IEEE Computer Chapter Southern Alberta Section

SCHULICH School of Engineering, UNIVERSITY OF CALGARY

IEEE Women in Engineering WIE Southern Alberta Section

SCHULICH IGNITE SCORCH

SESSION 0

With funding from Canada

Mindfuel™ Science Alberta Foundation

Disclaimer

We will be **recording** and **uploading** these **sessions** on **YouTube** so that you can access them from our **website** whenever you like.

We may also take **screenshots** during the sessions for **social media**.

If you unmute or turn on your camera during the session, you give us consent to use your video and audio.

- Breakout rooms are **not** recorded and we encourage you all to unmute and turn on your cameras in here!

Session overview

In this session, we'll be going over:

- More about Schulich Ignite
- Overview of the program content
- Places to contact us
- End of sessions Gala
- Getting your coding environment set up
- Games with your mentor!

Who are we

Schulich Ignite is a university student-run club aiming to make learning to code fun and accessible for all students.

Our goal is to **ignite** a passion for programming - so stick around!

Program Overview & goals

By the end of this course, we will:

- Be able to **create a website** from **scratch** or **template** and take it right through to **deployment**
- Understand **what is happening in a browser** when we visit a webpage
- Have an idea of some **industry practices** that will make us more effective in web development
- Have an idea of **other fields** web development skills can be used

Our approach

This course is a bit different than our other courses, on top of what you're used to we will also:

- Have exercises based off real-world projects/contracts
- Get you building multiple websites on various topics
- Cover some theory of how things work
- Cover real world tools people use in industry
- Set you up with everything you need to keep learning

Why learn web development

Universal Skills

Web technologies don’t only exist on the web, they are used in a ton of places including:

- Websites
- Car dashboards
- Operating systems
- Mobile apps
- Hardware integrations (printers)
- Game UI
- Web-based games

Practicality and Creativity

- There are **lots of jobs** in web development
- Build systems that are **easy & cheap/free to distribute**
- Endless options for appearance & layout
- Tons of **pre-built libraries** for anything you want
- Can be designed for any purpose
  - e.g games, messaging, marketing, tourism, event planning, education, analytics, etc

Content overview

Session 1: HTML, CSS, basic design

Session 2: Advanced CSS & design fundamentals

Session 3: Bootstrap & Javascript

Session 4: Advanced JS, Vendoring (Using other people’s code)

Session 5: Markup languages, templating & SSG's

Session 6: Basic networking & Deploying sites

Session 7: Intro to backend and API's

Session 8: Security

Content overview

Put more simply

Session 1-4: Basics, making sites look good, and developing some features

Session 5-8: Hosting/networking sites, working faster, developing more advanced features, and other things you should know

More About the course

We will teach the **technical** aspects, but we want to take it **further** than a glorified online tutorial. Things like:

- The **reasons why** we suggest what we suggest
- **Alternative** approaches to what we suggest
- **Resources** to look into topics more deeply
- Practice in **multiple contexts** (more than 1 site)
- Talk with mentors about their experience developing
- etc.

Multidisciplinary

This course will also go beyond the technical, and give you experience in **everything** you need to write websites, like:

- Design principles
- Content writing
- Case studies of real-world websites
- Security principles

Things you need to answer questions like…

Why is the right easier to read when it has more text?

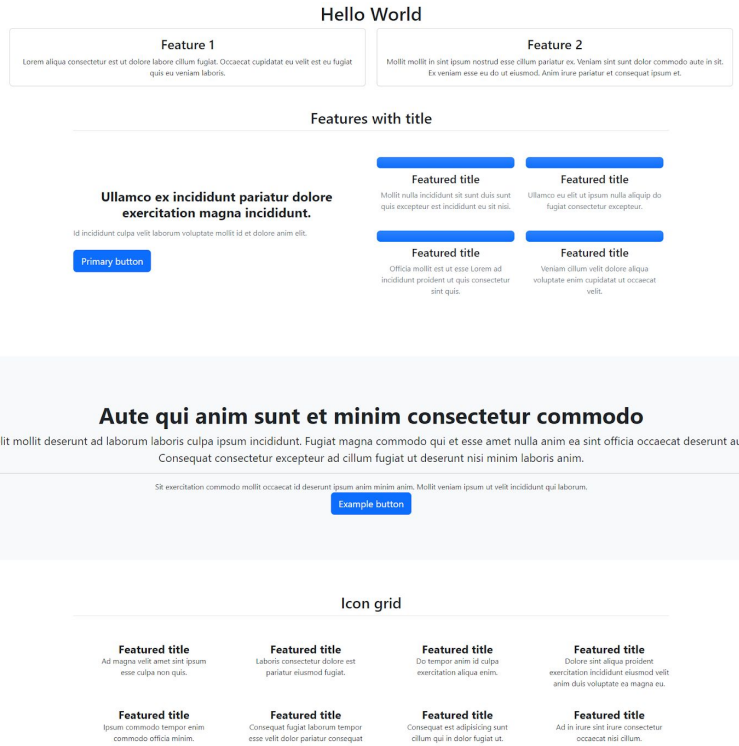
Hello World

Lorem aliqua consectetur est ut dolore labore cillum fugiat. Occaecat cupidatat eu velit est eu fugiat quis eu veniam laboris. Mollit mollit in sint ipsum nostrud esse cillum pariatur ex. Veniam sint sunt dolor commodo aute in sit. Ex veniam esse eu do ut eiusmod. Anim irure pariatur et consequat ipsum et.

