CSYM019
Internet Programming

Dr Muawya Eldaw
muawya.eldaw@northampton.ac.uk

Week 2 - CSS

- What is CSS?
- How to attach a CSS file to a HTML document
- Targeting elements
- Example properties
- Useful resources
- Exercises
- Tracking completion of exercises though Github.

HTML

- A basic valid HTML file looks like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

HTML

- When you put something in an HTML tag, the browser interprets that tag to mean a specific thing
  - Headings, paragraphs, lists, etc
- The browser makes some assumptions about how these should look
  - E.g. headings are larger and bold, text is black, the page background is white, etc

CSS

- CSS can be used to override this behaviour and describe how these elements should look
- CSS stands for *cascading style sheets*
- CSS is a list of *rules* which are applied to elements on the webpage
- In english a sample rule would be:
  - *All headings should have red text*

CSS

- CSS code goes in its own file
- CSS code looks very different from HTML. You can tell a CSS file apart from an HTML file just by looking at the structure of the code
- You have to specify the CSS file to use in the HTML document using the *link* tag in the <head> section of the page

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="styles.css" />
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Name of the CSS file
(should have a .css extension*)

The “rel” (relationship) tells
The browser what type of
File is being referenced

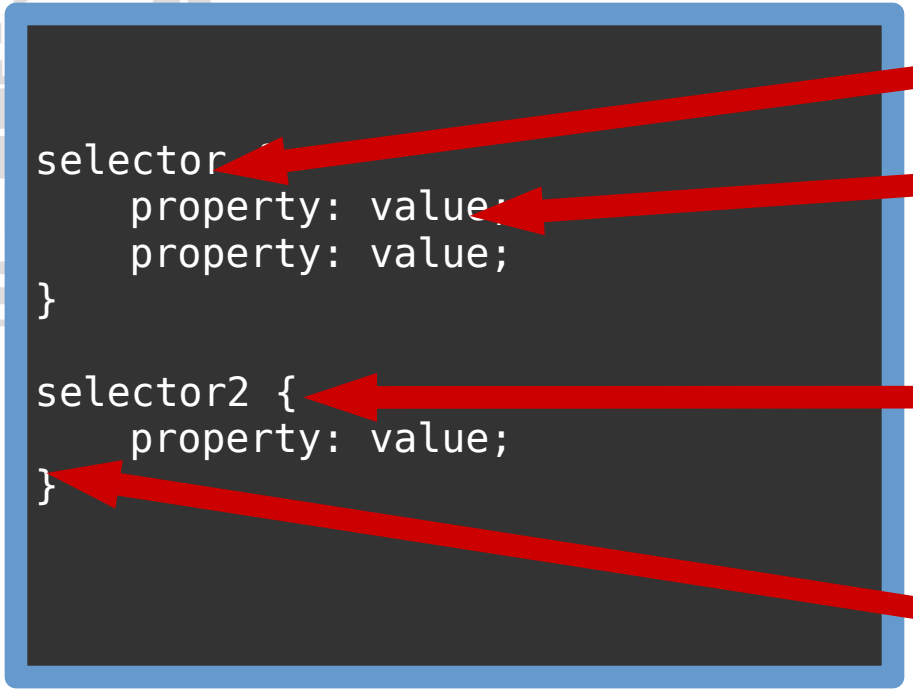
Link tag

CSS file

- A CSS file can be placed in any local or remote location that you can reference, but the simplest location to place it would be in the same folder/directory as the HTML file.
- ACSS file must have a .css extension.
- To create a CSS file, use your editor (Atom, VS Code, Sublime – see last week’s notes).

CSS code

- CSS code uses this format:



Selector is used to find one or more HTML elements

One or more properties are applied to the selected elements

The CSS file can contain as many blocks as you like
Each block is a selector and properties

Braces are important:
Every block must have an opening and closing brace!

CSS real example

- A selector can be a tag name
- One property is “color” (Note the American spelling!)
- To set H1 elements to red and paragraphs to blue:

```
h1 {
    color: red;
}

p {
    color: blue;
}
```

CSS properties

- When you target a tag it will affect any tag of that type:

```
h1 {
    color: red;
}

p {
    color: blue;
}
```

Page heading

Paragraph 1

Paragraph 2

```
<body>
  <h1>Page heading</h1>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</body>
```

Class name selector

- Any element can be given a *class* this is an *HTML attribute*
- The class name selector works in the format
  - .name
  - A dot followed by the name of the class you want to match
- This will match any element that has *class=”name”*
- **Note that the attribute does not include the dot in the HTML!**

Class name selector

- The class name selector is a name prefixed with a dot and selects any element with a class attribute set to that value

