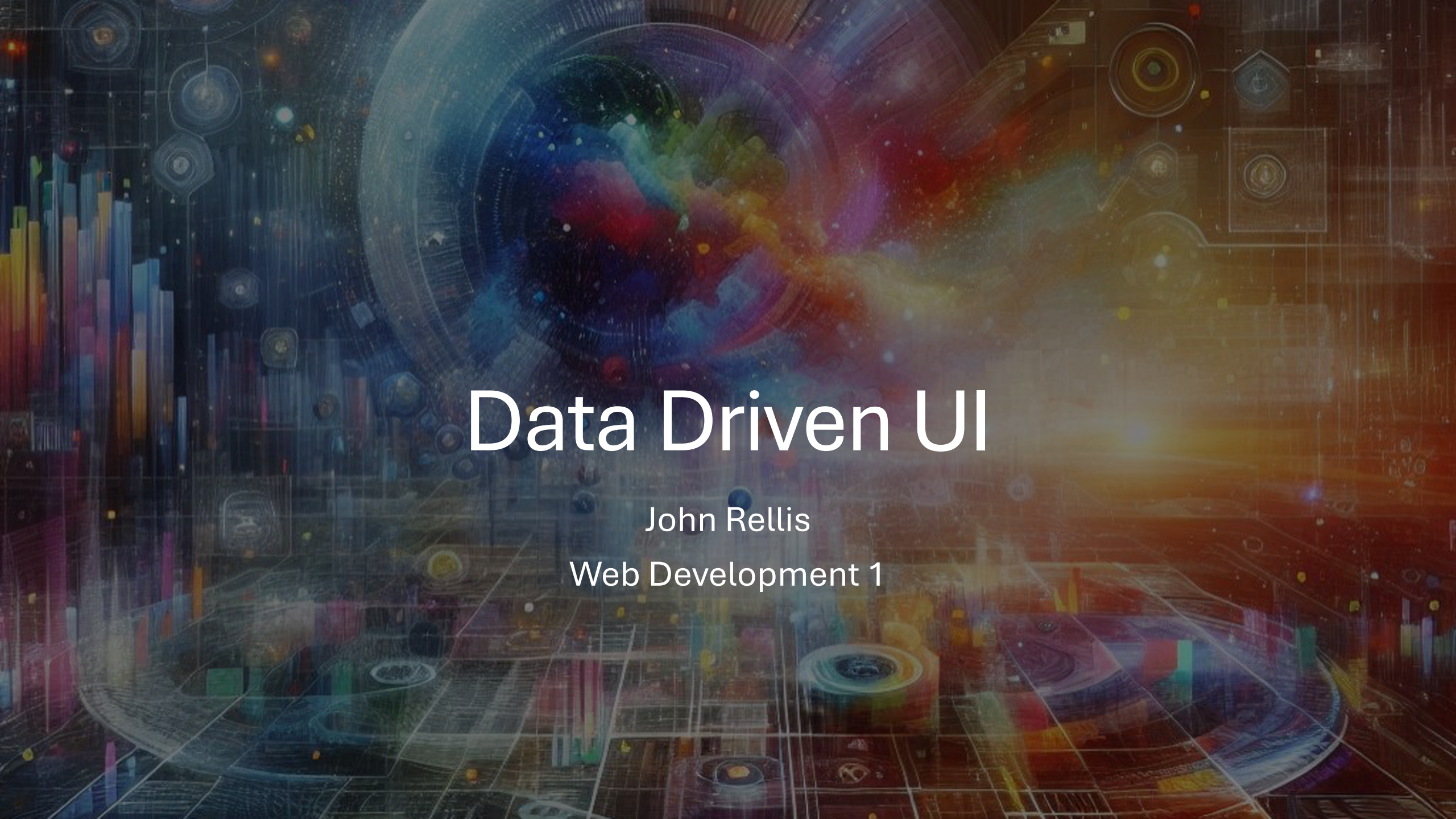


Data Driven UI

John Rellis

Web Development 1

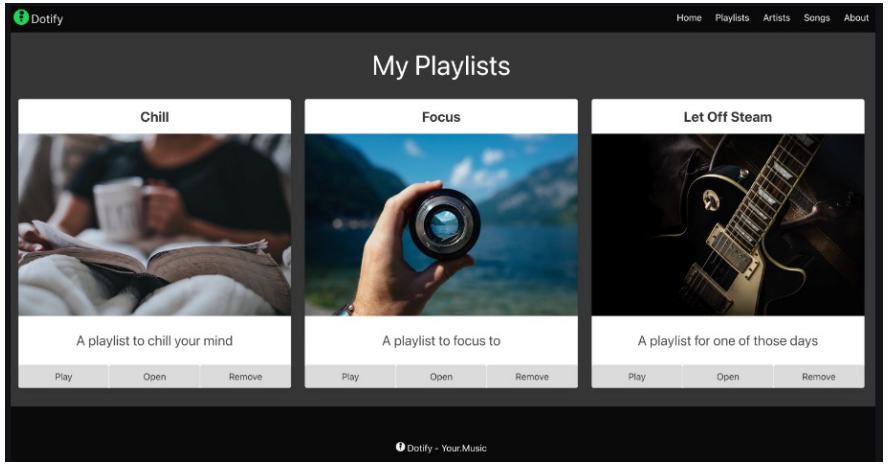
Today

- What is a data model?
- Building a UI from data
- Introducing components
- What is model view controller?
- Introducing a datastore
- Creating a namespace

