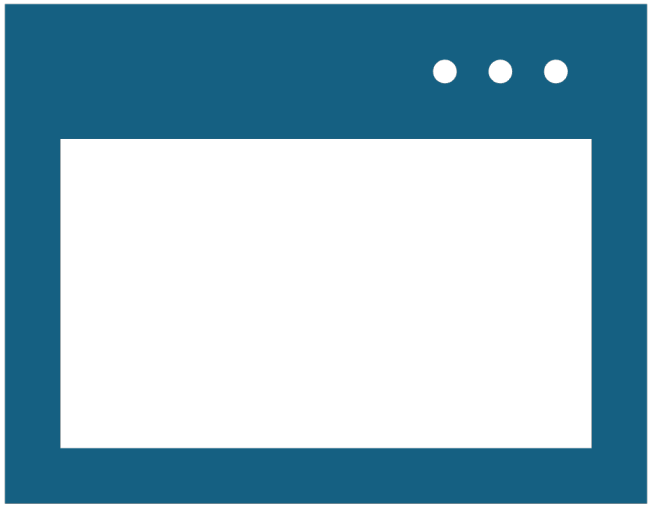
localStorage

Web Development 1

John Rellis

What is localStorage?

- The localStorage read-only property of the window interface allows you to access a Storage object for the Document's origin; the stored data is saved across browser sessions.
- localStorage data for a document loaded in a "private browsing" or "incognito" session is cleared when the last "private" tab is closed
- https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage



Storage?

- https://developer.mozilla.org/en-US/docs/Web/API/Storage
- The Storage interface of the Web Storage API provides access to a particular domain's session or local storage.
- It allows, for example, the addition, modification, or deletion of stored data items.
- To manipulate, for instance, the session storage for a domain, a call to Window.sessionStorage is made; whereas for local storage the call is made to Window.localStorage.

Instance methods

`Storage.key()`

When passed a number `n`, this method will return the name of the nth key in the storage.

`Storage.getItem()`

When passed a key name, will return that key's value.

`Storage.setItem()`

When passed a key name and value, will add that key to the storage, or update that key's value if it already exists.

`Storage.removeItem()`

When passed a key name, will remove that key from the storage.

`Storage.clear()`

When invoked, will empty all keys out of the storage.

Examples on MDN

- Here we access a Storage object by calling localStorage.
- We first test whether the local storage contains data items using !localStorage.getItem('bgcolor').
- If it does, we run a function called setStyles() that grabs the data items using Storage.getItem() and uses those values to update page styles.
- If it doesn't, we run another function, populateStorage(), which uses Storage.setItem() to set the item values, then runs setStyles().

```
if (!localStorage.getItem("bgcolor")) {
  populateStorage();
} else {
  setStyles();
}

function populateStorage() {
  localStorage.setItem("bgcolor", document.getElementById("bgcolor").value);
  localStorage.setItem("font", document.getElementById("font").value);
  localStorage.setItem("image", document.getElementById("image").value);

  setStyles();
}

function setStyles() {
  const currentColor = localStorage.getItem("bgcolor");
  const currentFont = localStorage.getItem("font");
  const currentImage = localStorage.getItem("image");

  document.getElementById("bgcolor").value = currentColor;
  document.getElementById("font").value = currentFont;
  document.getElementById("image").value = currentImage;

  htmlElem.style.backgroundColor = `#${currentColor}`;
  pElem.style.fontFamily = currentFont;
  imgElem.setAttribute("src", currentImage);
}
```

Examples on MDN

JS

```
localStorage.setItem("myCat", "Tom");
```

The syntax for reading the `localStorage` item is as follows:

JS

```
const cat = localStorage.getItem("myCat");
```

The syntax for removing the `localStorage` item is as follows:

JS

```
localStorage.removeItem("myCat");
```

The syntax for removing all the `localStorage` items is as follows:

JS

```
localStorage.clear();
```

localStorage

- localStorage is a key/value store of strings
- It is important that both the keys and values are recognised as strings
- There is an implicit call to “toString” if you give it a non string value
- It survives page refreshes and navigation away
- It is specific to the domain e.g. google.com, linkedin.com etc
- You cannot access localStorage from another domain
- It is not secure and can be edited on the fly in the browser

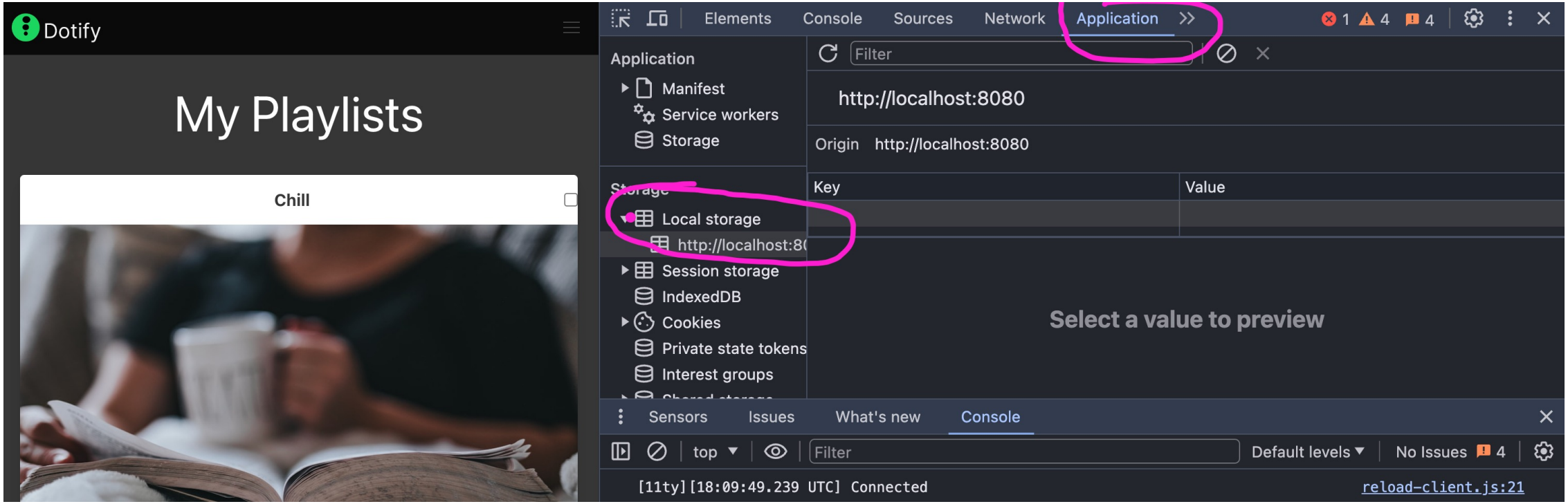
```
localStorage.setItem('isReal', true);
console.log(typeof localStorage.getItem('isReal')); // string
console.log(typeof localStorage.getItem('isReal') === true); // false
console.log(typeof localStorage.getItem('isReal') === 'true'); // true
```