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="styles.css"
  </head>
  <body>
    <h1 class="myclass">Page heading</h1>
    <p class="myclass">Paragraph one</p>
    <p>Paragraph two</p>
  </body>
</html>
```

```
.myclass {
        color: green;
}
```

Notice that both elements with the class “myclass” are green

Class name selector

- The class name selector is useful whenever you want to be able to target specific elements on the page without affecting other elements by accident

ID Selector

- The ID selector works very similarly to the class name selector
  - Uses a # prefix
  - #myid
  - Matches any element with id=”myid” set as an attribute
- Note that the HTML does not contain the # symbol!
- IDs are different to class names. You can only use an ID once: Each ID should apply to a single element on the page

ID selector

- The ID name selector is a name prefixed with a hash and selects the element with that ID

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="st%ylesheet" href="demo.css
  </head>
  <body>
    <h1>Page heading</h1>
    <p id="myid">Paragraph one</p>
    <p>Paragraph two</p>
  </body>
</html>
```

```
#myid {
        color: red;
}
```

You may only have one element with each ID

Which should I use?

- In a lot of cases you can use a class name, or an element name or an ID to achieve exactly the same result!
- **You should avoid IDs** because you cannot easily reuse the style (if you want to make two elements red, you need to add two css rules and two IDs!)
- You should use element selectors when you want to affect all elements of that type
- If you want a specific style on a single part of the page, you should use class names
- **Don’t use IDs when you have multiple elements to style, use classes instead.**

Combining Selectors

- HTML is a *nested* notation
  - (Some) Elements can exist inside (some) other elements
- You can use css to target nested elements. By combining a selector with a space you can target specific elements in the HTML DOM Tree
  - **article h1 {font-weight: bold}** – Sets any <h1> element inside an <article> element to bold
  - **.myclass span {font-style: italic}** – Sets any element inside an element with *class=”myclass”* to italic
- This will target elements in any *depth!*

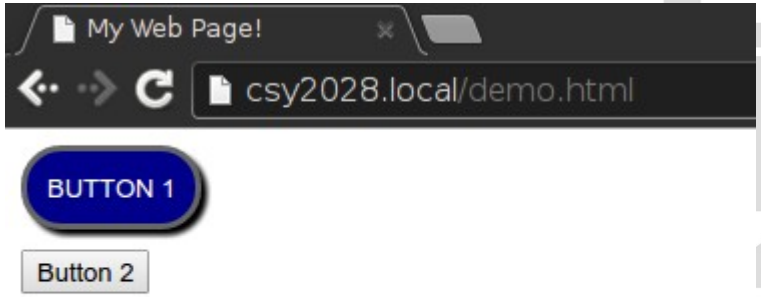
Combining selectors

- You can combine selectors to be more specific

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
      <input type="button" value="Button 1" />
    </div>

    <input type="button" value="Button 2" />
  </body>
</html>
```

```
.myform input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```



Notice that only the button inside the element with the class *myform* has been styled

Combining selectors

- This will target elements of any *depth*

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
       <input type="button" value="Button 1" />
      <section>
        <form>
          <input type="button" value="Button 2" />
        </form>
      </section>
    </div>

    <input type="button" value="Button 3" />
  </body>
</html>
```

```
.myform input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```

Notice that only the button inside the element with the class *myform*

Combining selectors

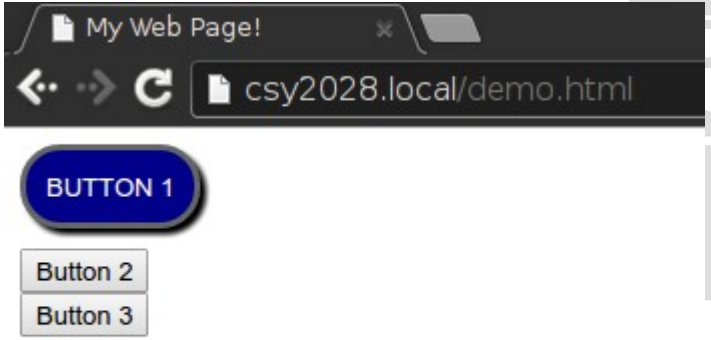
- You can use the *direct decedent* selector to filter to only elements that are only one level down
- The direct decedent selector uses the greater than symbol >

Combining selectors

- This will target elements only one level down

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
       <input type="button" value="Button 1" />
      <section>
        <form>
          <input type="button" value="Button 2" />
        </form>
      </section>
    </div>
    <input type="button" value="Button 3" />
  </body>
</html>
```

```
.myform > input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```



Notice that only the button inside the element with the class *myform*

Combining selectors

- You can combine selectors into very complex expressions e.g.:
- **`#myelement > .myclass article section header h1`**
- However, this is generally discouraged
- We call this *tightly coupled* to the HTML: A slight change to the HTML will stop the h1 tag being styled
- As a rule of thumb you shouldn't need more than three levels of selector