Id incididunt culpa velit laborum voluptate mollit id et dolore anim elit. Ullamco ex incididunt pariatur dolore exercitation magna incididunt. Officia mollit est ut esse Lorem ad incididunt proident ut quis consectetur sint quis. Mollit nulla incididunt sit sunt duis sunt quis excepteur est incididunt eu sit nisi. Veniam cillum velit dolore aliqua voluptate enim cupidatat ut occaecat velit. Ullamco eu elit ut ipsum nulla aliquip do fugiat consectetur excepteur.

Velit mollit deserunt ad laborum laboris culpa ipsum incididunt. Fugiat magna commodo qui et esse amet nulla anim ea sint officia occaecat deserunt aute. Consequat consectetur excepteur ad cillum fugiat ut deserunt nisi minim laboris anim. Aute qui anim sunt et minim consectetur commodo. Sit exercitation commodo mollit occaecat id deserunt ipsum anim minim anim. Mollit veniam ipsum ut velit incididunt qui laborum.

Ad magna velit amet sint ipsum esse culpa non quis. Laboris consectetur dolore est pariatur eiusmod fugiat. Do tempor anim id culpa exercitation aliqua enim. Dolore sint aliqua proident exercitation incididunt eiusmod velit anim duis voluptate ea magna eu. Ipsum commodo tempor enim commodo officia minim.



How do I turn this site

What is Cross Site Scripting (XSS)?

Cross site scripting (XSS) is a type of web security vulnerability that allows an attacker to inject malicious code into a website. This code is executed by the victim's browser, and can be used to steal sensitive information, manipulate the website's content, or perform other malicious actions.

There are two main types of XSS: reflected and stored.

Reflected XSS

Reflected XSS occurs when an attacker injects malicious code into a URL that is then reflected back to the user's browser. Here's an example of how a reflected XSS attack might work:

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search?q=${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

With this code being a script that the victim's website runs as part of it's normal execution

```
// Victim's website
const searchTerm = new URL(window.location.href).searchParams.get('q');
document.write(searchTerm); // Used to display the search term on the site usually
```

In this example, the attacker's website creates a URL with malicious code injected into the query string. When the victim clicks on the link, the victim's browser sends a request to the victim's website with the malicious code in the query string. The victim's website then echoes the search term back to the browser, causing the malicious code to be executed.

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search/${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

With this code being a script that the victim's website runs as part of it's normal execution

```
// Victim's website
const searchTerm = new URL(window.location.href).pathname.split('/').pop();
document.write(searchTerm); // Used to display the search term usually
```

In this example, the attacker's website creates a URL with malicious code injected into the path parameter. When the victim clicks on the link, the victim's browser sends a request to the victim's website with the malicious code in the path parameter. The victim's website then echoes the search term back to the browser, causing the malicious code to be executed.

Stored XSS

Stored XSS occurs when an attacker injects malicious code into a website's database or other permanent storage, which is then executed whenever a user loads the affected page. Here's an example of how a stored XSS attack might work. Imagine you have a comment form on a news site that then displays the comments on the page when people read an article:

```
  <form id="comment-form">
<input type="text" name="name">
<input type="textarea" name="comment">
<button type="sumit">Submit</button>
  </form>
```

Into this

What is Cross Site Scripting (XSS)?

Cross site scripting (XSS) is a type of web security vulnerability that allows an attacker to inject malicious code into a website. This code is executed by the victim's browser, and can be used to steal sensitive information, manipulate the website's content, or perform other malicious actions.

There are two main types of XSS: reflected and stored.

Reflected XSS

Reflected XSS occurs when an attacker injects malicious code into a URL that is then reflected back to the user's browser. Here's an example of how a reflected XSS attack might work:

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search?q=${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

