GirlCode 2019

Javascript Workshop 2

Sponsored by Entelect Software

Image credit: Duart Breedt

TOOLS.js

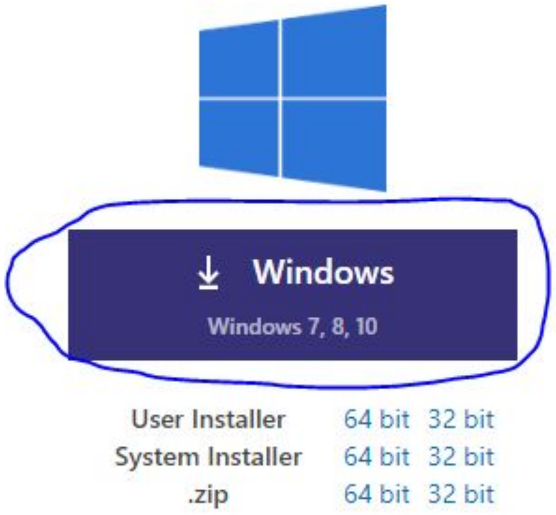
Tools for this workshop

Tooling to make our experience easier

- ➔ **Visual Studio Code**
  A modern **I**ntegrated **D**evelopment **E**nvironment (**IDE**) - Where we're going to write our code
- ➔ **Web Browser of your choice**
  Chrome, Firefox, Edge, Internet explorer
- ➔ **.HTML and .JS Files**
  You can use .HTML files from an earlier workshop or create some new ones - It's up to you!

IDE Setup

code.visualstudio.com/download



User Installer	64 bit	32 bit
System Installer	64 bit	32 bit
.zip	64 bit	32 bit

1. Open extensions menu
2. Search for GirlCode
3. Install the extension

If you don't have internet access, please let us know!

Slides

<<GET THIS LINK>>

If you don’t have internet access, please let us know!

Developer Tools

Simply press F12 on your keyboard and navigate over to your console section.

Please call one of us over if you need help!

.HTML and .js files

Each section has its own exercise files. We will instruct you on when to move to the next exercise