Block and inline elements

- Every element in HTML defaults to one of two *display* properties:
  - inline
  - block

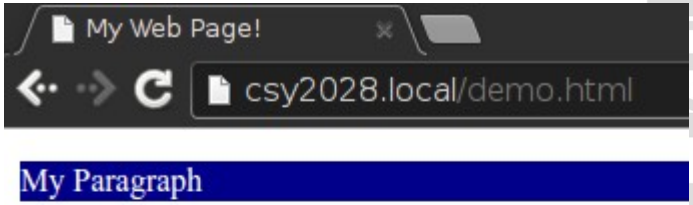
  These are treated very differently by the browser and determine how elements interact with one another.
- Block level elements extend all the way across the page and nothing can appear on the same line as them
- Inline elements are only as long as their content (e.g. the text inside them) and can appear on the same line

- By default:
  - Paragraphs, headings, lists, etc are *block* elements
  - Buttons, images, textboxes and elements are *inline*
- But this is only a default CSS can be used to override the display property in CSS can be set to inline or block to override the default behaviour

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>My Paragraph</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
}
```

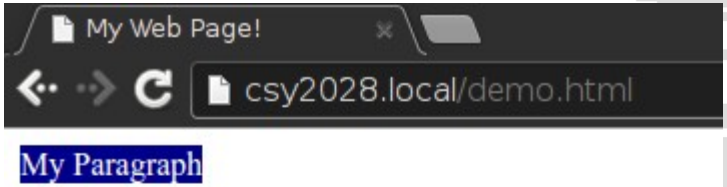


Notice that the blue background goes all the way across the page

Inline elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>My Paragraph</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
}
```



Notice that the blue background only extends to the width of the text

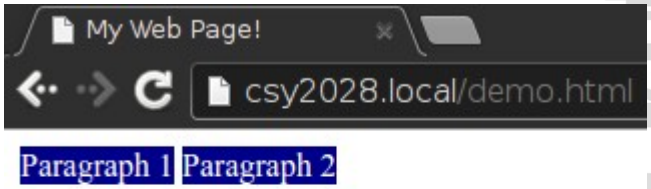
Block vs inline

- Block and inline affects how elements interact with each other
- Block elements will always be on their own line
- Inline elements will appear on the same line if there is enough space

Inline elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
}
```

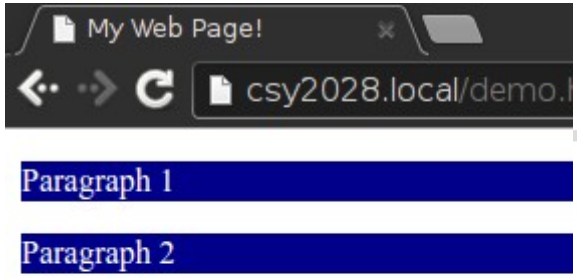


The elements appear on the same line

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
}
```



The elements appear on the same line

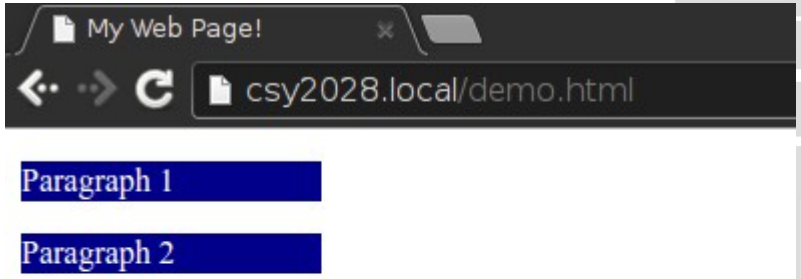
Block vs inline

- Block elements can have a width and height
- Setting a width and a height on inline elements will have no effect

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
}
```

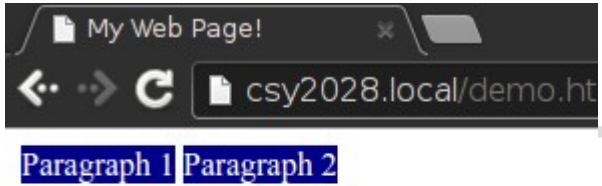


The elements appear on their own lines but now don't span the entire width

Inline elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
  width: 150px;
}
```



Width has no effect, they both stretch only as wide as the content

Float

- How do you put elements with a fixed with/height on the same line?
- *Float* can be applied to *block* elements and have them appear on the same line but with fixed dimensions
- Float has three options:
  - None (default, no float)
  - Left
  - Right