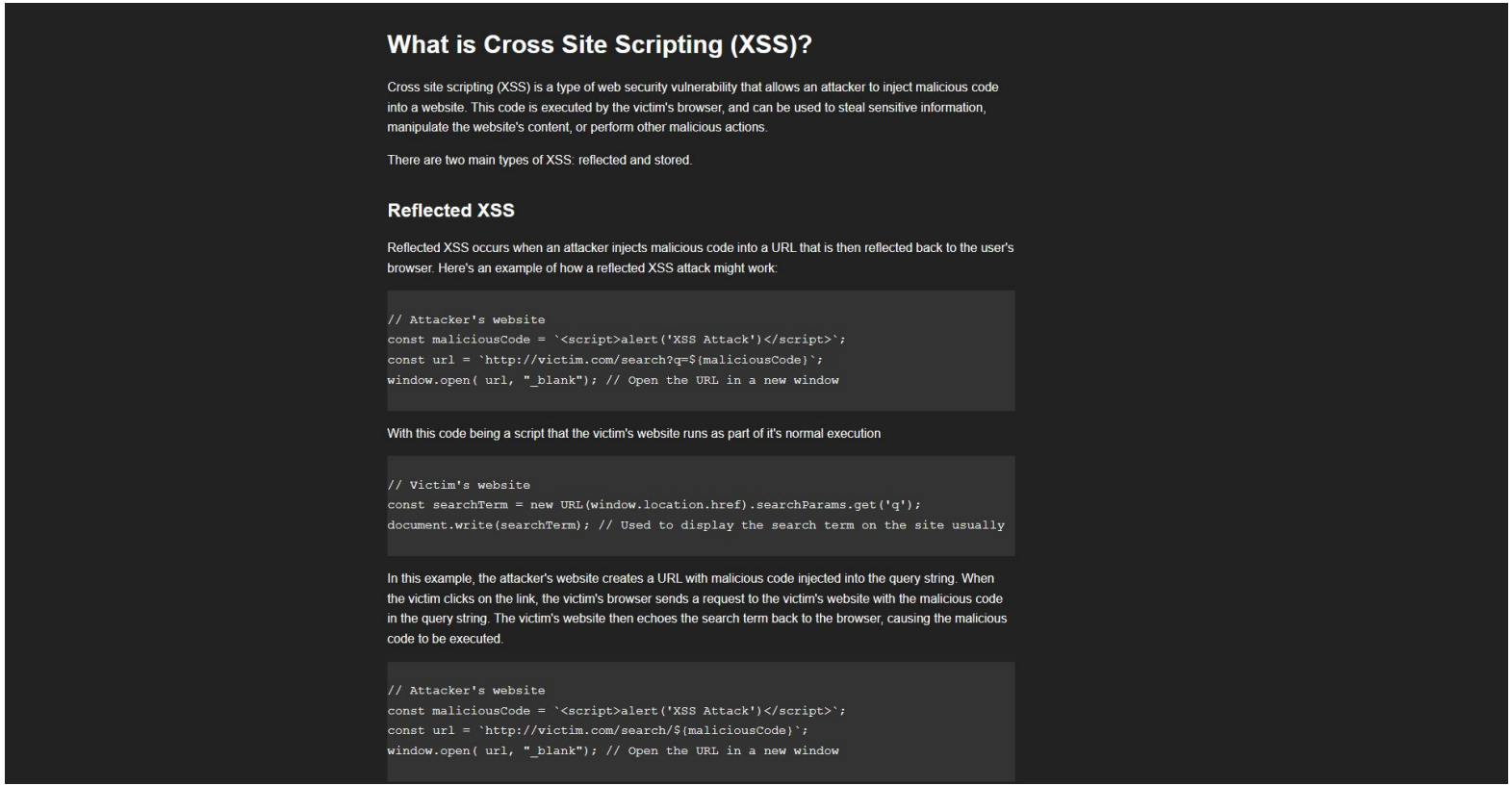
With this code being a script that the victim's website runs as part of it's normal execution

```
// Victim's website
const searchTerm = new URL(window.location.href).searchParams.get('q');
document.write(searchTerm); // Used to display the search term on the site usually
```

In this example, the attacker's website creates a URL with malicious code injected into the query string. When the victim clicks on the link, the victim's browser sends a request to the victim's website with the malicious code in the query string. The victim's website then echoes the search term back to the browser, causing the malicious code to be executed.

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search/${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

Or this



What is Cross Site Scripting (XSS)?

Cross site scripting (XSS) is a type of web security vulnerability that allows an attacker to inject malicious code into a website. This code is executed by the victim's browser, and can be used to steal sensitive information, manipulate the website's content, or perform other malicious actions.

There are two main types of XSS: reflected and stored.

Reflected XSS

Reflected XSS occurs when an attacker injects malicious code into a URL that is then reflected back to the user's browser. Here's an example of how a reflected XSS attack might work:

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search?q=${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

With this code being a script that the victim's website runs as part of it's normal execution

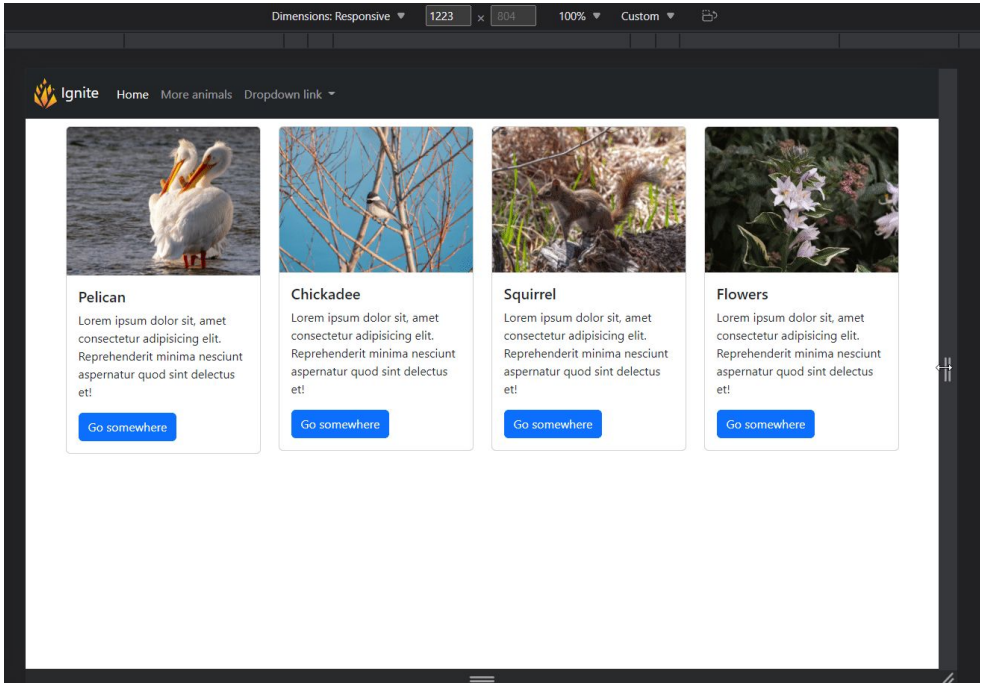
```
// Victim's website
const searchTerm = new URL(window.location.href).searchParams.get('q');
document.write(searchTerm); // Used to display the search term on the site usually
```