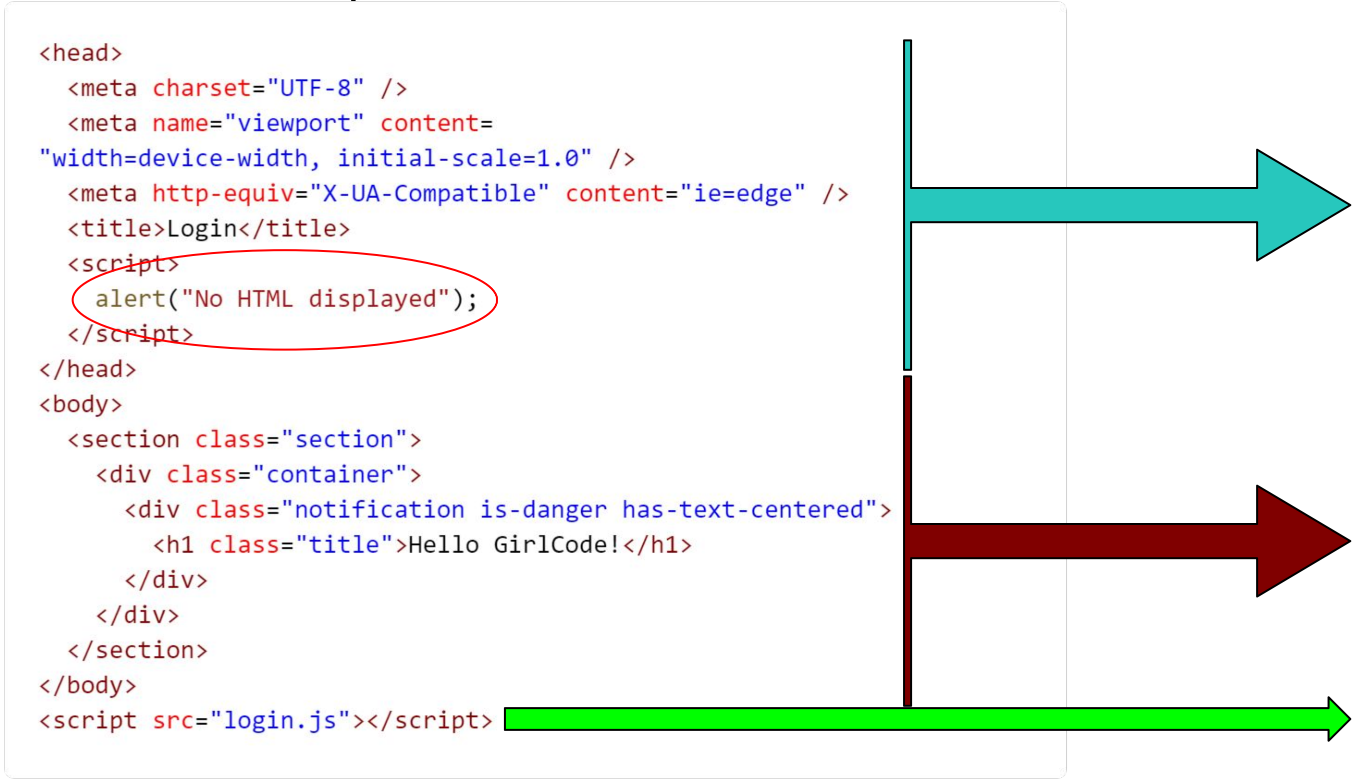
Closures.js

Closures

- ➔ **Important for order of execution**
  Used for explaining what is going on in code
- ➔ **Define scoping**
  Used to tell a story about the problem. That is very specific to the problems domain. Can also be used to document use of functions.

Closures Exercise

Closures are important for order of execution



```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content=
"width=device-width, initial-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>Login</title>
  <script>
    alert("No HTML displayed");
  </script>
</head>
<body>
  <section class="section">
    <div class="container">
      <div class="notification is-danger has-text-centered">
        <h1 class="title">Hello GirlCode!</h1>
      </div>
    </div>
  </section>
</body>
<script src="login.js"></script>
```

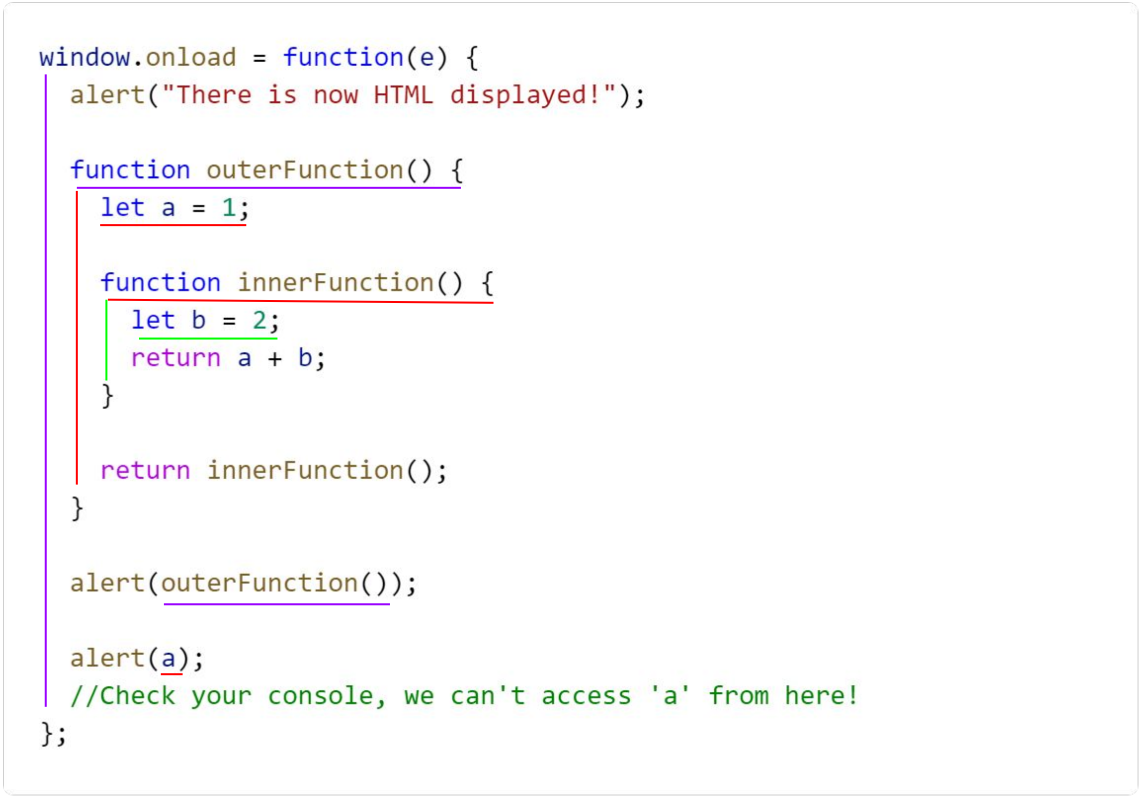
Head always happens first

Body happens next

This call will happen after the body

Closures Exercise 2

Closures are important for order of execution



```
window.onload = function(e) {
  alert("There is now HTML displayed!");

  function outerFunction() {
    let a = 1;

    function innerFunction() {
      let b = 2;
      return a + b;
    }

    return innerFunction();
  }

  alert(outerFunction());

  alert(a);
  //Check your console, we can't access 'a' from here!
};
```