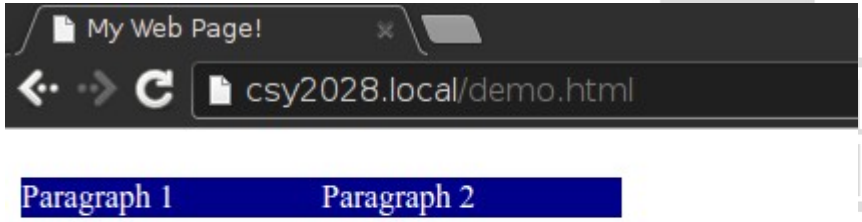
Float: left

- Float left will have the affect of stacking block elements left to right. When elements get too wide for the browser they will wrap to the next line

Float: left

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}
```



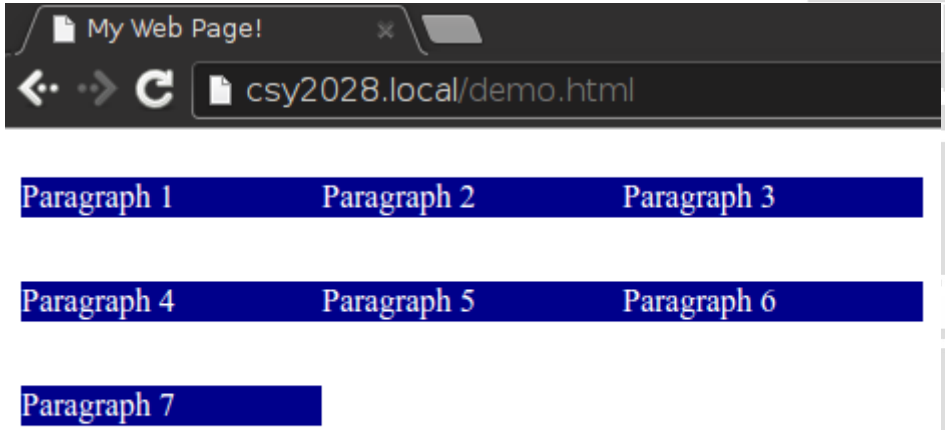
Float: left

- As more elements are added, they will be stacked left to right until there is no more room on the line

Float: left

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
  </body>
</html>
```

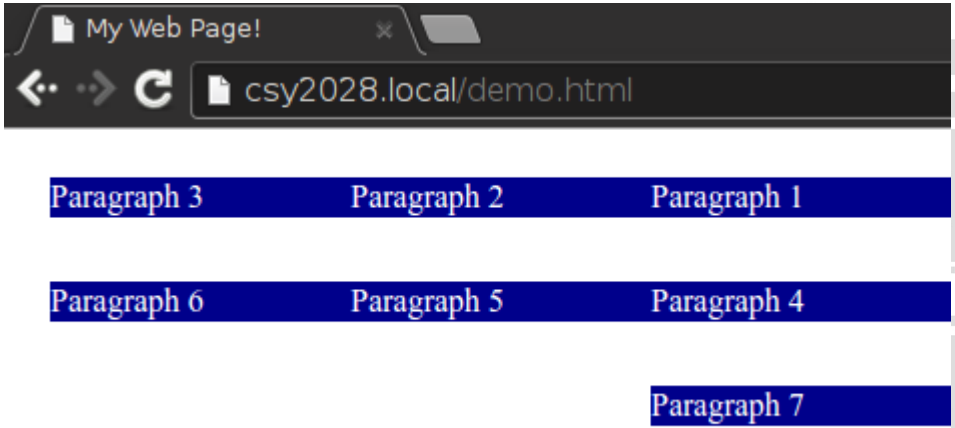
```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}
```



Float: right

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: right;
}
```

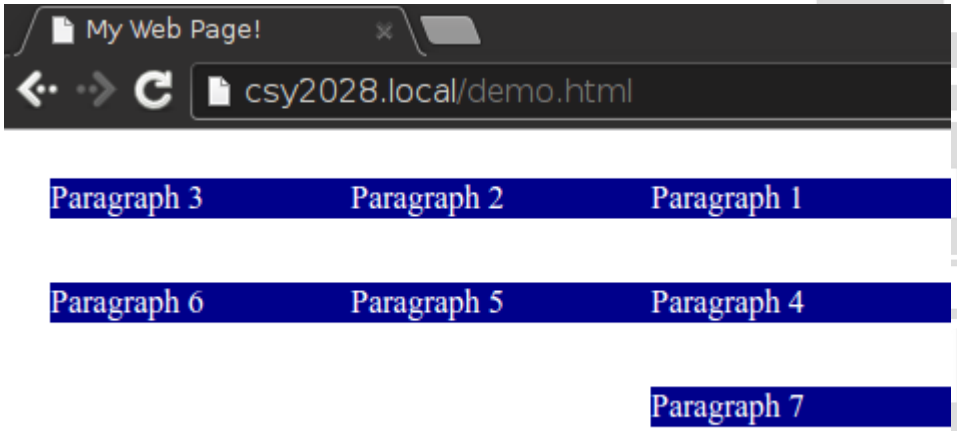


Float: right stacks the elements from right to left

Float: right

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
  </body>
</html>
```

```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: right;
}
```