In this example, the attacker's website creates a URL with malicious code injected into the query string. When the victim clicks on the link, the victim's browser sends a request to the victim's website with the malicious code in the query string. The victim's website then echoes the search term back to the browser, causing the malicious code to be executed.

```
// Attacker's website
const maliciousCode = `<script>alert('XSS Attack')</script>`;
const url = `http://victim.com/search/${maliciousCode}`;
window.open( url, "_blank"); // Open the URL in a new window
```

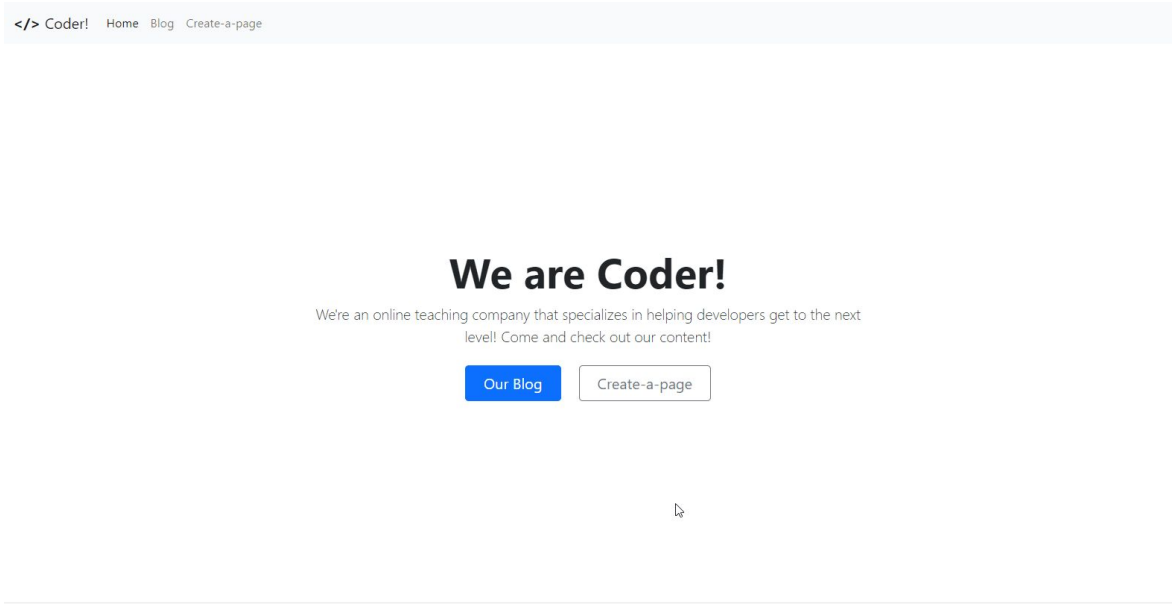
Sneak peek at what we're going to build Throughout

By the end of session 3

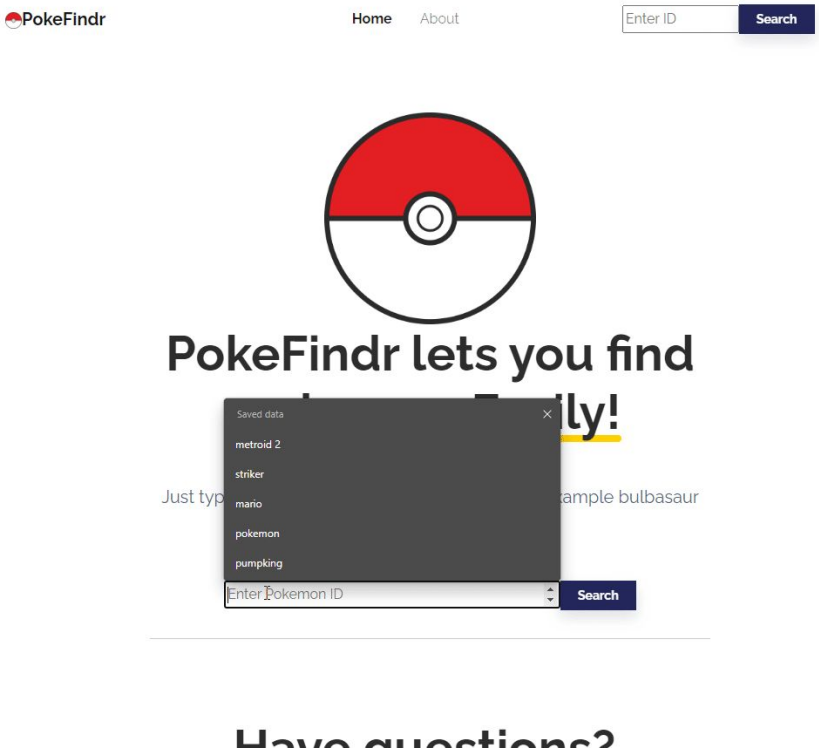


Sneak peek at what we're going to build Throughout

By the end of session 4



By the last session!



Some notes about the slides

The slides aren’t all meant to be read **entirely**. They’re also there for reference **so you can go back** to them for more details later.

I will skip over them without reading every line.

It’s better to pay attention to what’s being said than trying to just read!

Notes about slides cont'd

We have colour coded slides to make it simpler to understand. Ones with no colour like this are “main slides” we will cover.