Our present situation

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const playlistOneHeading = document.querySelector('#playlist-1-heading');
    playlistOneHeading.innerHTML = 'Chill';

    const playlistOneImage = document.querySelector('#playlist-1-image');
    playlistOneImage.src = 'https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ';

    const playlistOneDescription = document.querySelector('#playlist-1-description');
    playlistOneDescription.innerHTML = 'A playlist to chill your mind';

    const playlistTwoHeading = document.querySelector('#playlist-2-heading');
    playlistTwoHeading.innerHTML = 'Focus';

    const playlistTwoImage = document.querySelector('#playlist-2-image');
    playlistTwoImage.src = 'https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA';

    const playlistTwoDescription = document.querySelector('#playlist-2-description');
    playlistTwoDescription.innerHTML = 'A playlist to focus to';

    const playlistThreeHeading = document.querySelector('#playlist-3-heading');
    playlistThreeHeading.innerHTML = 'Let Off Steam';

    const playlistThreeImage = document.querySelector('#playlist-3-image');
    playlistThreeImage.src = 'https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo';

    const playlistThreeDescription = document.querySelector('#playlist-3-description');
    playlistThreeDescription.innerHTML = 'A playlist for one of those days';
  });
</script>
```

- When the page is loaded, hardcoded data is used to populate our user interface
- This is useful for learning JavaScript but in the real world it is pointless, we may as well just hardcode the values in the HTML as before



Introduce a data model

- Create a data structure that can represent (or model) our domain
- Our domain here is music streaming, specifically, playlists.
- To model a playlist we can use an object with:
  - Name
  - Description
  - Image Url
- We will add all of these to an array

```
const playlists = [
    {
        name: "Chill",
        description: "A playlist to chill to",
        imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
    },
    {
        name:"Focus",
        description: "A playlist to focus to",
        imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
    },
    {
        name: "Let Off Steam",
        description: "A playlist for one of those days",
        imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
    },
];
```

Introduce a data model

- We can now use this data model to populate our user interface (or view)
- We iterate over the data model to add data to our view
- This is the first step in decoupling the user interface from the data itself

```
document.addEventListener('DOMContentLoaded', () => {
  const playlists = [
    {
      name: "Chill",
      description: "A playlist to chill to",
      imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
    },
    {
      name: "Focus",
      description: "A playlist to focus to",
      imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
    },
    {
      name: "Let Off Steam",
      description: "A playlist for one of those days",
      imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
    },
  ];


  for (let index = 0; index < playlists.length; index++) {
    const playlist = playlists[index] // playlists[0]

    const playlistHeading = document.querySelector(`#playlist-${index + 1}-heading`);
    playlistHeading.innerHTML = playlist.name;

    const playlistImage = document.querySelector(`#playlist-${index + 1}-image`);
    playlistImage.src = playlist.imageUrl;

    const playlistDescription = document.querySelector(`#playlist-${index + 1}-description`);
    playlistDescription.innerHTML = playlist.description;
  };
});
```

Use data to create the UI

- Let's remove our playlists from our HTML so that we now only have a *main* element with our heading "My Playlists"
- Iterate over our data model and add a card for each entity (or playlist)
- We create a string using a template literal and add to the html of the main element

```
<main class="columns is-multiline box has-background-grey-darker">
  <div class="column is-12 has-text-centered has-text-white">
    <p class="is-size-1">My Playlists</p>
  </div>
</main>
```

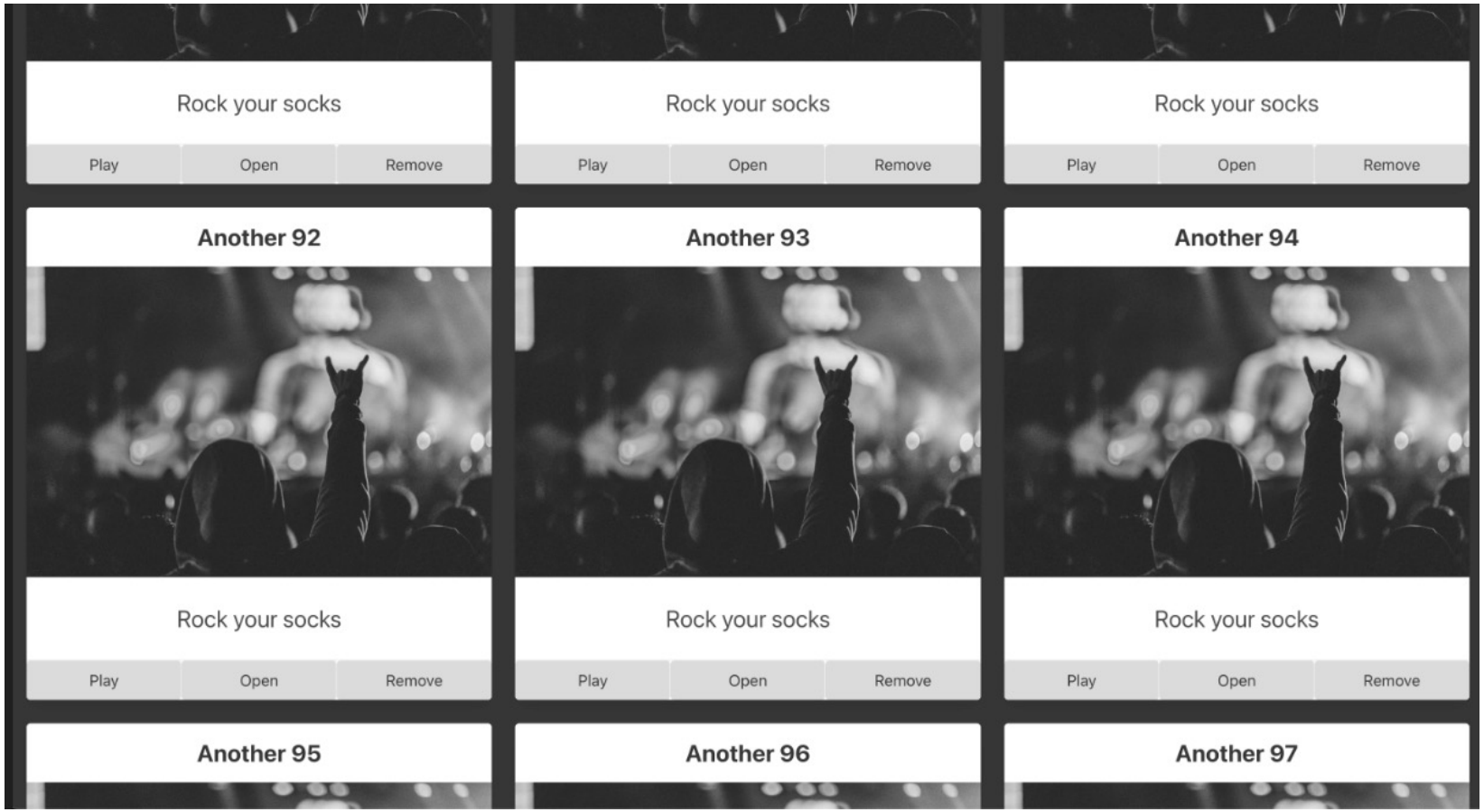
```
const playlists = [
  {
    name: "Chill",
    description: "A playlist to chill to",
    imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
  },
  {
    name: "Focus",
    description: "A playlist to focus to",
    imageUrl:
  },
  {
    name: "Le
    descripti
    imageUrl:
  }
];
```