Float: right stacks the elements from right to left

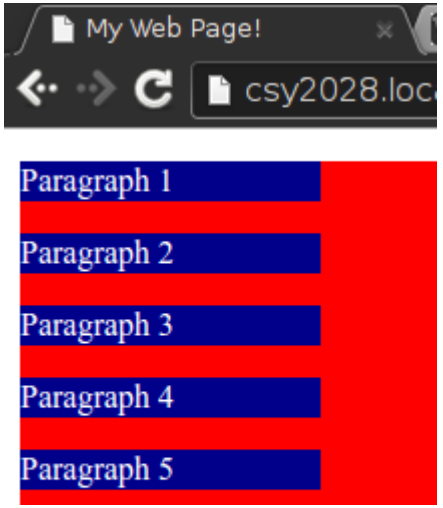
Float

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
    </div>
  </body>
</html>
```

```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;

}

.red {
  background-color: red;
}
```



As you expect, the paragraphs sit in a container with a red background

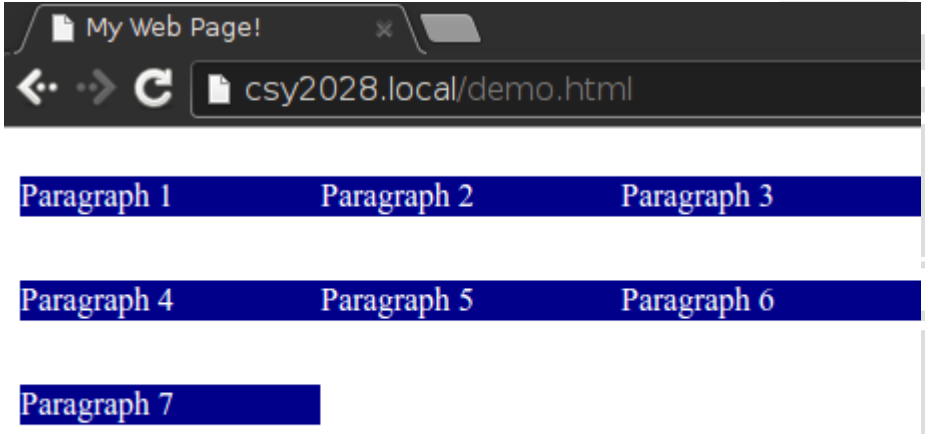
Float

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
      <p>Paragraph 1</p>
      <p>Paragraph 2</p>
      <p>Paragraph 3</p>
      <p>Paragraph 4</p>
      <p>Paragraph 5</p>
      <p>Paragraph 6</p>
      <p>Paragraph 7</p>
    </div>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;

}

.red {
  background-color: red;
}
```



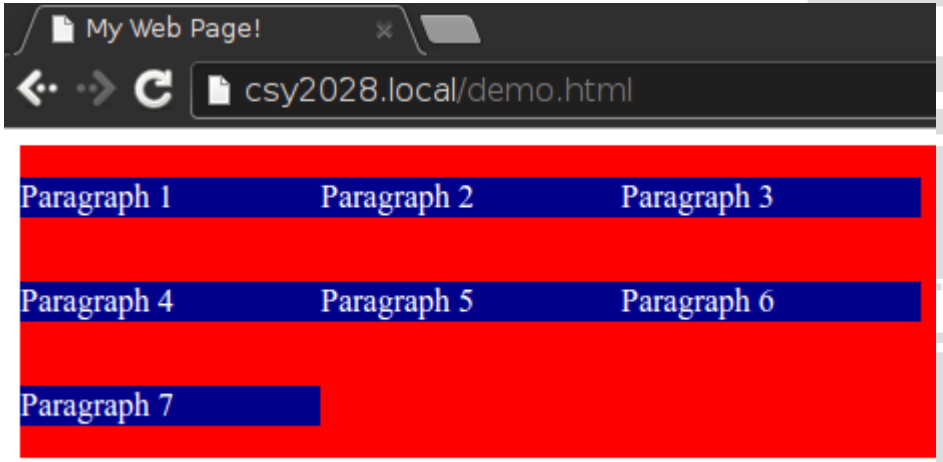
But when float: left is added,
The background disappears!