Resources

Slides with green headers have resources, this is stuff you can check out on your own time like:

- Libraries and software that does what we’re talking about
- Extra concepts we don’t have time to cover
- Resources that go more in depth if you’re interested

Notes about slides cont'd

Quick Questions

These are multiple choice questions that you will have time to answer.

Exercise

These will be when the coding begins!

End of session exercise

These are longer form exercises that are optional and can be completed for ignite points (explained later)

End of session exercises

The end of session exercises are there for you to:

- Test skills you've learned at 3 levels of difficulty
- Work with similar requirements you might see in the real world
- Get to build sites with "real" content (not just filler)

This is entirely optional!

You can also submit them to be "marked" for **ignite points**:

- The webpage meets the requirements asked
- The content was used correctly
- If someone asked you to do it they would be "happy"

Ignite points

Points that can be collected to win prizes! Can be collected through:

- Consistent attendance (Streak Score)
- Answering Quick Questions
- End of Session Exercises

Winners and prizes

The **top 3** students with the most **Ignite Points** gets Amazon eGift Card (Amount tentative)!

STREAK SCORE

- You will get rewarded for consistent attendance!
- For your first attended session, you get 5 points, but with each consecutive session you attend, you get 5 more points than you did in the last session!
- Example on the next slide

Streak score example

Anna is a mentee who attended all the 8 weekly sessions **except for session 6.**

Anna gets 5 points on session 1, 10 on session 2 and so on until 25 on session 5.

She gets 0 points for not attending session 6 and her **streak score resets.**

She gets 5 points for session 7 , and 10 points for session 8.

Session	Points Per Session
1	5
2	10
3	15
4	20
5	25
6	0
7	5
8	10

Quick QUESTIONS

Slides with this icon have questions that can be answered for points!

- Worth **10-20** points each
- Multiple choice answers
- Can be answered through TopHat

Top Hat Check!

Please make sure you have a way to get on Top Hat either via the web page or download the app.

Create your account name using your first and last name, and please use the same email as your Zoom account

Join the course using code:
789689

Think you are ready? Please go ahead and answer the questions on Top Hat!

End of session exercises

- A challenge at the end of each week that changes to match the topic
- 3 Levels, what you need to do for each level changes each week
  - **Level 1:** 15 Points
  - **Level 2:** 25 Points
  - **Level 3:** 35 points

Our website

Website: https://schulichignite.com/

- Here’s where you can see all the slides for all of our sessions & recordings
- Easy access to all important links

Discord

Join us on **Discord**!

- Dedicated channel for your group
- Ask your mentor questions outside of class time
- Show off any cool projects!

https://discord.gg/SDDdcZ9

Discord - Sign Up For Private Group Channel

1. Check your **email** for your **day of week** and **group number**
2. Then on Discord, in the **#bot-commands** channel, type in:

    !squad set <day>-<group number>

Example:

!squad set monday-3

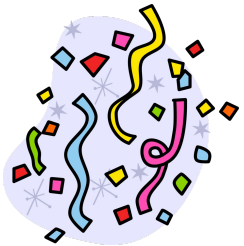
Discord - Sign up for private group channel

Note: EmberBot is not yet online, so please join the server for now, and we will update you when the bot is up and running!

End of sessions gala

We will be having a **Gala** at the **end** of the 8 weeks to celebrate your completion of Schulich Ignite!

- Opportunity to show off your final project
- Learn more about software engineering from industry professionals
  - In the past we have had software engineers and managers from Google, Microsoft, and AMD attending and speaking about their experience!



Let's Get Started!

Getting Started - Browser

You will need a **Chromium-based** browser. The easiest one to use is google chrome, microsoft edge will also work.

If you typically use firefox, safari, or other browsers you will need to have one of the above browsers installed to use for these sessions.

You can find how to download and install chrome at the link here

Getting Started - Python

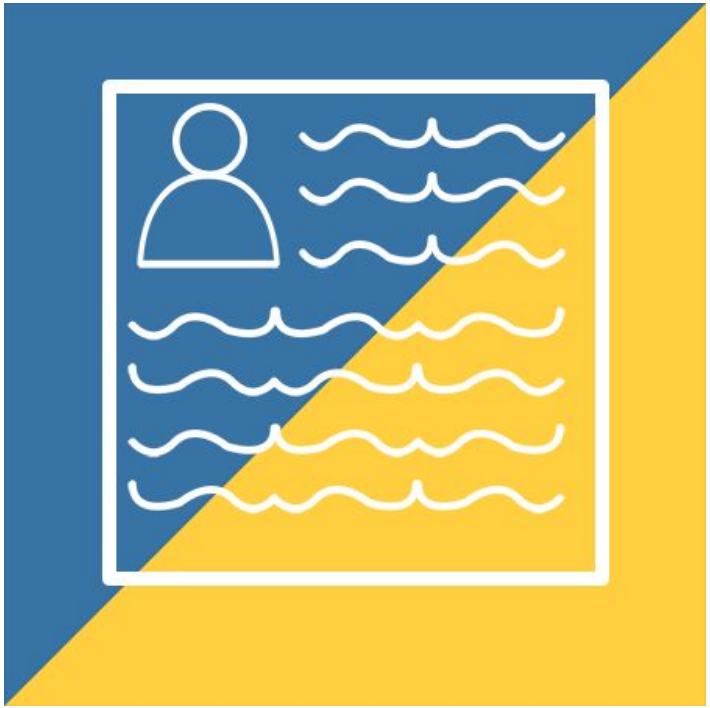
You will need to setup python 3 (version 3.7+) and pip.