```
const main = document.querySelector('main');
playlists.forEach((playlist, index) => {
  const element = `
  <div class="column is-4">
    <section class="card has-text-centered">
      <header class="card-header">
        <p class="card-header-title is-size-4 is-centered">
          ${playlist.name}
        </p>
      </header>
      <div class="card-image">
        <figure class="image">
          <img src="${playlist.imageUrl}" alt="Image">
        </figure>
      </div>
      <article class="card-content">
        <p class="content is-size-4">
          ${playlist.description}
        </p>
      </article>
      <footer class="card-footer">
        <button class="card-footer-item button has-background-grey-lighter">Play</button>
        <button class="card-footer-item button has-background-grey-lighter">Open</button>
        <button class="card-footer-item button has-background-grey-lighter">Remove</button>
      </footer>
    </section>
  </div>`
  main.innerHTML = main.innerHTML + element;
});
```

Use data to create the UI

- Now we have used data to drive the creation of playlist cards on our page

```
for (let i = 0; i < 100; i++) {
  playlists.push({
    name: `Another ${i}`,
    description: "Rock your socks",
    imageUrl: "https://source.unsplash.com/grayscale-photo-of-person-in-hoodie-top-watching-a-concert-97p-JwqdyW4"
  })
}
```



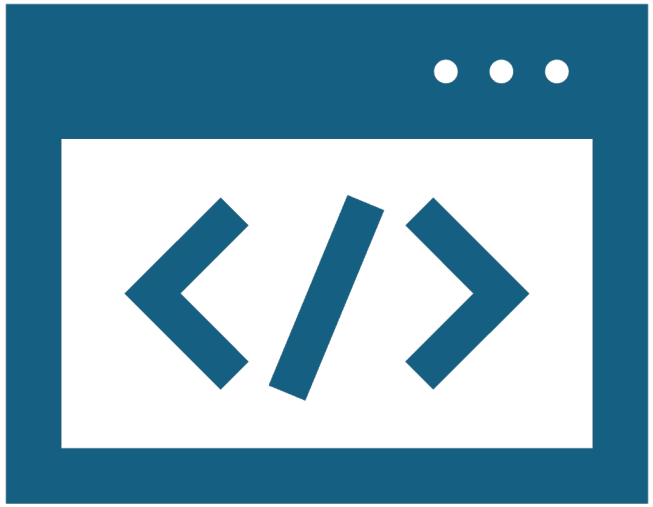

```
<!DOCTYPE html>
<html class="has-background-black" lang="en">
▶ <head>…</head>
▼ <body>
    <!-- START NAV -->
  ▶ <nav class="navbar has-background-black">…</nav>
    <!-- END NAV -->
  ▼ <main class="columns is-multiline box has-background-grey-darker is-centered">
    ▶ <div class="column is-12 has-text-centered has-text-white">…</div> == $0
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
    ▶ <div class="column is-4">…</div>
```

- The JavaScript on our page is still quite complicated and difficult to maintain
- Let's refactor the HTML creation into a component

```
<script>
    document.addEventListener('DOMContentLoaded', () => {
      const playlists = [
        {
          name: "Chill",
          description: "A playlist to chill to",
          imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
        },
        {
          name: "Focus",
          description: "A playlist to focus to",
          imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
        },
        {
          name: "Let Off Steam",
          description: "A playlist for one of those days",
          imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
        }
      ];

      const main = document.querySelector('main');
      playlists.forEach((playlist, index) => {
        const element = `
        <div class="column is-4">
          <section class="card has-text-centered">
            <header class="card-header">
              <p class="card-header-title is-size-4 is-centered">
                ${playlist.name}
              </p>
            </header>
            <div class="card-image">
              <figure class="image">
                <img src="${playlist.imageUrl}" alt="Image">      You, 19 hours ago • built functions
              </figure>
            </div>
            <article class="card-content">
              <p class="content is-size-4">
                ${playlist.description}
              </p>
            </article>
            <footer class="card-footer">
              <button class="card-footer-item button has-background-grey-lighter">Play</button>
              <button class="card-footer-item button has-background-grey-lighter">Open</button>
```

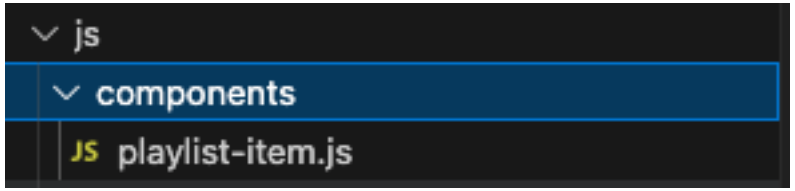
What is a component?

- A UI (User Interface) component is a reusable element of a graphical user interface (GUI) that serves a specific purpose in presenting information or enabling user interaction within a software application.
- UI components are building blocks used to create the visual layout and functionality of user interfaces.
- Think buttons, checkboxes, bulma cards
- We will use the term component to refer to reusable pieces of user interface and to extract their creation to a separate file



playlist-item.js component

- Create a new file
  - js/components/playlist-item.js
  - Add a function called createPlaylistItem
  - Returns a string that has playlist item information populated