Float

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
      <p>Paragraph 1</p>
      <p>Paragraph 2</p>
      <p>Paragraph 3</p>
      <p>Paragraph 4</p>
      <p>Paragraph 5</p>
      <p>Paragraph 6</p>
      <p>Paragraph 7</p>
    </div>
  </body>
</html>
```

```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}

.red {
  background-color: red;
  overflow: auto;
}
```



By adding overflow: auto,
The background appears

Exercises

- 1. Download exercise1.html
  - Create the css file and create rules that make the various elements
  - **Hint: Be sure to look at the structure of the HTML!**
  - Necessary CSS properties:
    - Color (sets the colour of the font)
    - Background-color (sets an element’s background colour)

Exercise 2

1. Download Exercise2.html

   - Create the css file

   - Make sure that each module appears in its own column with its own background colour

CSYM019 - Internet Programming

Lecturer: Tom Butler

This module focuses on creating client and server side software as well as web applications for the World Wide Web us. It concentrates on the technologies used to allow such software to be designed implemented and deployed.

CSYM015 - Intelligent Systems

Lecturer: Scott Turner

To teach students the fundamental theory and practical applications of search methods and agents.The underpinning concepts will be introduced followed by examples of how intelligent systems are used in applications on the Internet.

Hint: You can use 50% widths on the Elements!

Exercise 3

- Research CSS and try to make your pages look nicer
  - Some things to start off:
    - Margins and Padding
    - Font-family
    - Border-radius
- Try creating a page with a grid of two columns that contains information about 4 different modules in a 2x2 grid

Exercise 4 (Difficult!)

- Try creating a page with 3 columns and a header and footer
