). When you call init(), this will create the drawing context.
- By default, Kontra will use the first HTML Canvas element on the page.

Kontra Game Loop

ZENVA

Kontra Game Loop

- Create a Kontra Game Loop
- Adding the update function
- Adding the render function

ZENVA

Summary

Kontra Game Loop

- To create a Game Loop in Kontra, you need to use the GameLoop function:

```
let loop = GameLoop({  // create the main game loop
  update: function() { // update the game state
  },
  render: function() { // render the game state
  }
});
```

ZENVA

Draw Sprites

ZENVA

Draw Sprites

- Create a Kontra Sprite
- Differences between Rectangle Sprite and Image Sprite

ZENVA

Summary

Create Sprite

- To create a Sprite in Kontra, you need to use the Sprite function:

```
let { Sprite } = kontra

let sprite = Sprite({
  x: 300,
  y: 100,
  anchor: {x: 0.5, y: 0.5},

  // required for a rectangle sprite
  width: 20,
  height: 40,
  color: 'red'
});

sprite.render();
```

ZENVA

Draw Image Sprite

ZENVA

Draw Image Sprites

- Create a Kontra Image Sprite
- Create an HTML Image element
- Load an Image

ZENVA

Summary

Create Image Sprite

- To create an Image Sprite in Kontra, you need to use the Sprite function:

```
let image = new Image();
image.src = 'assets/imgs/character.png';
image.onload = function() {
  let sprite = Sprite({
    x: 300,
    y: 100,
    anchor: {x: 0.5, y: 0.5},

    // required for an image sprite
    image: image
  });

  sprite.render();
};
```

ZENVA

Interact With Sprites

ZENVA

Interact WIth Sprites

- Using the Kontra Pointer API
- Listening for Kontra Pointer events
- Tracking Sprites with Kontra

ZENVA

Summary

Sprite Interaction

- To interact with Sprites in Kontra, you need to use the pointer API:

```
let { initPointer, track, Sprite } = kontra;

// this function must be called first before pointer
// functions will work
initPointer();

let sprite = Sprite({
  onDown: function() {
    // handle on down events on the sprite
  },
  onUp: function() {
    // handle on up events on the sprite
  },
  onOver: function() {
    // handle on over events on the sprite
  }
});

track(sprite);
sprite.render();
```

ZENVA

Game Class

Game Class

- Create Game Class
- Refactor main.js

Summary

bind()

- The bind() method creates a new function that when it is called, it will have the this keyword set to the context that is provided by the user.

```
1   let user = {
2     firstName: "John"
3   };
4
5   function getName() {
6     console.log(this.firstName);
7   }
8
9   funcUser(); // undefined
10
11  let funcUser = func.bind(user);
12  funcUser(); // John
```

ZENVA

Game Background

Game Background

- Add CSS to our game
- Add Background Image to our game

Summary

CSS background-repeat

- By default, a background image will be repeated both vertically and horizontally. The background-repeat property allows you to disable this.

```
1   <style>
2   body {
3     background-image: url("paper.gif");
4     background-repeat: no-repeat;
5   }
6   </style>
```

ZENVA

Scaling

Scaling

- Resize Canvas Element
- Adding a DOM EventListener for the resize event

Summary

onresize Event

- The onresize event is fired any time the browser window has been resized.
- When this event fires, you can use the innerWidth and innerHeight properties of the window to resize your game.

```
1    window.addEventListener("resize", myFunction);
2
```

ZENVA

Grid Class

Grid Class

- Create Grid Class
- Initialize Grid Class instance

Summary

Class Declarations

- One important thing to remember about using a class declaration is that these declarations are not hoisted. This means that your class needs to be first declared before you can initialize it.

```
1    const p = new Rectangle(); // ReferenceError
2
3    class Rectangle {}
```

Drawing the Grid

Drawing the Grid

- Update the Grid Class
- Draw the grid background
- Draw the grid lines

ZENVA

Summary

Draw Lines

- In order to draw a line in Kontra, you can use the basic Rectangle Sprite.
- To draw a line, you will just need to set the width or height property to 1.

Canvas Global Alpha

ZENVA

Canvas Global Alpha

- CanvasRenderingContext2D.globalAlpha
- Update global alpha in Grid Class
- Make grid transparent

Summary

CanvasRenderingContext2D.globalAlpha

- The globalAlpha property of the of the Canvas 2D API allows you to specify the alpha, or transparency, value that will be applied to all shapes and images before they are drawn to the Canvas.

- In order to only set the alpha for one image, you will need to update the global alpha to the value you want, render that image, and then set the global alpha back to 1 before rendering the other game objects.

Object Pools

ZENVA

Object Pools

- Create Object Pool
- Update grid class to use object pool

Summary

Kontra Object Pool

- To create an Object Pool in Kontra, you need to use the Pool function. This function expects an object with a create method which should return a new Sprite or object.

```
let { Pool, Sprite } = kontra;

let pool = Pool({
  // create a new sprite every time the pool needs a new object
  create: Sprite
});

// properties will be passed to the sprites init() function
pool.get({
  x: 100,
  y: 200,
  width: 20,
  height: 40,
  color: 'red',
  ttl: 60
});
```

ZENVA

Board Class

Board Class

- Create Board Class
- Create and populate grid in Board Class
- Create and populate reserve grid in Board Class

Summary

2D Arrays

- In JavaScript, a 2d array is a collection of items that are organized as a matrix in the form of rows and columns.
- These arrays are standard one dimensional array objects that stores arrays for the values in the array.
- You can access elements from the inner arrays like so: arr[row][col]

Print Grid

ZENVA

Print Grid

- Update consoleLog method in Board Class
- Pretty print Board Class grid to console
- Pretty print Board Class reserve grid to console

Summary

Console Log

- When you are logging information to the console, if you want to print a new line, you can use the “\n” character.

```
1    const printMe = "Hi there, my name is: \n Joe!";
2    console.log(printMe);
```

ZENVA

Swap Blocks

Swap Blocks

- Update Board Class
- Add swap method to Board Class

Summary

Swap Blocks

- In order to swap two blocks, we need to create a temporary variable to hold the required information for one of the blocks.