Selectors.js

Selectors

- ➔ **Used to Select items on the HTML Page**
  Labels, Text, images and more
- ➔ **Uses attributes to select items**
  Get By ID, Class, or others
- ➔ **Allows us to get values from and put values back into the page**
  This is the start of how we use JS to cause changes to our websites (Dynamic web pages)

Selectors

➔ **document**

This variable is given to us by the browser, and allows us to access the page

➔ **document.getElementByID(“subtitle”)**

Provides us access to an HTML element using the “id” tag on that element

➔ **document.getElementsByClassName(“card-title”)**

Gives us an Array of HTML elements which have the same “class” tag

Selectors Exercise

Using Selectors to get data

```
// Select the Title and Subtitle from the HTML file
let title = document.getElementsByClassName("card-title");
let subtitle = document.getElementById("subtitle");

//Use console.log to print their values to the console
console.log(title[0].innerHTML);
console.log(subtitle.innerHTML);
```

Selectors Exercise

Using Selectors to change data

```
// Select the Title and Subtitle from the HTML file
let title = document.getElementsByClassName("card-title");
let subtitle = document.getElementById("subtitle");

// Use innerHTML to modify their content
title[0].innerHTML = "Changed Title";
subtitle.innerHTML = "Changed Subtitle"
```

Input Handling.js

Input Handling

➔ **We need to be able to get a user's input!**

We can combine selectors with html <input> tags to do this

➔ **Validations**

We can also use Javascript to validate a user's input, such as a long enough password, and more

We're going to see these in action later!

Getting an Input’s value

```
// Get the Email and Password elements
let emailInput = document.getElementById("email");
let passwordInput = document.getElementById("password");

// Use "alert" to display their values
alert("Email: " + emailInput.value);
alert("Password: " + passwordInput.value);
```

Events.js

Events

➔ **We need to handle actions from users**

We do this with events.

➔ **How are events added**

We use the on selector on html elements or the event listners

Click Event

```
function clickTheButton() {
    let emailAddress = document.getElementById("email").value
    if (emailAddress === '') {
        emailAddress = 'Please type in the email address.'
    }
    alert(emailAddress);
}
```

```
<button type="button" class="btn btn-primary" onclick="clickTheButton()">Login</button>
```

Double Click Event

```
function doubleCLickTheButton() {
    let emailAddress = document.getElementById("email").value
    if (emailAddress === '') {
        emailAddress = 'Please type in the email address.'
    }
    alert(emailAddress);
}
```

```
<button type="button" class="btn btn-primary" ondblclick="doubleCLickTheButton()">
Login</button>
```

Mouse Up and Mouse Down Event

```
function mouseDownButton() {
    let emailAddress = document.getElementById("email").value
    if (emailAddress === '') {
        emailAddress = 'Please type in the email address.'
    }
    console.log('Mouse Down ' + emailAddress);
}

function mouseUpButton() {
    let emailAddress = document.getElementById("email").value
    if (emailAddress === '') {
        emailAddress = 'Please type in the email address.'
    }
    console.log('Mouse up ' + emailAddress);
```

```
<button onmousedown="mouseDownButton()" onmouseup="mouseUpButton()" type="button"
class="btn btn-primary" >Login</button>
```

Changing the DOM.js

Changing the dom

- ➔ **What is the dom**
  The dom is short for Document Object Model

- ➔ **Why do we need to change the dom**
  This is how javascript adds to a web page.