Start by downloading python here: https://www.python.org/downloads/

From there we have guides for installing on each operating system:

Windows: Guide
MacOS: Guide
Linux: Debian Based guide(you will need to find a guide for your distro if it's not debian based)

Getting Started - ezcv



Now that we have python you will need to install a package we will need later in the course. To do this open a terminal and run the following commands based on your operating system:

Windows: pip install ezcv

Macos/linux: pip3 install ezcv

Getting Started - VS Code

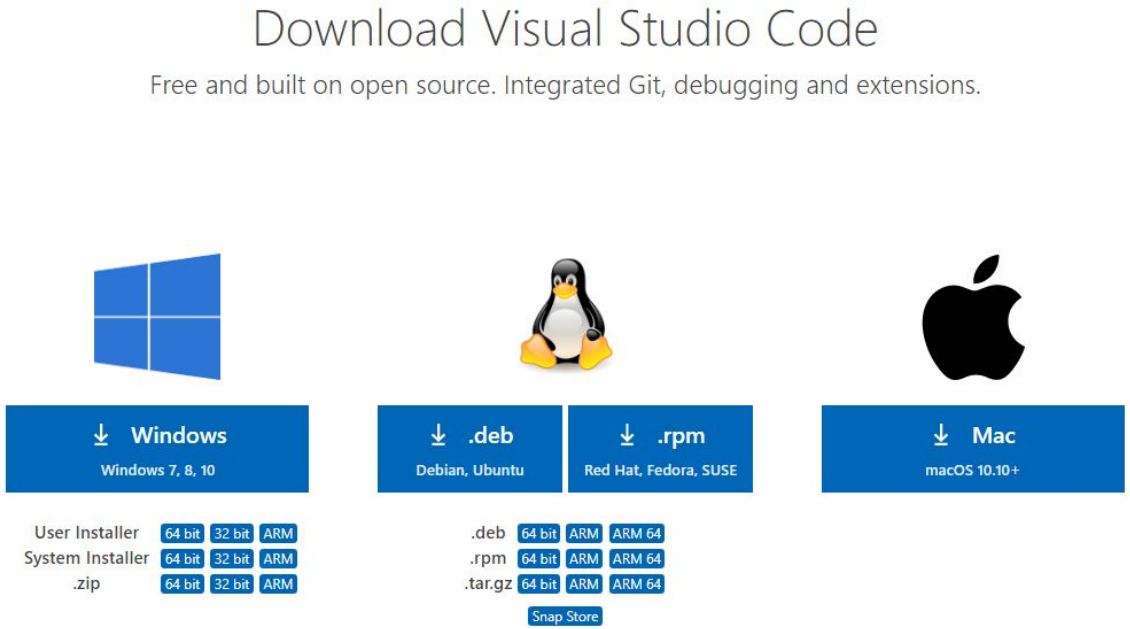
A popular and lightweight IDE (programming tool)

Strongly recommend using **VS Code** as our lecture lead and mentors will be using it

Installing VS Code

Go to code.visualstudio.com and click download.

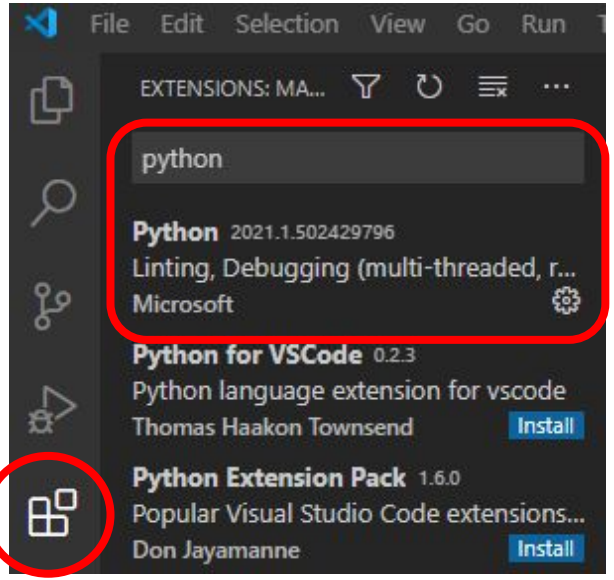
Select the download for your operating system.



Setup

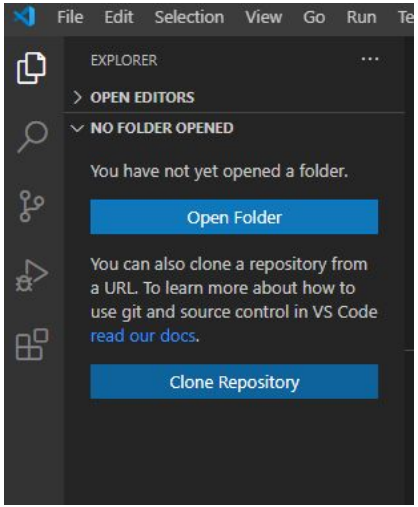
Follow the setup instructions.