```
const createPlaylistItem = (playlist) => {
    return `
    <div class="column is-4">
      <section class="card has-text-centered">
        <header class="card-header">
          <p class="card-header-title is-size-4 is-centered">
            ${playlist.name}
          </p>
        </header>
        <div class="card-image">
          <figure class="image">
            <img src="${playlist.imageUrl}" alt="Image">
          </figure>
        </div>
        <article class="card-content">
          <p class="content is-size-4">
            ${playlist.description}
          </p>
        </article>
        <footer class="card-footer">
          <button class="card-footer-item button has-background-grey-lighter">Play</button>
          <button class="card-footer-item button has-background-grey-lighter">Open</button>
          <button class="card-footer-item button has-background-grey-lighter">Remove</button>
        </footer>
      </section>
    </div>`
}
```

```
<script src="js/components/playlist-item.js"></script>
```

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const playlists = [
      {
        name: "Chill",
        description: "A playlist to chill to",
        imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
      },
      {
        name: "Focus",
        description: "A playlist to focus to",
        imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"      You, 20
      },
      {
        name: "Let Off Steam",
        description: "A playlist for one of those days",
        imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
      }
    ];

    const main = document.querySelector('main');
    playlists.forEach((playlist, index) => {
      const element = `
      <div class="column is-4">
        <section class="card has-text-centered">
          <header class="card-header">
            <p class="card-header-title is-size-4 is-centered">
              ${playlist.name}
            </p>
          </header>
          <div class="card-image">
            <figure class="image">
              <img src="${playlist.imageUrl}" alt="Image">
            </figure>
          </div>
          <article class="card-content">
            <p class="content is-size-4">
              ${playlist.description}
            </p>
          </article>
          <footer class="card-footer">
            <button class="card-footer-item button has-background-grey-lighter">Play</button>
            <button class="card-footer-item button has-background-grey-lighter">Open</button>
            <button class="card-footer-item button has-background-grey-lighter">Remove</button>
          </footer>
        </section>
      </div>`
      main.innerHTML = main.innerHTML + element;
    });
  });
</script>
```

Before vs After

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const playlists = [
      {
        name: "Chill",
        description: "A playlist to chill to",
        imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
      },
      {
        name: "Focus",
        description: "A playlist to focus to",
        imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
      },
      {
        name: "Let Off Steam",
        description: "A playlist for one of those days",
        imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
      },
      {
        name: "Rock",
        description: "Rock your socks",
        imageUrl: "https://source.unsplash.com/grayscale-photo-of-person-in-hoodie-top-watching-a-concert-97p-JwqdyW4"
      }
    ];

