

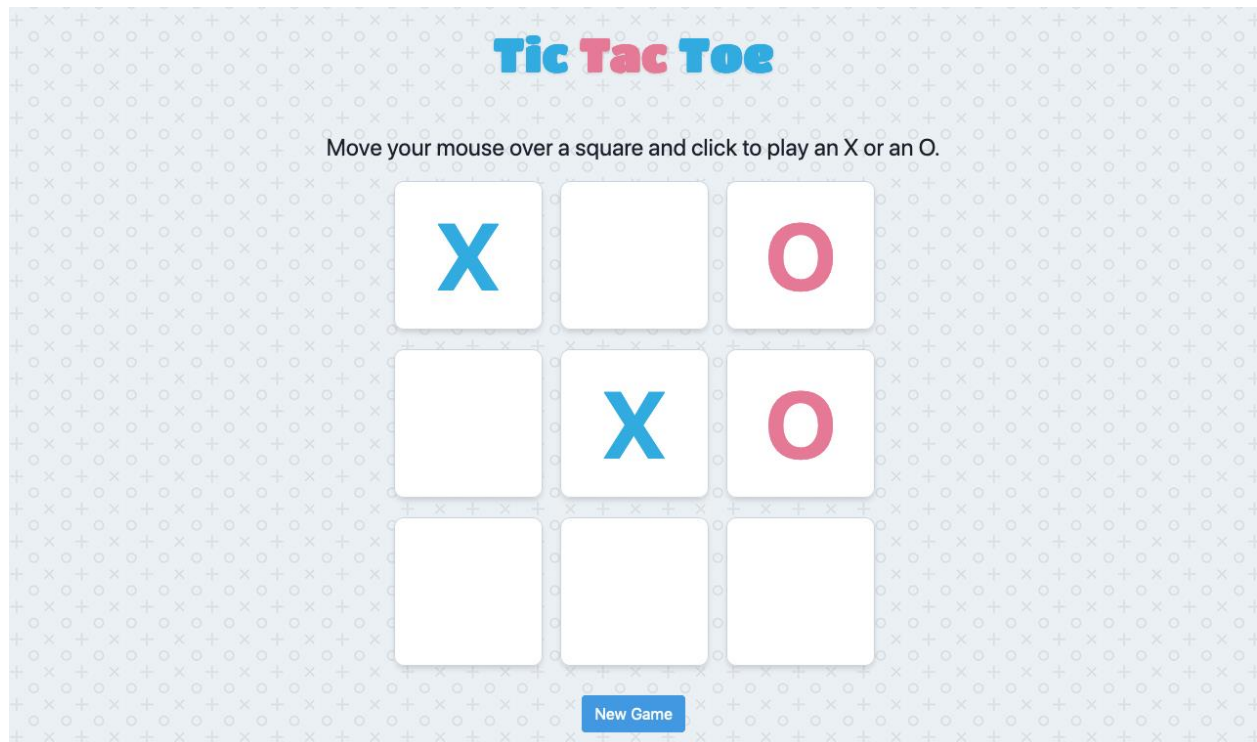
INFO2180 - Lab 3 (20 marks)

Due Date: **October 17, 2025 at 11:59 PM**

Tic-Tac-Toe

Tic-tac-toe is typically a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

In this lab we'll be using HTML, CSS and JavaScript to setup and play a simple game of Tic-



Tac-Toe in a web browser.

You're given files **index.html** and **tic-tac-toe.css**. You should get the starter code by downloading the files from the following link:

<https://github.com/uwi-info2180/tic-tac-toe/archive/master.zip>

It contains a HTML and CSS file, but these should **NOT** be modified. You should only write a script file **tic-tac-toe.js** in the same directory as these files that provides all the event handling and behavior to make the Tic Tac Toe game work as specified below.

Note: That the *Inspector*, *Console* and *Debugger* in the Developer Tools in your Web Browser can be very useful in helping you to debug your code in this Lab.

Create your Repository

You are also required to create a Git repository on Github called **info2180-lab3** and after each exercise commit your code and push to Github. So let us start by creating that repository.