- ➔ **How do we add to the**
  There are multiple pure javascript functions that can be used

Add a single value to the dom

```javascript
function AddValueToDom() {
    let emailAddress = document.getElementById("email").value
    if (emailAddress === '') {
        emailAddress = 'Please type in the email address.'
    }
    let nodeToAppendTo = document.getElementById("AppendDataHere");
    nodeToAppendTo.insertAdjacentHTML("afterend", "<p>"+ emailAddress +"</p>"
};
```

Add a multiple values to the dom

```
let listOfNames = ['OLIVIA' ,'RUBY' ,'EMILY' ,'GRACE' ,'JESSICA' ,'CHLOE' ,
'SOPHIE' ,'LILY' ,'AMELIA' ,'EVIE' ,'MIA' ,'ELLA' ,'CHARLOTTE' ,'LUCY' ,'MEGAN']
function AddValuesToDom() {
    let nodeToAppendTo = document.getElementById("AppendDataHere");
    for (let index = 0; index < listOfNames.length; index++) {
        let name = listOfNames[index];
        nodeToAppendTo.insertAdjacentHTML("afterend", "<p>"+ name +"</p>"
);
    }
}
```

Forms.js

Forms

- ➔ **What are forms used for**
  Allows a user to post data to a server

- ➔ **Validation is used to check the form before submission**
  Allows client side validation

Form Validation

```
function ValidatePassword(){
    let password = document.getElementById("password").value;
    let passwordLength = 6;
    let isEmailAddressInvalid = false;
    if (password.length < passwordLength) {
        isEmailAddressInvalid = true;
    }
    if(isEmailAddressInvalid){
        document.getElementById("validation").innerHTML = "<div class=\"message-body\"
>password is to short.</div>"
    }
}
```

Submit a form

```
let userObject ={
    email : 'user@test.com',
    password:'password'
}

function submitForm(){
    event.preventDefault()
    let password = document.getElementById("password").value;
    let email = document.getElementById("email").value;

    if(email === userObject.email && password === userObject.password){
        document.location = "Success.html";
    }
    else{
        document.getElementById("validation").innerHTML = "<div class=\"message-body\"
>username or password is incorect</div>"
    }
}
```

Prompt_Confirm.js

Prompt and Confirm

- ➔ **Prompt**

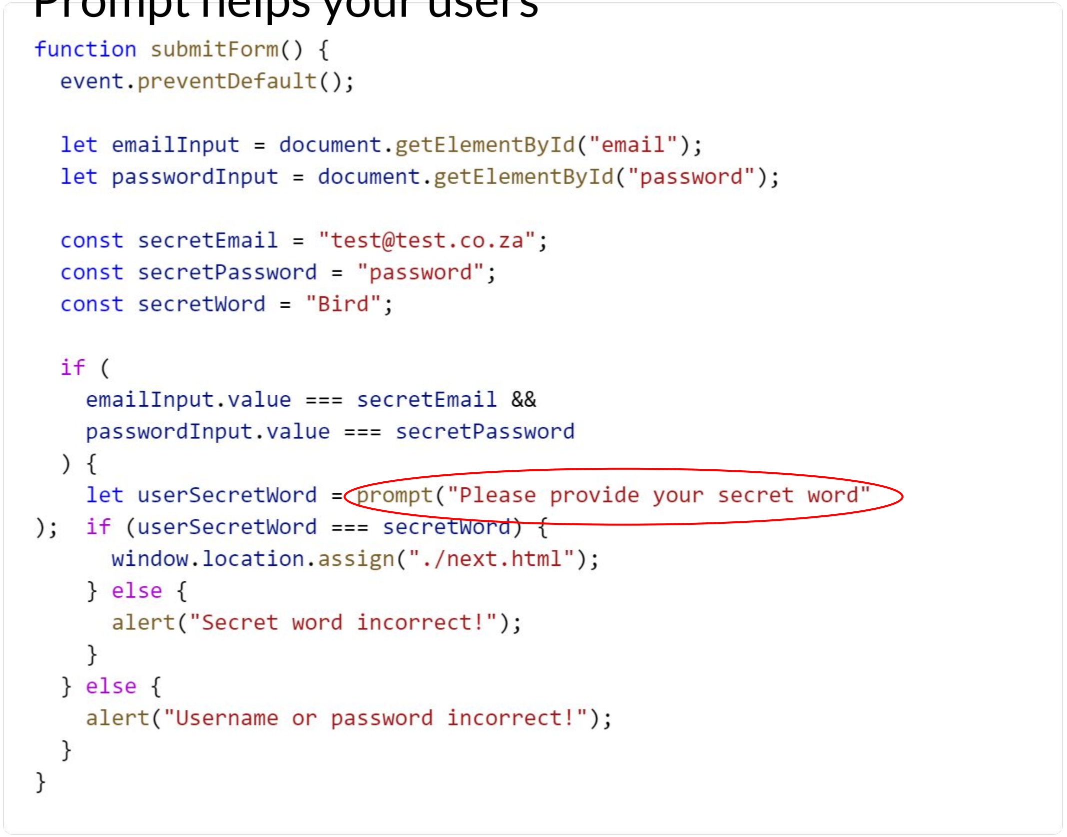
  Good for **prompting** users to input something small