    const main = document.querySelector('main');
    playlists.forEach((playlist) => {
      main.innerHTML = main.innerHTML + createPlaylistItem(playlist);
    });
  });
</script>
```

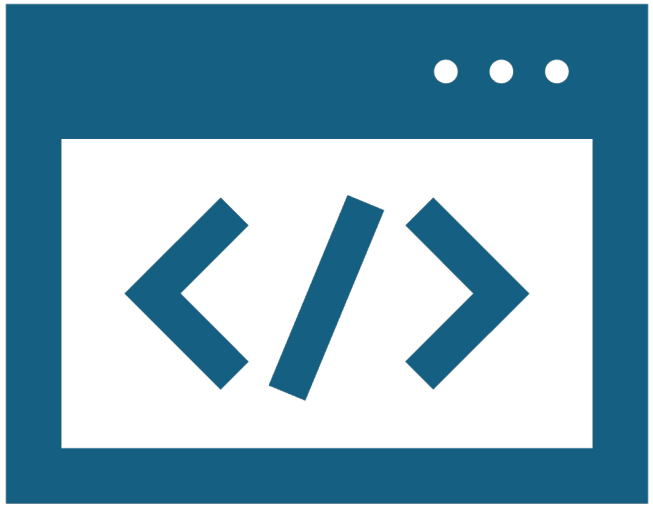
Lots of new words...

model

- A representation of a real-world domain in data, in our domain of music streaming a model is made up of
  - Entities ➔ playlists, songs, artists, albums
  - Attributes ➔ the properties that make up our entities, e.g. album name, album art
  - Relationships ➔ How are entities related? A playlist has many songs, an artist has many albums
- A domain refers to a specific subject area, problem space, or application area that a software system or model is designed to address such as music streaming, banking, e-commerce, education, movie industry, home espresso making

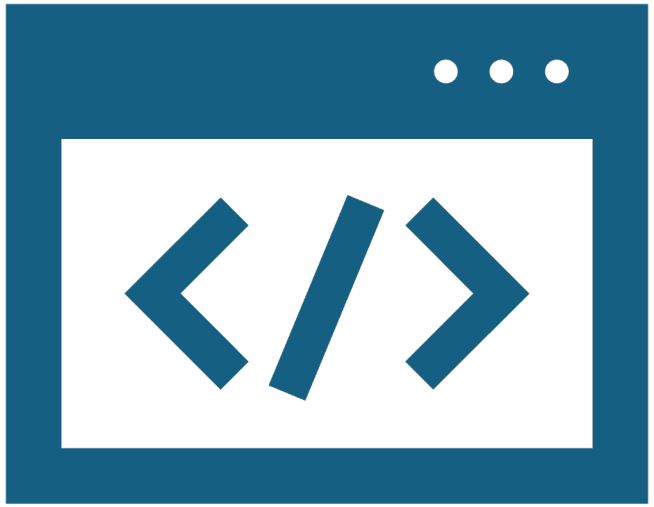
view

- A view is our user interface, in this case, our HTML, CSS and JavaScript
- It is what the user sees and interacts with
- A view is used to allow viewing of the data model as well as interacting with the data model

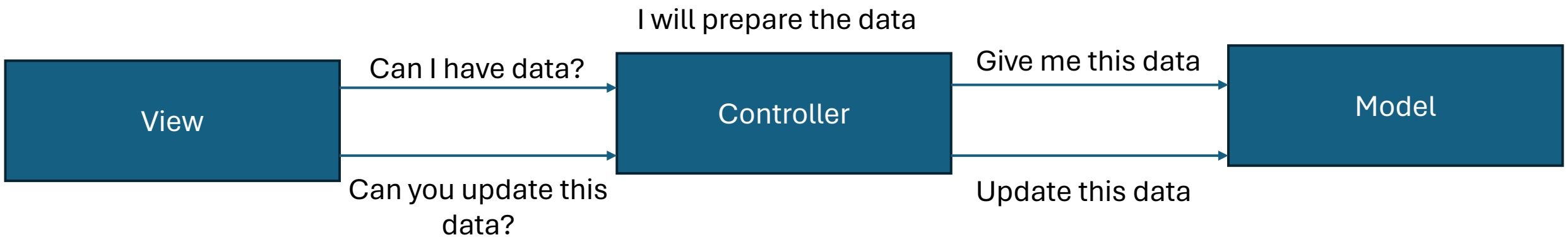


component

- A reusable piece of user interface



Model – View –Controller (kind of)

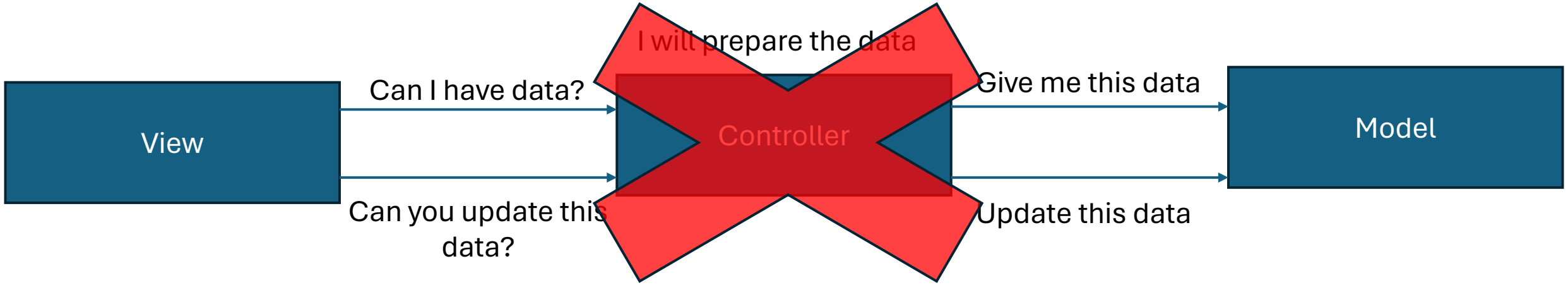


- Model View Controller, or MVC, is typically an event driven pattern for creating a data driven user interface
- We will use a simplified version to help us compartmentalise our data from our view without the events
  - Our view will request data from the controller
  - The controller will format the user request so that it is suitable for model, it may perform multiple queries for one user request
  - The controller will format the data so that it makes sense to the view from the perspective of it's request
- This is a layered approach to application design to help us enforce Separation of Concerns
  - The view is concerned with displaying the data
  - The controller is concerned with formatting and preparing user requests and data
  - The model is concerned with storing and allowing access to data

Eh?

That’s a lot, we will return to Model-View-Controller next week, I just wanted to plant the seed

Model – View –Controller (kind of)



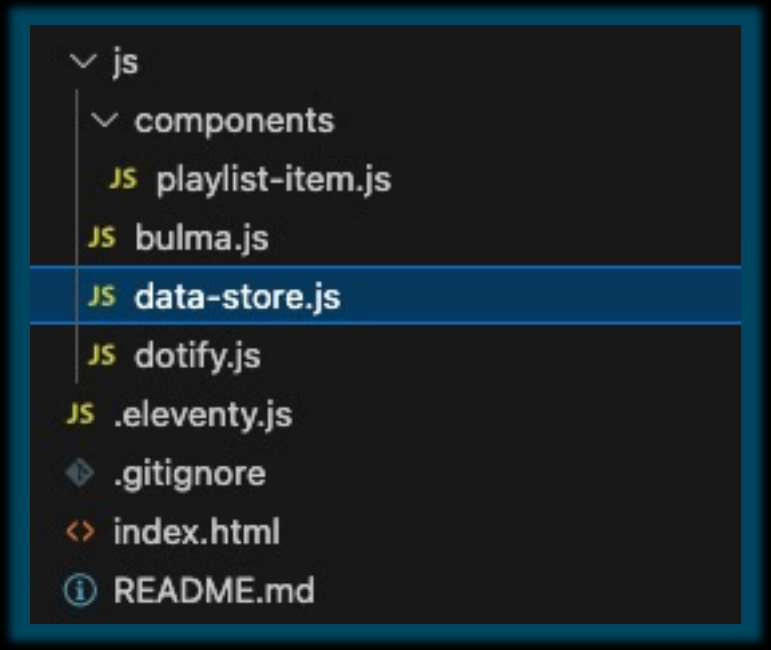
- For now, we will ignore the controller and allow the view to interact directly with the model

Introduce a data store

We have our view and our model

BUT our model is within our view

To move our model from our view, we will introduce a data store

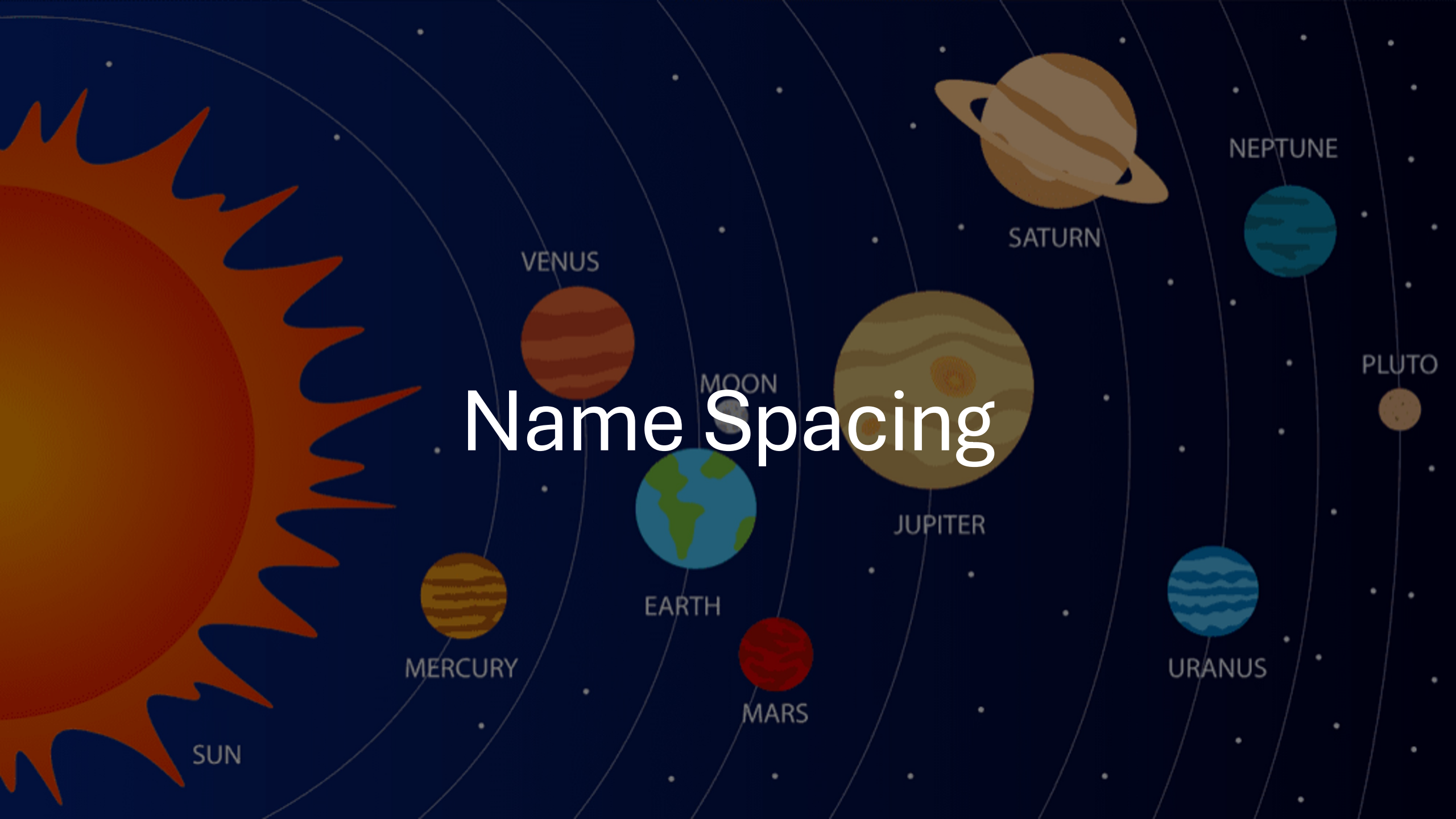