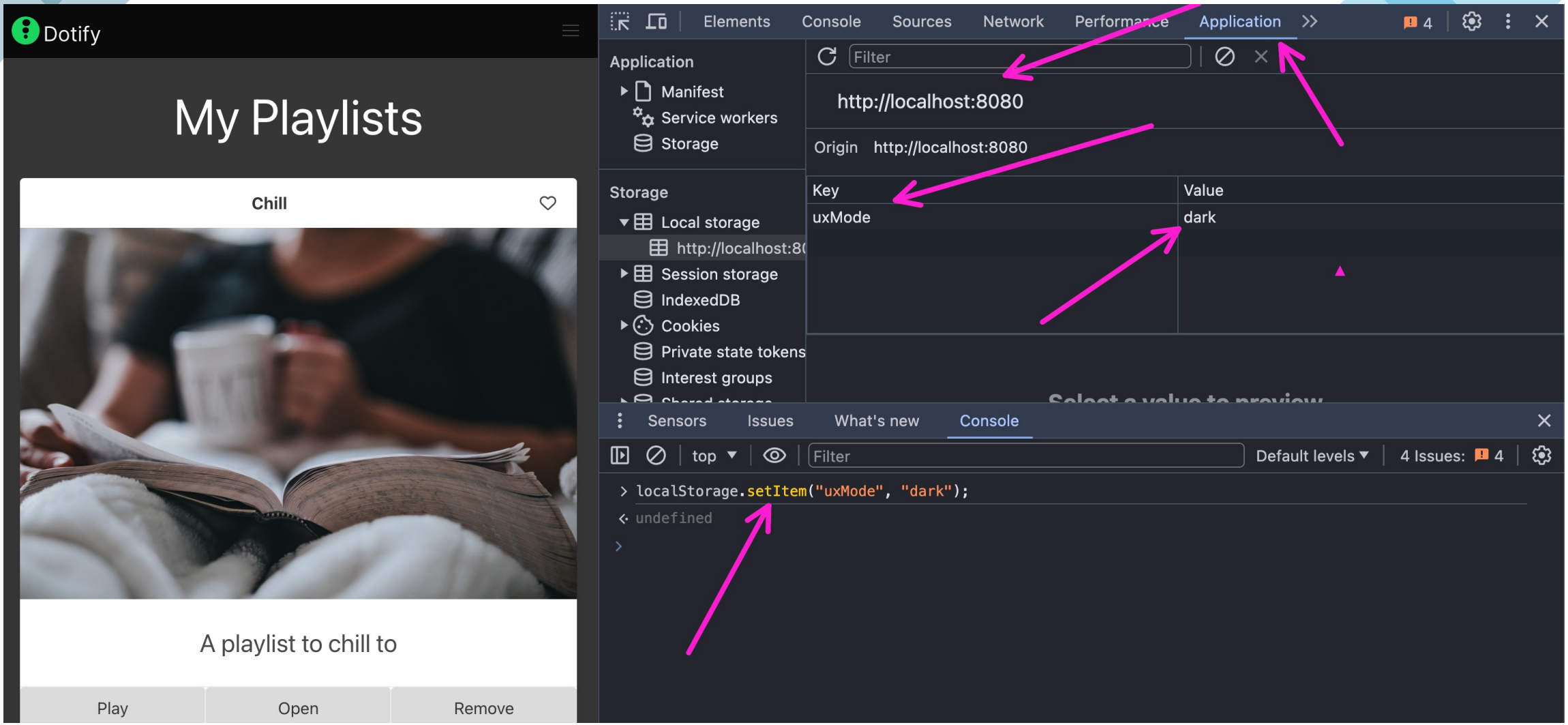
Web Storage API

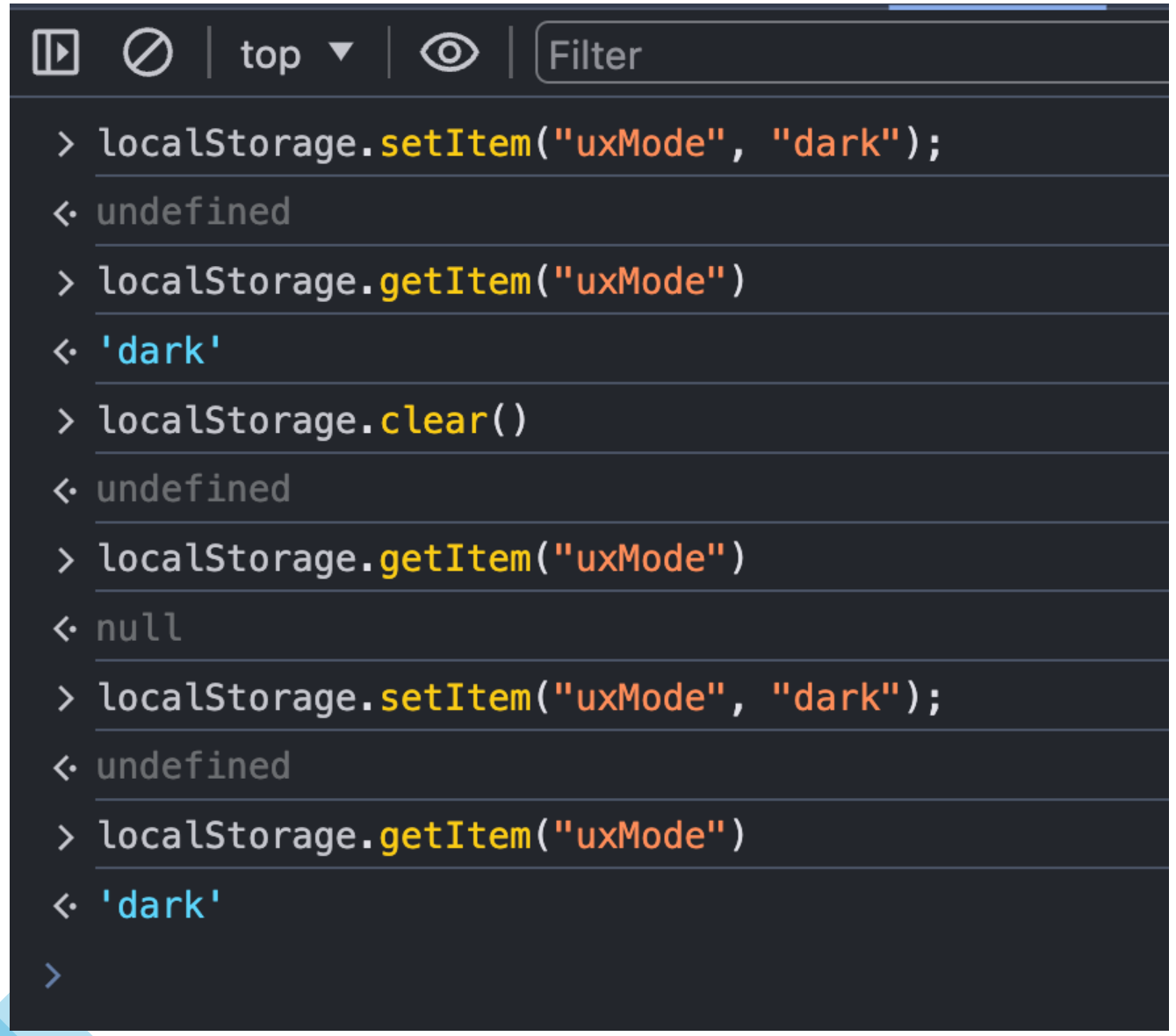
- Storage objects (such as localStorage) are simple key-value stores, similar to objects, but they stay intact through page loads.
- The keys and the values are always strings (note that, as with objects, integer keys will be automatically converted to strings).
- You can access these values like an object, or with the Storage.getItem() and Storage.setItem() methods.
- It's recommended to use the Web Storage API (setItem, getItem, removeItem, key, length) to prevent the pitfalls associated with using plain objects as key-value stores.
- These three lines all set the (same) colorSetting entry:

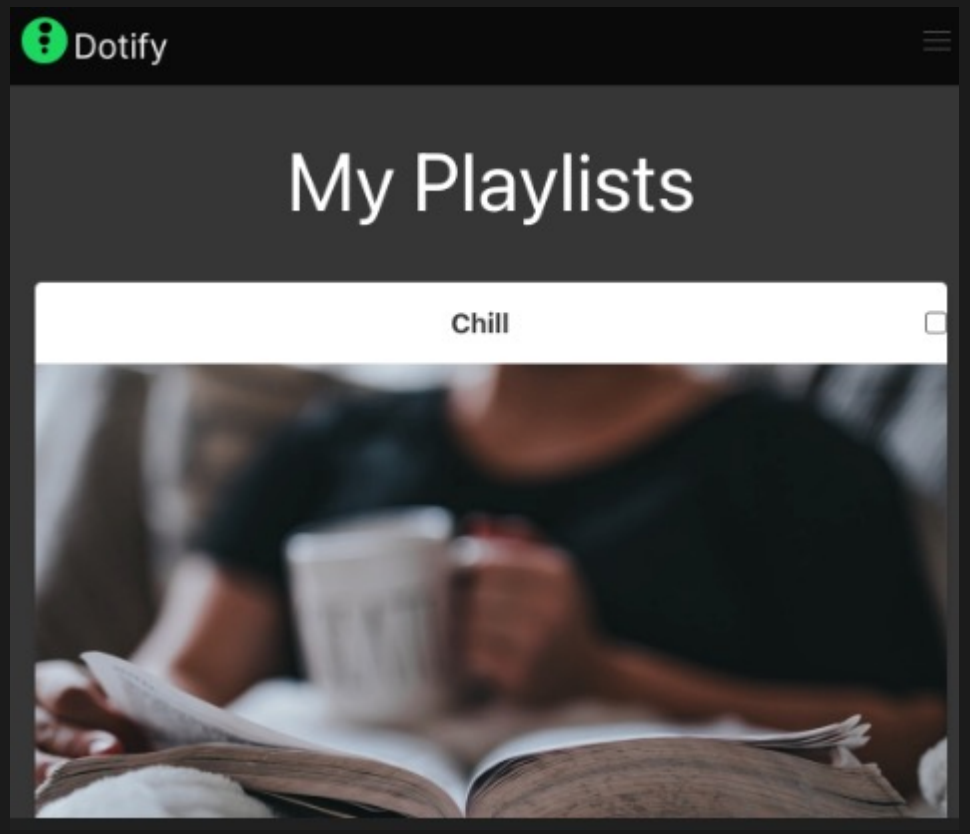
```
localStorage.colorSetting = "#a4509b";
localStorage["colorSetting"] = "#a4509b";
localStorage.setItem("colorSetting", "#a4509b");
```

Using localStorage in the Browser




Dotify
My Playlists
Chill
A playlist to chill to
Play
Open
Remove
Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
http://localhost:8
Session storage
IndexedDB
Cookies
Private state tokens
Interest groups
Filter
http://localhost:8080
Origin http://localhost:8080
Key
Value
uxMode
dark
Sensors
Issues
What's new
Console
top
Filter
Default levels
4 Issues: 4
> localStorage.setItem("uxMode", "dark");
undefined


top
Filter
localStorage.setItem("uxMode", "dark");
undefined
localStorage.getItem("uxMode")
'dark'
localStorage.clear()
undefined
localStorage.getItem("uxMode")
null
localStorage.setItem("uxMode", "dark");
undefined
localStorage.getItem("uxMode")
'dark'