- ➔ **Confirm**

  Good for getting **confirmation** from users

Prompt Exercise

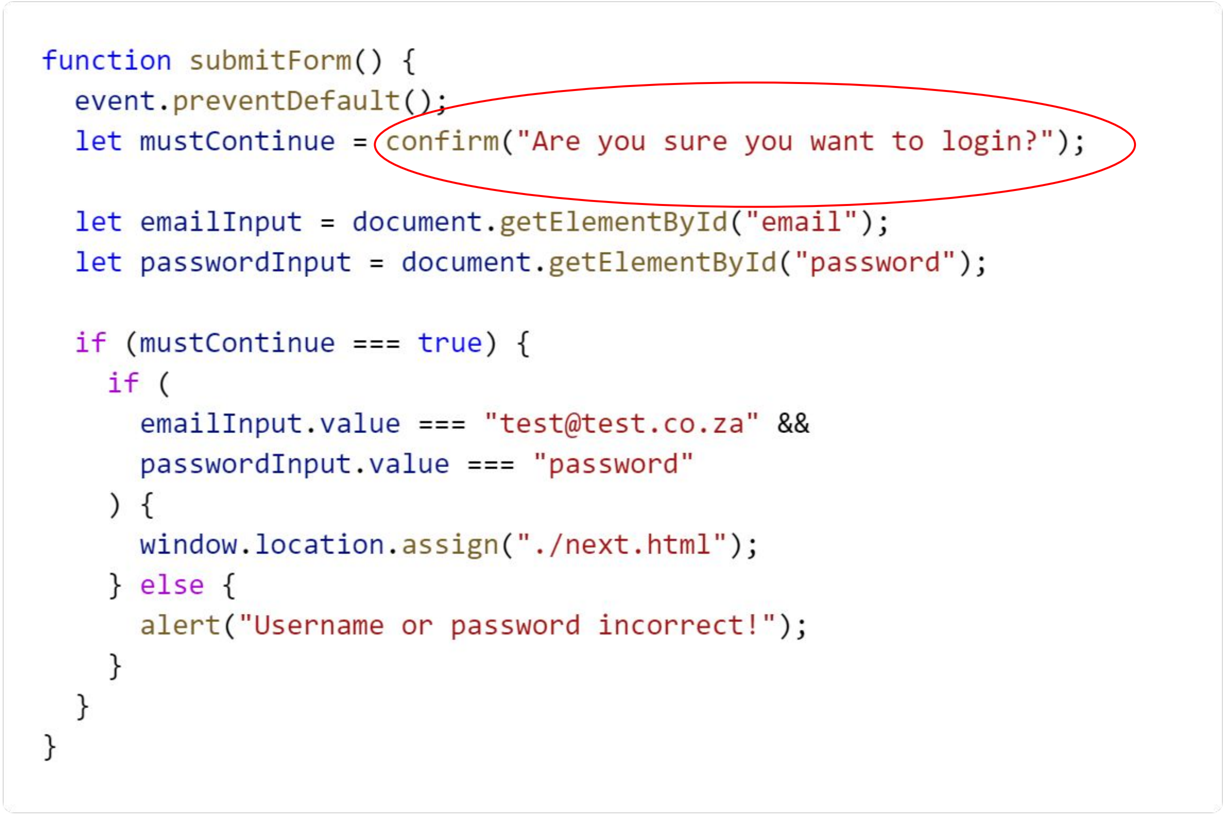
Prompt helps your users



```
function submitForm() {
  event.preventDefault();

  let emailInput = document.getElementById("email");
  let passwordInput = document.getElementById("password");

  const secretEmail = "test@test.co.za";
  const secretPassword = "password";
  const secretWord = "Bird";

  if (
    emailInput.value === secretEmail &&
    passwordInput.value === secretPassword
  ) {
    let userSecretWord = prompt("Please provide your secret word"
);  if (userSecretWord === secretWord) {
      window.location.assign("./next.html");
    } else {
      alert("Secret word incorrect!");
    }
  } else {
    alert("Username or password incorrect!");
  }
}
```