```
const dataStore = {
    data: [
        {
            name: "Chill",
            description: "A playlist to chill to",
            imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
        },
        {
            name: "Focus",
            description: "A playlist to focus to",
            imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
        },
        {
            name: "Let Off Steam",
            description: "A playlist for one of those days",
            imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
        },
        {
            name: "Rock",
            description: "Rock your socks",
            imageUrl: "https://source.unsplash.com/
            grayscale-photo-of-person-in-hoodie-top-watching-a-concert-97p-JwqdyW4"
        }
    ],
    list(){
        return this.data
    },
}
```

```
const dataStore = {
    data: [
        {
            name: "Chill",
            description: "A playlist to chill to",
            imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
        },
        {
            name: "Focus",
            description: "A playlist to focus to",
            imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
        },
        {
            name: "Let Off Steam",
            description: "A playlist for one of those days",
            imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
        },
        {
            name: "Rock",
            description: "Rock your socks",
            imageUrl: "https://source.unsplash.com/
            grayscale-photo-of-person-in-hoodie-top-watching-a-concert-97p-JwqdyW4"
        }
    ],
    list(){
        return this.data
    },
}
```

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const main = document.querySelector('main');
    dataStore.list().forEach((playlist) => {
      main.innerHTML = main.innerHTML + createPlaylistItem(playlist);
    });
  });
</script>
```


Last piece



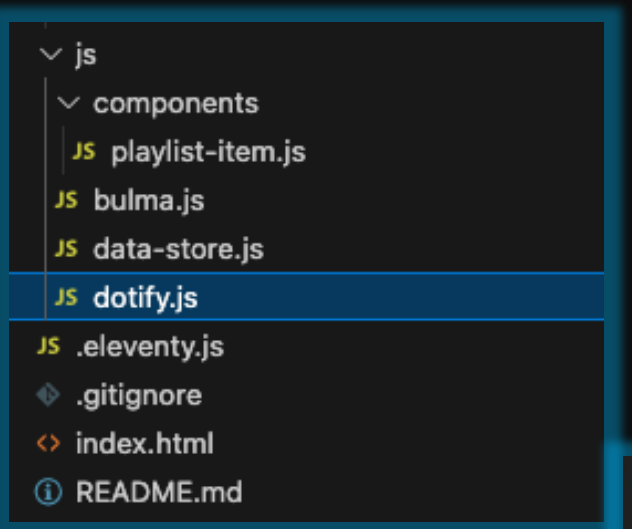
Name Spacing

Name Space

- So far we have created:
  - dataStore
  - createPlaylistItem
- These reside in the global scope
- If another application adds or manipulates a dataStore or a createPlaylistItem function, it could cause issues on the page.
- To help prevent this, we introduce our own namespace