  - Note: This is difficult, there are multiple solutions and we will cover this in detail next week

Introduction to version control with Git

- Version control software is an essential part of modern-day software developer practices.

- By far, *Git* is the most popular version control software in the world. It is an open source project originally developed by Linus Torvalds - the famous creator of the Linux operating system kernel.

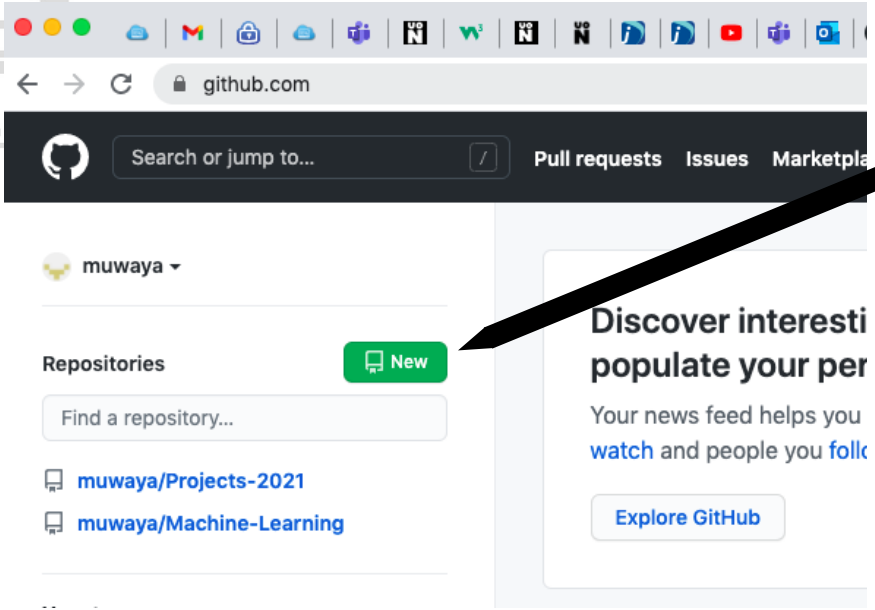
- Since July 2005, Git has been maintained by Junio Hamano who remains the project's core maintainer.

Git Installation

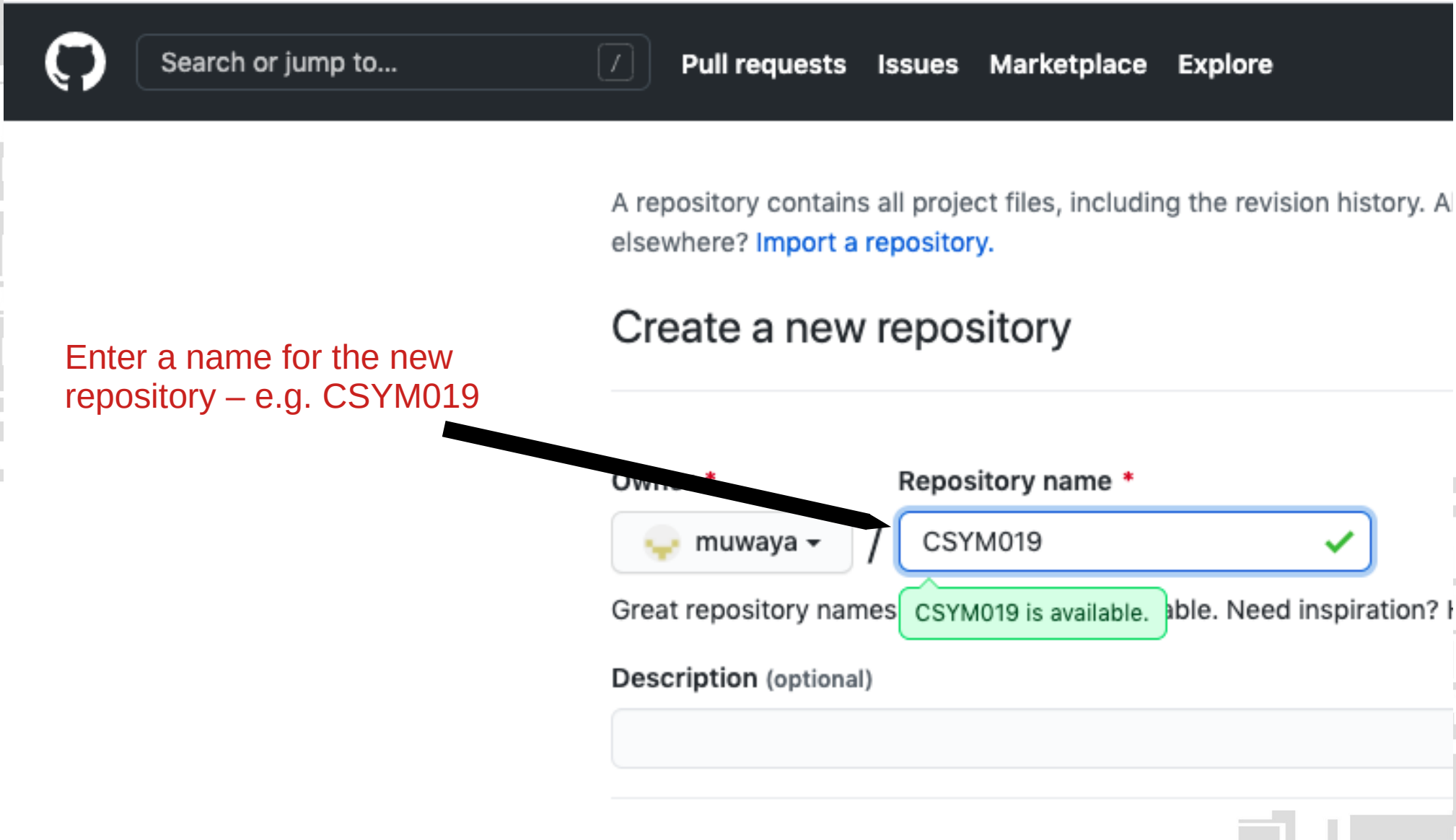
- Install Git from https://github.com/git-guides/install-git
- Make sure you use the correct installer for your Operating System (e.g. Windows, macOS, or Linux).
- Check that Git have been correctly installed – e.g. for Windows users,
  - open **Git Bash**
  - and type *git version* to verify Git was installed.

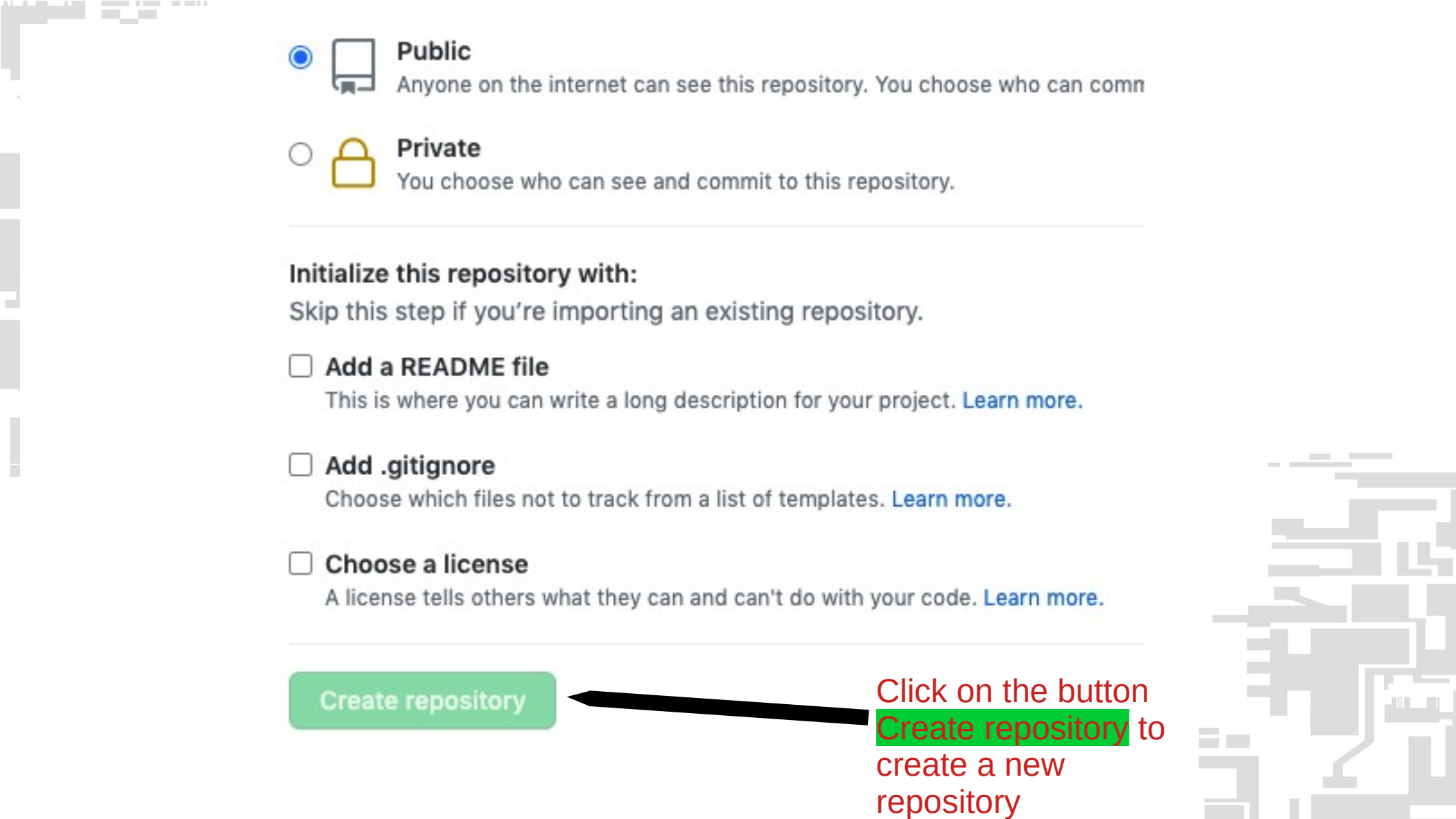
GitHub account

- If you **DO NOT** have a Github account, create one using the link https://github.com/
- Login into your account and Create a new repository.



Click on the button New to create a new repository


Search or jump to...
/
Pull requests
Issues
Marketplace
Explore
A repository contains all project files, including the revision history. A
elsewhere? Import a repository.
Create a new repository
Enter a name for the new repository – e.g. CSYM019
Repository name *
muwaya
/
CSYM019
CSYM019 is available.
Great repository names
able. Need inspiration?
Description (optional)


Public
Anyone on the internet can see this repository. You choose who can comn
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.
Create repository
Click on the button Create repository to create a new repository

Sync local repo with GitHub