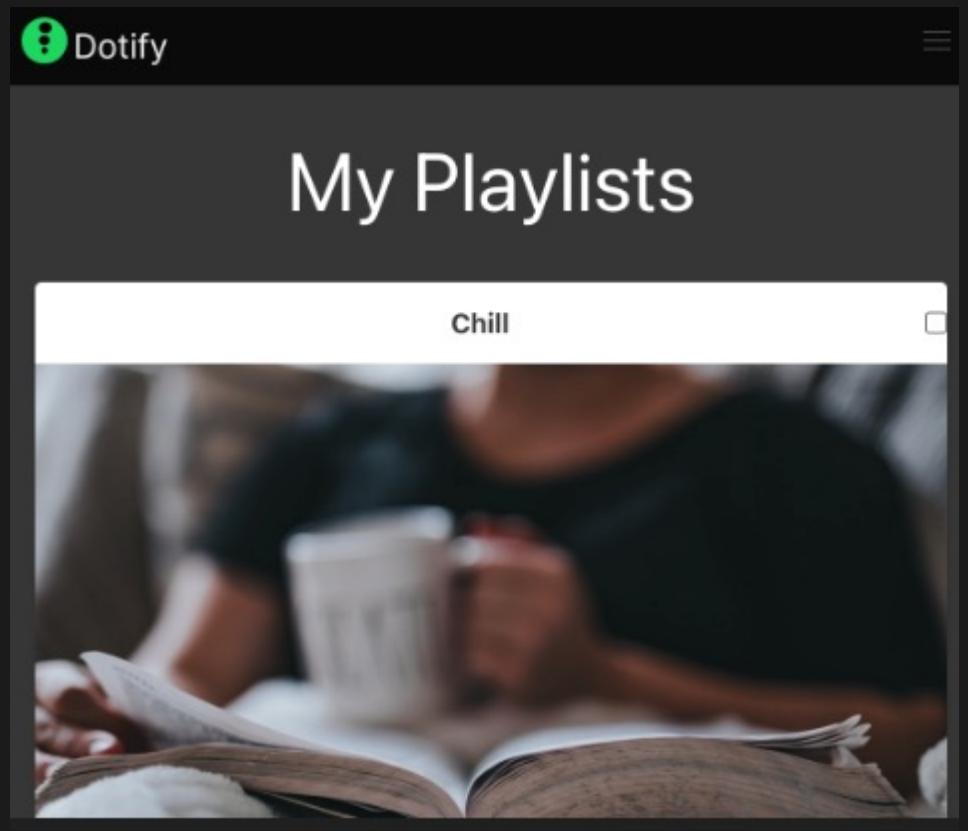


Save an item to localStorage

```
<header class="card-header">
  <p class="card-header-title is-centered">
    ${playlist.name}
  </p>
<input type="checkbox" id="fave-${playlist.name}" class="checkbox"/>
</header>
```