Confirm Exercise

Confirmation can be very useful



```
function submitForm() {
  event.preventDefault();
  let mustContinue = confirm("Are you sure you want to login?");

  let emailInput = document.getElementById("email");
  let passwordInput = document.getElementById("password");

  if (mustContinue === true) {
    if (
      emailInput.value === "test@test.co.za" &&
      passwordInput.value === "password"
    ) {
      window.location.assign("./next.html");
    } else {
      alert("Username or password incorrect!");
    }
  }
}
```

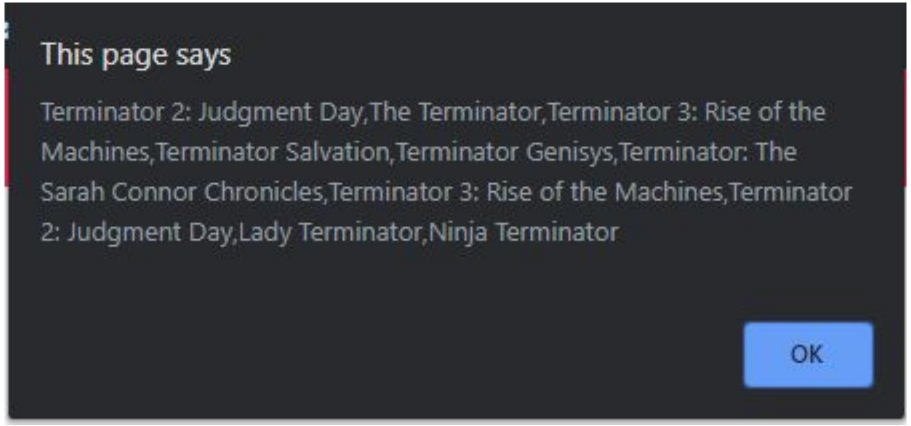
Let's
CODE
Something

Real World API Call

- **API**
  Application **P**rogramming **I**nterface - our application is going to be programmed to interface with theirs, this will require us to ask their application for some information
- **Fetch**
  We FETCH the information from their API. It's a special way of asking. Because we're working with something that won't be instant, we're going to have to wait for a bit.
- **Synchronous vs Asynchronous**
  Code that we've written so far is Synchronous, it's nearly instant. Things that we have to wait for are called Asynchronous.
- **Promises**
  In Asynchronous code, we get the data source to promise to return us some information later, and catch what happens if the promise doesn't work out.

API Call

Let’s get some data from somewhere online!



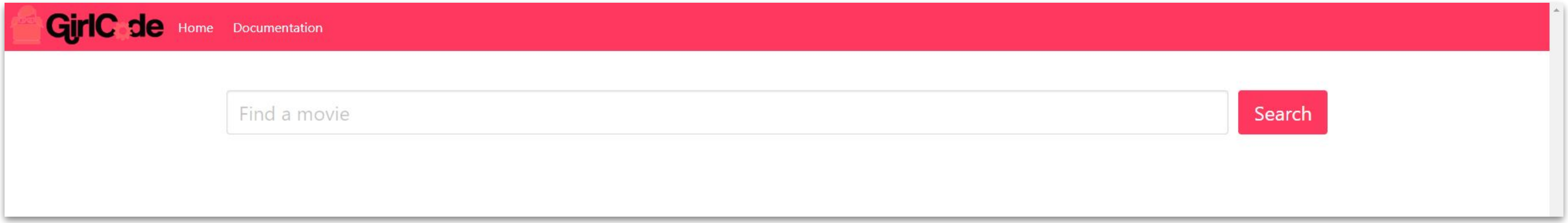
This page says

Terminator 2: Judgment Day,The Terminator,Terminator 3: Rise of the Machines,Terminator Salvation,Terminator Genisys,Terminator: The Sarah Connor Chronicles,Terminator 3: Rise of the Machines,Terminator 2: Judgment Day,Lady Terminator,Ninja Terminator

Let’s remix it to get another movie franchise!

API Call

We can do better!



Rules For Remixing

1. Share Your Passion
2. It's OKAY to make ~~misterks~~ mistakes
3. Enjoy Yourself

References

- https://javascript.info/function-basics
- https://javascript.info/array
- https://quizlet.com/236992008/mjs-array-methods-flash-cards/
- https://www.i-programmer.info/babbages-bag/263-stacks.html
- https://javascript.info/while-for#tasks

Test

<<Get this link>>

If you don’t have internet access, please let us know!

RESOURCE SLIDES AFTER THIS POINT

Links

- **Comparison Operators**
  - https://apprize.info/javascript/20lessons/4.html
  - https://www.w3schools.com/js/js_comparisons.asp