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const main = document.querySelector('main');
    dataStore.list().forEach((playlist) => {
      main.innerHTML = main.innerHTML + createPlaylistItem(playlist);
    });
  });
</script>
```

dotify namespace




```
window.dotify = {
  components: {},
  dataStore: {}
}       You, 2 weeks
```

```
dotify.dataStore.data =  [
    {
        name: "Chill",
        description: "A playlist to chill to",
        imageUrl: "https://source.unsplash.com/person-holding-coffee-mug-cspncX4cUnQ"
    },
    {
        name: "Focus",
        description: "A playlist to focus to",
        imageUrl: "https://source.unsplash.com/person-holding-camera-lens-7KLa-xLbSXA"
    },
    {
        name: "Let Off Steam",
        description: "A playlist for one of those days",
        imageUrl: "https://source.unsplash.com/black-and-white-electric-guitar-TW-wknV1oZo"
    },
    {
        name: "Rock",
        description: "Rock your socks",
        imageUrl: "https://source.unsplash.com/grayscale-photo-of-person-in-hoodie-top-watching-a-concert-97p-JwqdyW4"
    }
]

dotify.dataStore.list = () => {
    return dotify.dataStore.data
}       You, 5 days ago • added data store
```

```
window.dotify.components.createPlaylistItem = (playlist) => {
    return `
    <div class="column is-4">
      <section class="card has-text-centered">
        <header class="card-header">
          <p class="card-header-title is-size-4 is-centered">
            ${playlist.name}
          </p>
        </header>
        <div class="card-image">
          <figure class="image">
            <img src="${playlist.imageUrl}" alt="Image">
          </figure>
        </div>
        <article class="card-content">
          <p class="content is-size-4">
            ${playlist.description}
          </p>
        </article>
        <footer class="card-footer">
          <button class="card-footer-item button has-background-grey-lighter">Play</button>
          <button class="card-footer-item button has-background-grey-lighter">Open</button>
          <button class="card-footer-item button has-background-grey-lighter">Remove</button>
        </footer>
      </section>
    </div>`
}       You, 5 days ago • added data store
```

Before vs After

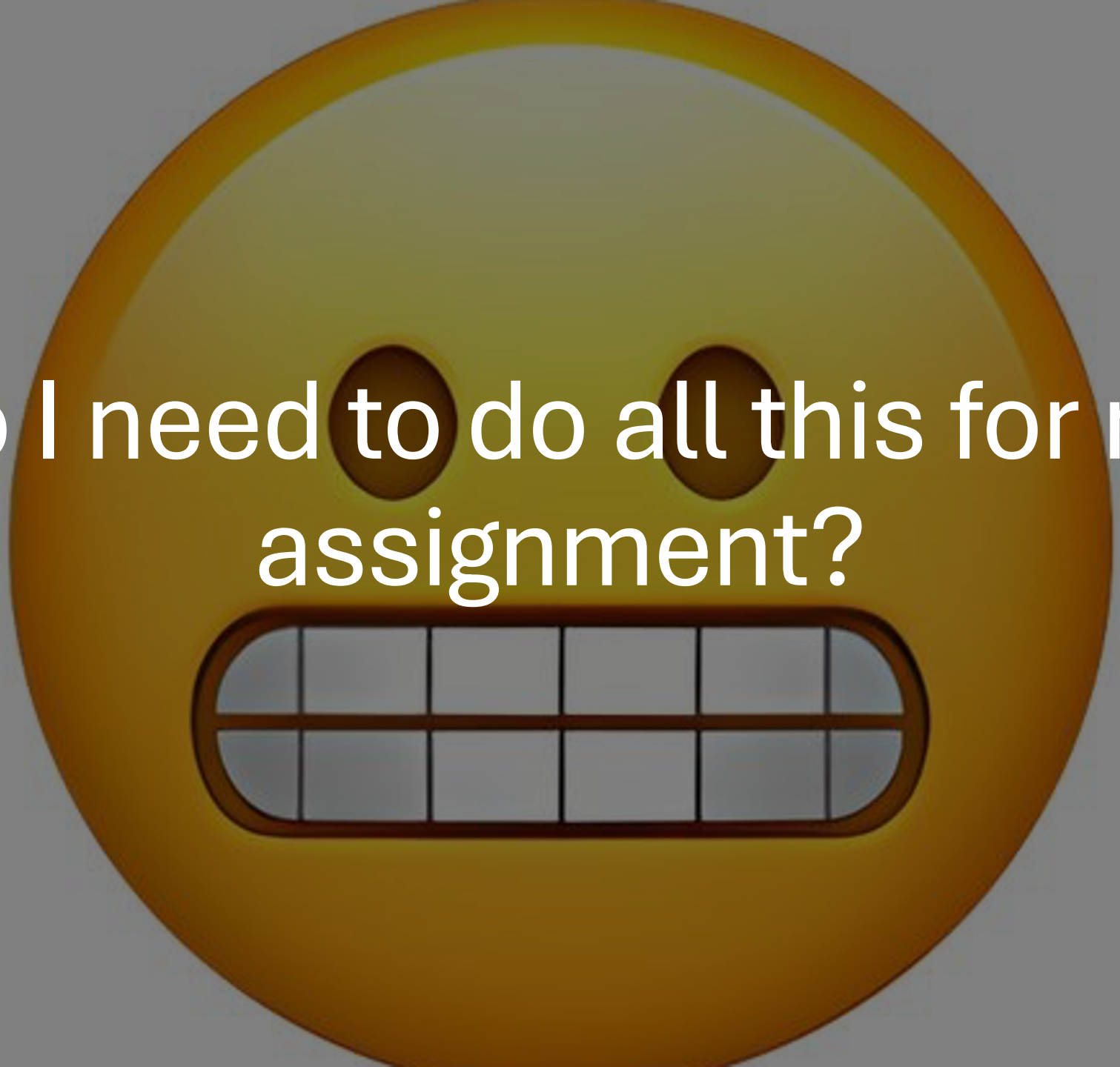
```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const playlistOneHeading = document.querySelector('#playlist-1-heading');
    playlistOneHeading.innerHTML = 'Chill';