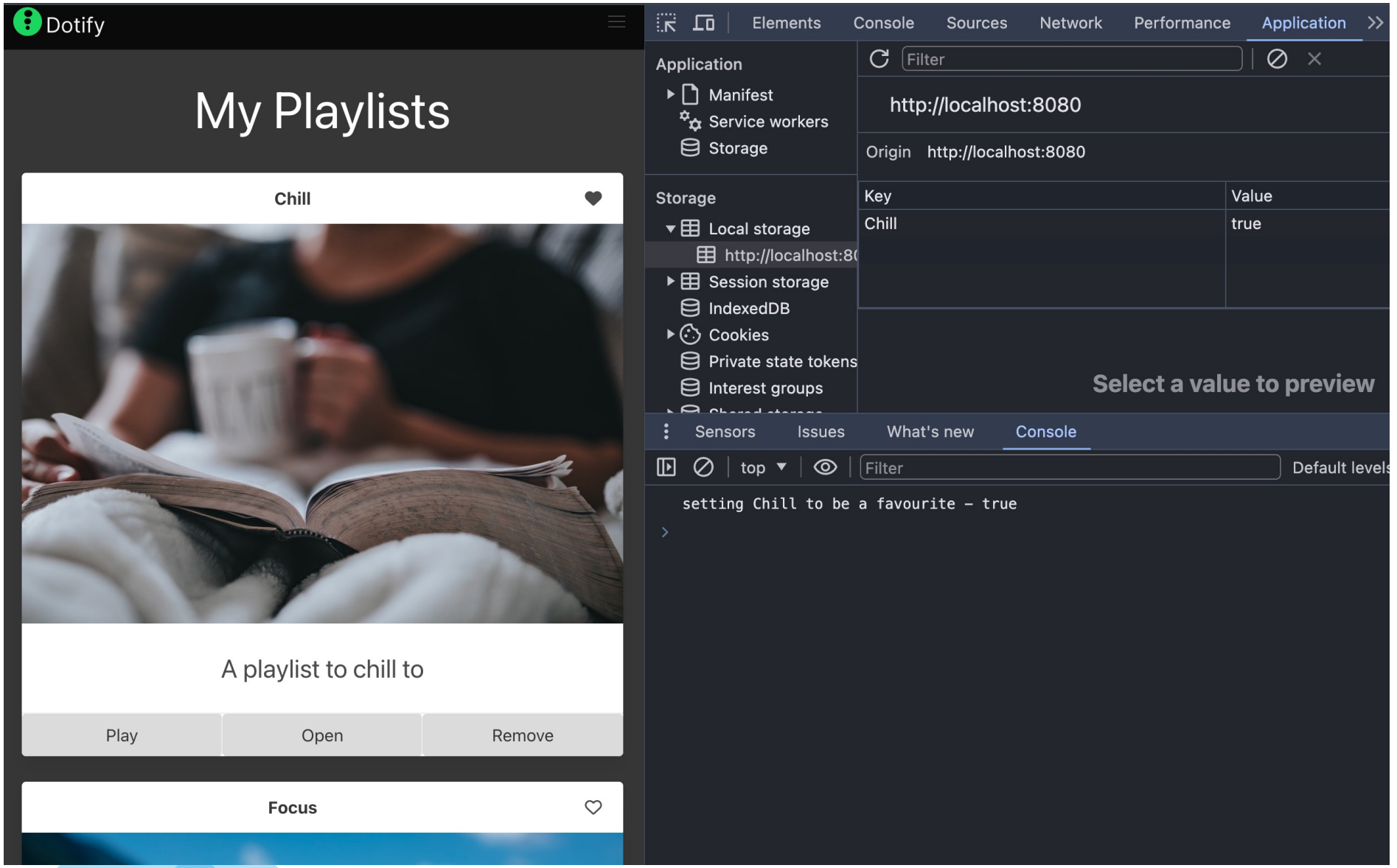
```
document.querySelectorAll("[id^=fave-]").forEach(checkbox => {
    checkbox.addEventListener('click', (event) => {
        const playlistName = event.target.id.replace('fave-','');
        const isFavourite = event.target.checked;
        localStorage.setItem(playlistName, isFavourite);
    })
});
```