1. You can create a new Github repository by going to <https://github.com/new> or by clicking the "+" icon in the top right once you are logged into the website and select "New Repository".
2. For this lab, use the repository name as **info2180-lab3**.
3. Ensure that your repository is *Public* and that you select the option to initialize the project with a README file.
4. The remaining options can be left at their defaults. Then click on **Create Repository**.

Search or jump to... / Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner ^{*} Repository name ^{*}

 uwi-info2180 / info2180-lab1 ✓

Great repository names are short and memorable. Need inspiration? How about fuzzy-octo-palm-tree?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

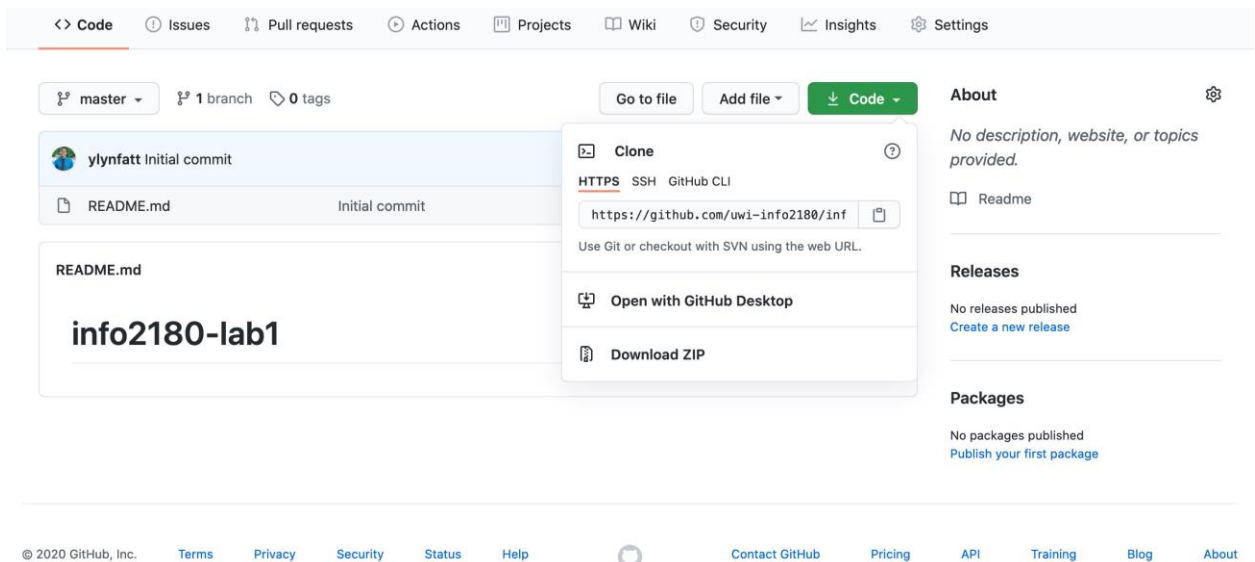
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set **master** as the default branch. Change the default name in uwi-info2180's [settings](#).

[Create repository](#)

[About GitHub](#) [Features](#) [Pricing](#) [Security](#) [Partners](#) [Mobile](#) [Product Hunt](#) [Press](#) [API](#) [Training](#) [Blog](#) [About](#)

5. On your newly created repository, click on the "Code" button and copy the URL in the box that appears.



6. Open your command prompt, navigate to the folder (e.g. using the cd command) where you would like to clone your repository and using the URL you copied in step 5, type the following:

```
$ git clone https://github.com/<your-username>/info2180-lab3.git
```

NOTE: Ensure you change the URL and use the one that you copied in Step 5 with your username.

7. Then navigate to your newly cloned repository by typing:

```
$ cd info2180-lab3
```

8. You can then type the command ls and you should see a README.md file.
9. In the file add:

```
# INFO2180 Lab 3
```

This is Lab 3 for <Your Name>

Of course, change <Your Name> to your actual name.

10. Save the file and then at the command line (or Windows Powershell), type:

```
$ git add README.md  
$ git status
```

You should then see that the file has been staged.

11. Commit these changes to your local Git repository:

```
$ git commit -m "Edited README.md"
```

12. Check the status of your Git repository:

```
$ git status
```

It should now mention that there is nothing new to commit and that the working tree is clean.