    const playlistOneImage = document.querySelector('#playlist-1-image');
    playlistOneImage.src = 'https://source.unsplash.com/
    person-holding-coffee-mug-cspncX4cUnQ';

    const playlistOneDescription = document.querySelector('#playlist-1-description');
    playlistOneDescription.innerHTML = 'A playlist to chill your mind';

    const playlistTwoHeading = document.querySelector('#playlist-2-heading');
    playlistTwoHeading.innerHTML = 'Focus';

    const playlistTwoImage = document.querySelector('#playlist-2-image');
    playlistTwoImage.src = 'https://source.unsplash.com/
    person-holding-camera-lens-7KLa-xLbSXA';

    const playlistTwoDescription = document.querySelector('#playlist-2-description');
    playlistTwoDescription.innerHTML = 'A playlist to focus to';

    const playlistThreeHeading = document.querySelector('#playlist-3-heading');
    playlistThreeHeading.innerHTML = 'Let Off Steam';

    const playlistThreeImage = document.querySelector('#playlist-3-image');
    playlistThreeImage.src = 'https://source.unsplash.com/
    black-and-white-electric-guitar-TW-wknV1oZo';

    const playlistThreeDescription = document.querySelector('#playlist-3-description');
    playlistThreeDescription.innerHTML = 'A playlist for one of those days';
  });
</script>
```

```
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const main = document.querySelector('main');
    dotify.dataStore.list().forEach((playlist) => {
      main.innerHTML = main.innerHTML + dotify.components.createPlaylistItem(playlist);
    });
  });
</script>
```

- Our UI (view) is now data driven via a data store (model)
- We now have a dotify namespace to prevent clashes
- We now have the ability to create components

Today

- What is a data model?
- Building a UI from data
- Introducing components
- What is model view controller?
- Introducing a datastore
- Creating a namespace



Do I need to do all this for my assignment?

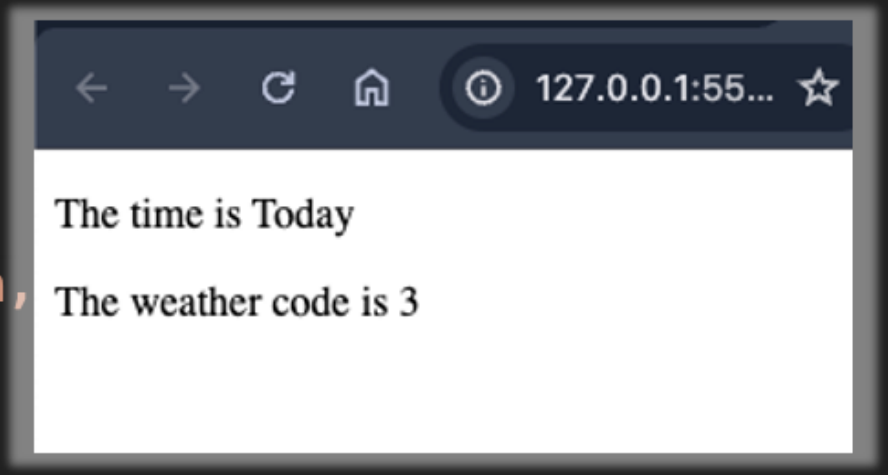
No

Passing Pathway

Grade Band	JavaScript	Dev Experience	Submission and Deployment	Release	Release Features
	Understanding of best practices: • Usage of object	Clearly laid out project cture – HTML, CSS and JavaScript	Zip file to moodle	POC – 1 - City Focus 1 1 City with today's weather	• Weather Code Mapped to Correct Weather • Display City Name • Clear it is today's weather • Max Temp and Wind
baseline	• Add and Remove elements from the DOM without the need for a screen refresh	DRY - Very little Repetition of HTML, JS and Styles	+ github repository link + github commit history	POC – 2 – City Focus 2 + current hour's weather	• Added current hours forecast • Correctly mapped hourly weather code • Correct hour taken from browser
good	• Understanding of application name spacing and scopes	Layered JavaScript Architecture - MVC	+ github tags	POC – 3 – City Focus 3 + 7 days summary	• Clear and concise summary for the 7 days • Weather code correctly mapped • Additional weather information
excellent	• Well maintained utility file(s) • Clear and concise loading of javascript • Functions utilized correctly	Excellent README	+ Manual upload to netlify	Release -1 + Dashboard	• Focus on user experience • Ability to navigate to city view • Ability to navigate back to dashboard • Usage of URL paths and parameters
outstanding			+ github push to master deploys to netlify	Release – 2 + Configure and Persist User Preferences	• User preferences user interface • User preferences saved in localStorage • Clear navigation • User preferences affect how the dashboard render • Reset preferences
amazing			-	Release – 3 + Build out your own features	• Start small and build from there • Do not start this without a lot of the previous releases done, don't feel pressure to get here

Considered more advanced

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    <title>Test Data</title>
    <script src="weather_data.js"></script>
    <script>
        document.addEventListener("DOMContentLoaded", () => {
        const currentCity = "amsterdam";
        const currentCityData = weatherData[currentCity + "_daily"]
        const time = document.getElementById("time");
        const weatherCode = document.getElementById("weatherCode");

        time.innerHTML = currentCityData.daily.time[0];
        weatherCode.innerHTML = currentCityData.daily.weather_code[0];
    });
    </script>
</head>
<body>
    <p>The time is <span id="time"></span></p>
    <p>The weather code is <span id="weatherCode"></span></p>
</body>
</html>
```



This page has been added to the data.zip