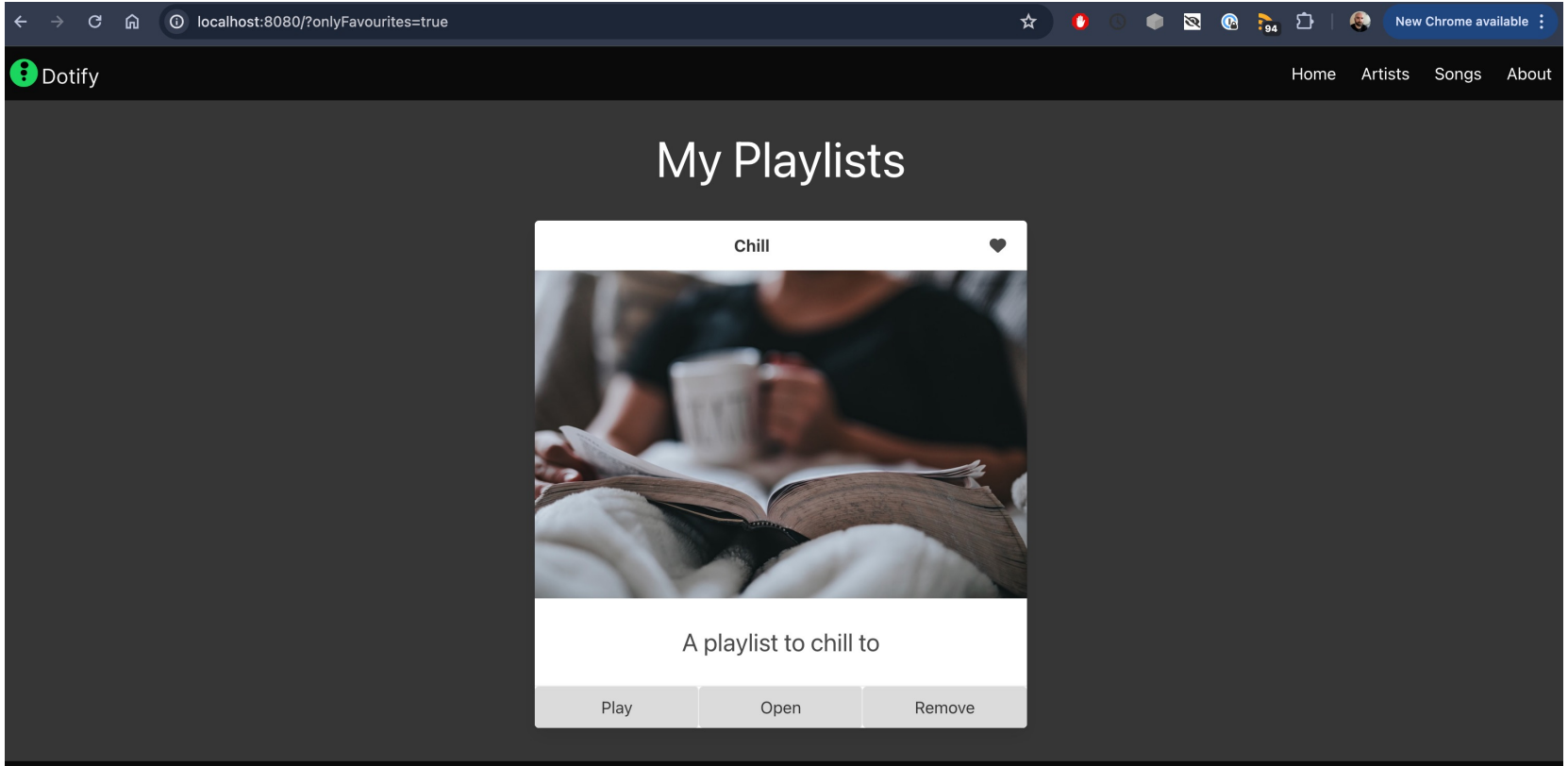
Reading an item to localStorage




```
<header class="card-header">
  <p class="card-header-title is-centered">
    ${playlist.name}
  </p>
<input type="checkbox" id="fave-${playlist.name}" class="checkbox"/>
</header>
```

```
document.querySelectorAll("[id^=fave-]").forEach(checkbox => {
    const playlistName = checkbox.id.replace('fave-','');
    const isFavourite = localStorage.getItem(playlistName);
    console.log(`${playlistName} - is fave - ${isFavourite}`);
});
```


Dotify
My Playlists
Chill
A playlist to chill to
Play
Open
Remove
Focus
Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service workers
Storage
Filter
http://localhost:8080
Origin http://localhost:8080
Storage
Local storage
http://localhost:8
Session storage
IndexedDB
Cookies
Private state tokens
Interest groups
Key
Value
Chill
true
Select a value to preview
Sensors
Issues
What's new
Console
top
Filter
Default levels
setting Chill to be a favourite – true



Or you could make a favourites only page

```
const onlyFavourites = urlParams.get('onlyFavourites') === 'true';

dotify.dataStore.list().forEach((playlist) => {
    if(onlyFavourites) {
        const isFavourite = localStorage.getItem(playlist.name) === 'true';
        if(isFavourite) {
          main.innerHTML = main.innerHTML + dotify.components.createPlaylistItem(playlist);
        }
    } else {
        main.innerHTML = main.innerHTML + dotify.components.createPlaylistItem(playlist);
    }
});
```

Lab Preview