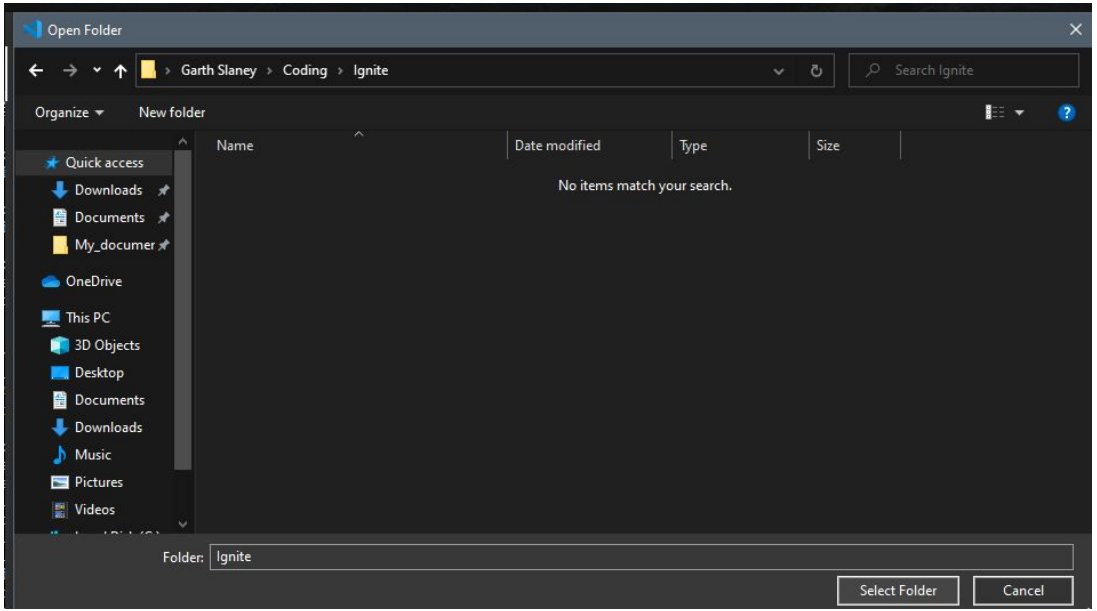
Navigate to extensions and install Python from Microsoft



Make a folder

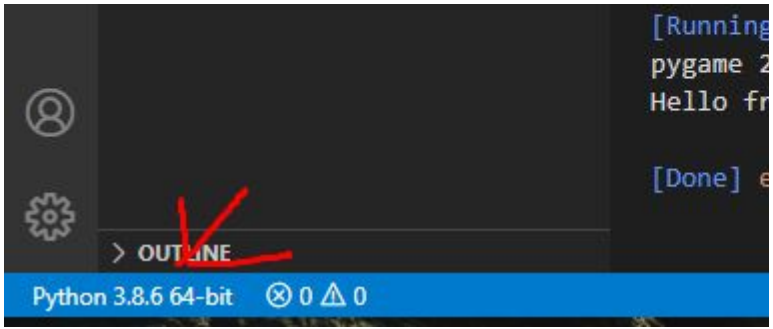
Click on open a folder and navigate to an empty folder on your computer.





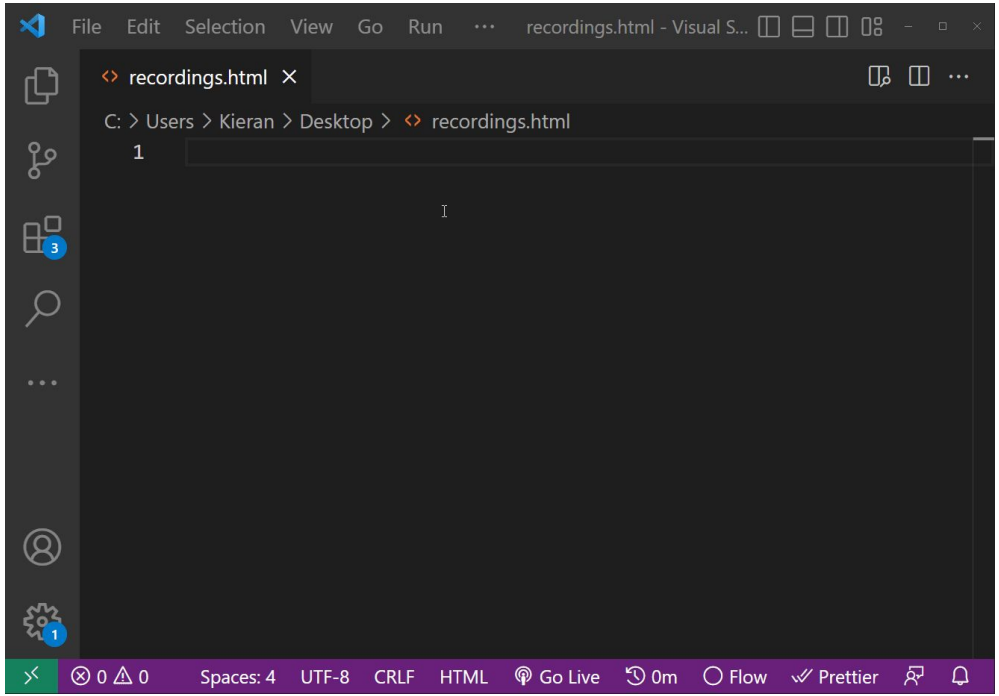
Using VS Code

- **Save** your changes before you run! (CTRL + S or CMD + S)
- Any file added to the directory will appear in your VS code editor
- Only **.py** files will run
- You can change your Python version by clicking it in the bottom left



Memorizing code is bad

We will be introducing various extensions for VS code as we go, and use the **emmet** system, this will basically let you type in short commands to get longer snippets of code!