- Enter the following commands in your terminal – e.g. Git Bash
  1) echo "# TestRepo" >> README.md
  2) git init
  3) git add README.md
  4) git commit -m "first commit"
  5) git branch -M main
  6) git remote add origin https://github.com/muwaya/TestRepo.git
  7) git push -u origin main

Using git to monitor completion of exercises

- It strongly recommended that you commit your exercise solutions to your repository on Github.
- Weekly exercises will be monitored via your Github repository.
  - Everyone should send the URL for the Github repository that they use for exercises.
  - **Write meaningful messages when committing work to local git repository.**
  - While completing the exercises, you should:
    1) add new files to local repository - (git add .)
    2) commit using informative messages – (git commit -m “include a meaningful message here”)
    3) push changes to remote repository on Github – (git push origin master)

- Pushing to Github should happens regularly when completing the exercises – every 10 minutes.

Why should you write good git commit messages?

- You might say, "It's just a personal project." Yes, you work alone now, but what happens when you work with a team or contribute to open source?

- A well-crafted Git commit message is the best way to communicate context about a change to other developers working on that project, and indeed, to your future self or to your tutor who is monitoring your work progress.

- Have you ever tried running git log on one of your old projects to see the "weird" commit messages you have used since its inception? It can be hard to understand why you made some changes in the past, and you'll wish you read this article earlier :).

- Commit messages can adequately communicate why a change was made, and understanding that makes development and collaboration more efficient.