- By doing this, we can update one of the blocks values to be equal to another blocks, and then update the second block to be equal to the temporary variable.

Valid Moves

Valid Moves

- Update Board Class
- Add checkAdjacent method to Board Class

Summary

Valid Moves

- In order to check to see if a move is valid or not, we need to make sure that the two blocks are within 1 row or column of each other.

- By using the Math.abs() method, we can make sure that the distance between the two blocks is always positive.

Chains

Chains

- Update Board Class
- Add isChained method to Board Class

Summary

Checking for chains

- When you are trying to access a nested property in JavaScript, you will need to make sure that the properties up to the nested value exist, otherwise you will get an error.

```
1   const obj = {};
2
3   console.log(obj[x][i]); // with throw an error
4
```

ZENVA

Find All Chains

Find All Chains

- Update Board Class
- Add findAllChains method to Board Class

Summary

Enhancing the game

- Currently, when we are checking for chains, we are only checking to see if a block has the same value as two blocks next to it. In a lot of match 3 type games, there are special blocks that will allow you to do different things: remove all blocks that are one color, explode and remove blocks within a radius, etc.

- If we wanted to enhance our game to support these, we would need to account for those in our find chains methods.

Clear Chains

Clear Chains

- Update Board Class
- Add clearChains method to Board Class

Summary

forEach() Method

- This method is used to execute a function once for every element in an array.
- One important thing to remember about the forEach() method is that there is no way to break a loop unless you throw an exception. If you will need to terminate the loop early, you will want to use a different method.

Drop Blocks

Drop Blocks

- Update Board Class
- Add dropBlock method to Board Class
- Add dropReserveBlock method to Board Class

Summary

Game Enhancements

- Currently, in our Block class we have a lot of duplicated code that is used for the grid and reserve grid. One enhancement that we could make is that we could refactor this code into a new method that would allow us to pass which grid we are working with as an argument.

- This could also be done with the dropBlock and dropReserveBlock methods.

Update Grid

Update Grid

- Update Board Class
- Add updateGrid method to Board Class

Summary

Loop through Array Backwards

- One of the ways you can loop through the elements in an array in reverse order is that you can use a for loop.
- However, instead of incrementing a value until it is equal to the length of the array, you will want to decrement a value until it is less than 0.

```
1   for (let i = arr.length; i >= 0; i--) {
2     // logic
3   }
```

ZENVA

Load Assets

ZENVA

Load Assets

- Update Game Class
- Loading assets in Kontra with the load() method

Summary

Loading Assets in Kontra

- To load assets in Kontra, you can use the Asset Loader that is available. This loader is a promise based asset loader that will return a promise that resolves when all assets have been loaded.

```
let { load, on } = kontra;

let numAssets = 3;
let assetsLoaded = 0;

load(
  'assets/imgs/character.png',
  'assets/data/tile_engine_basic.json',
).then(function(assets) {
  // all assets have loaded
}).catch(function(err) {
  // error loading an asset
});
```

Draw Blocks

ZENVA

Draw Blocks

- Update Game Class
- Add drawBoard method to Game Class
- Create block pool

Summary

Kontra Sprite Class

- In order to extend the Kontra Sprite Class, you need to use the .class property since Sprite is a factory.

```
1   let { Sprite } = kontra;
2
3   class CustomSprite extends Sprite.class {
4     // ...
5   }
6
```

Pick Blocks

ZENVA

Pick Blocks

- Update Game Class
- Update drawBoard method to make Sprites interactive
- Add pickBlock method
- Add swapBlocks method
- Add clearSelection method

Summary

Kontra Pool getAliveObjects() Method

- The getAliveObjects() method can be used to return an array of Sprite objects that have the alive property set to true from your Kontra Pool.

```
let { Pool, Sprite } = kontra;

let pool = Pool({
  // create a new sprite every time the pool needs a new object
  create: Sprite
});

const arr = pool.getAliveObjects();

```

Update Board

ZENVA

Update Board

- Update Game Class
- Update swapBlocks method to swap the two selected Sprites
- Add updateBoard method to Game Class

Summary

Game Enhancements

- Currently, when we swap two blocks in our game, we are just updating the positions of the two blocks and their colors. Once enhancement we could make would be to animate the blocks so the user can see them swap positions.

- This could be done using the dx and dy properties on the Sprite objects.

Drop Blocks

Drop Blocks

- Update dropBlock method in Board Class
- Update dropReserve method in Board Class
- Add dropBlock method to Game Class
- Add dropReserve method to Game Class

Summary

Kontra Sprite.ttl

- The ttl property on a Sprite object is used to know how many frames the sprite should be alive for.

- This property is primarily used by the Pool to when an object should be recycled.

Drop Blocks Part 2

ZENVA

Drop Blocks Part 2

- Update dropReserve method in Game Class
- Refactor populateGrid method in Board Class
- Refactor populateReserveGrid method in Board Class

Summary

Code Refactoring

- Refactoring your code means you are updating your source code without changing the behavior of your code.
- Refactoring helps keep your code clean, solid, dry, and easy to maintain.
- Refactoring is an important part of the coding process.

Project Conclusion