13. Great! Now we should push this information to GitHub.

```
$ git push
```

It may ask you to enter your Github username and password. Enter it and then press enter. If all goes well you can then view your Github repository on the Github website and you should see that the file exists on Github with the changes you made to the file.

14. To see the history of the commits you can run the following command:

```
$ git log
```

15. Next, Create a new branch called **game-implementation**.

```
$ git checkout -b game-implementation
```

16. Now download the starter files (if you have not already done so) for the lab at the following URL:

<https://github.com/uwi-info2180/tic-tac-toe/archive/master.zip>

17. Unzip and copy the starter files (index.html, tic-tac-toe-bg.png and tic-tac-toe.css) from the link above into your **info2180-lab3** folder that you cloned in Step 6 and do an initial commit.

```
$ git add .  
$ git commit -m "Added initial starter files"
```

18. Now begin the next set of exercises.

Exercise 1 - Layout the board

The first task is to create a new JavaScript file called **tic-tac-toe.js** (the `<script>` tag to reference this file is already included in the index.html file from your starter code) and in that file write the necessary JavaScript event-handling code so that when the page loads, each square in the 3x3 grid game board is styled appropriately by adding the right class. Write your JavaScript code unobtrusively, **without** modifying either the **index.html** or **tic-tac-toe.css** files.

Hint: It might be helpful to set each **div** inside the game board to have the provided CSS class **square**, using the JavaScript **classList** property or **setAttribute()** method or **className** property that you learnt in your lecture.

Hint 2: You may also want to ensure you use the **onload** or **DOMContentLoaded** event handler.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

Exercise 2 - Add an X or O to a square when clicked

For your next task, when a user clicks on a square in the grid it should alternate putting an **X** or an **O** onto the square that was clicked. Also ensure you add the **class "X"** or **"O"** to the square so that it is styled with the appropriate colour from the stylesheet. Take a look at the stylesheet to see what these classes do.

Hint: You may want to initialize an empty array to keep track of the state of the game after each square is clicked so that you can use it later to check which user has won. You can also use the

JavaScript **innerHTML** or **textContent** property to make the appropriate **X** or **O** show up in the appropriate **div**.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

Exercise 3 - Change the style when you move your mouse over a square

Now let us see if we can make it a little more interactive by changing the look of the square whenever a user moves their mouse over a square and then return it to the original style when the user's mouse leaves the square. You may notice in your **tic-tac-toe.css** file that there is a class called **hover**, which may be helpful.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

Exercise 4 - Check for the winner and update the status

When the user gets three X's or O's in a row they are declared the winner. Your next task is to check when either X or O has won and update the message on the page to say "***Congratulations! X is the Winner!***" or "***Congratulations! O is the Winner!***" depending on who wins. There is a **div** with an **id** of **status**, that this message should be placed in. You should also ensure that the class **you-won** is added to the status **div** as well.

Hint: You can use the JavaScript **innerHTML** or **textContent** property to help you to place the message in the **status div**.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

Exercise 5 - Restart the game

One annoying thing you may be noticing as you test the game so far is that it can't easily be reset to try again. So our next task will be to make it so that when the user clicks the New Game button, the game state will reset. That is, if the game squares have an X or an O then those are to be removed and the status message returns to the original message so that the user can try to play the game again.

Hint: You should use the **click** event handler.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

Exercise 6 - Disallow Cheating

Depending on how you coded the previous exercises, you may have a case where if a user clicks on a square that already has a value then they are able to change what was there before. Fix this by making it so that a user is not able to change the value of a square that already has an X or an O.

Once you have completed this exercise, take the opportunity to commit and push your code to Github.

When you are done and are sure everything is working, switch back to your master (or main) branch and then merge the changes you made in your "game-implementation" branch back into your master (or main) branch. Then ensure you push those changes back to your Github repository.

Submission

Submit this lab via OurVLE using the "Lab 3" Submission link. The following information must be submitted:

1. Locate your Github repository URL and Copy the URL. It should be in the format <https://www.github.com/your-user-name/info2180-lab3>. For example, if your username is **johndoe**, then the URL you will need to submit is <https://www.github.com/johndoe/info2180-lab3>.