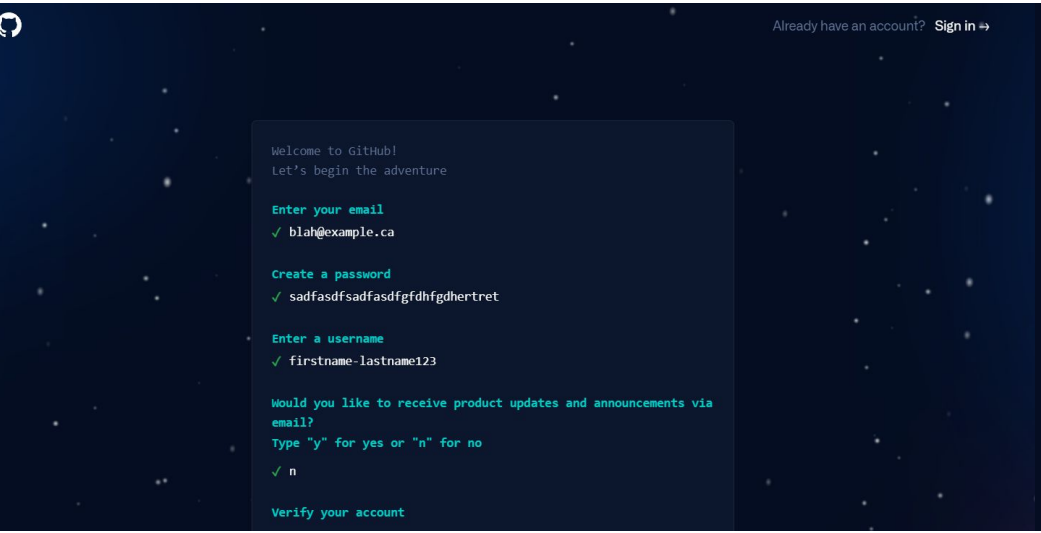
Getting Started - Github

Github is the system we are going to be using to provide the exercise files & using to deploy our sites later on.

If you are interested you can learn more about git and github (though it's not necessary) here: https://www.youtube.com/watch?v=NwASRGFz5Wg

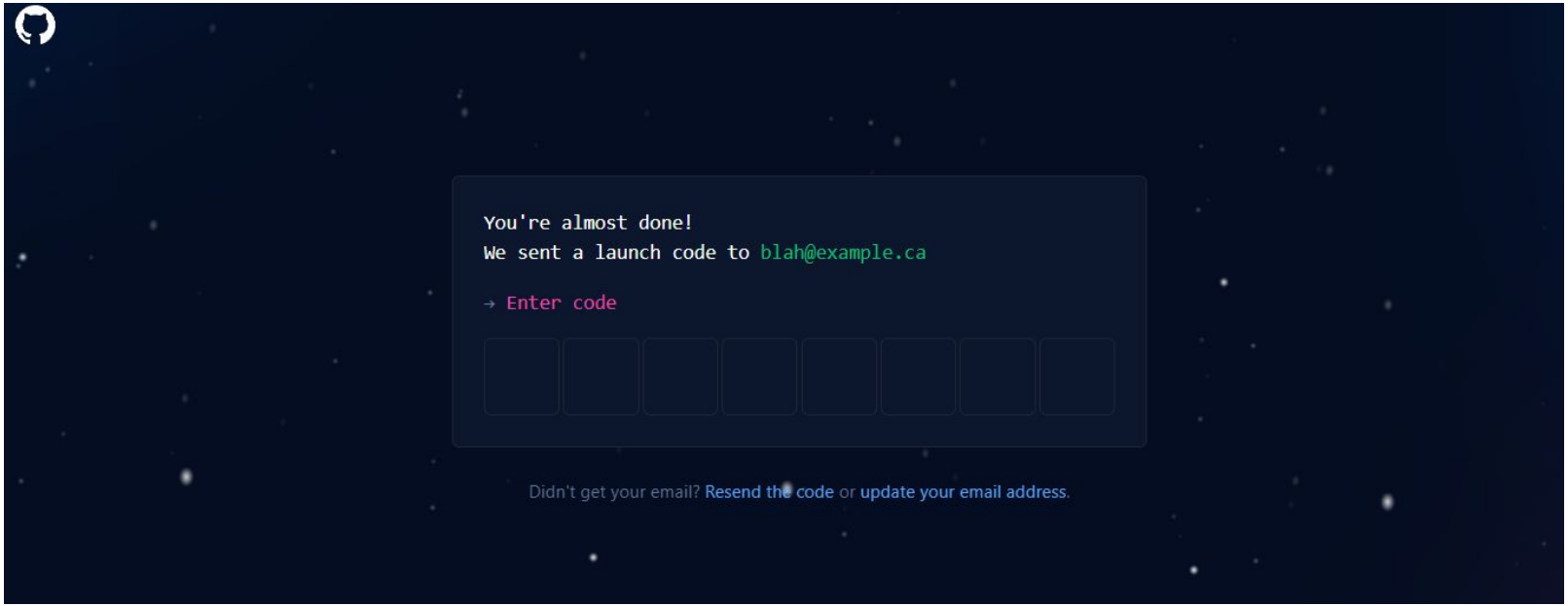
Making a Github Account

1. Head to https://github.com/signup
2. Enter your email, password, a username and answer if you want to receive emails from them
3. Complete the “I’m not a robot quiz” and hit create account



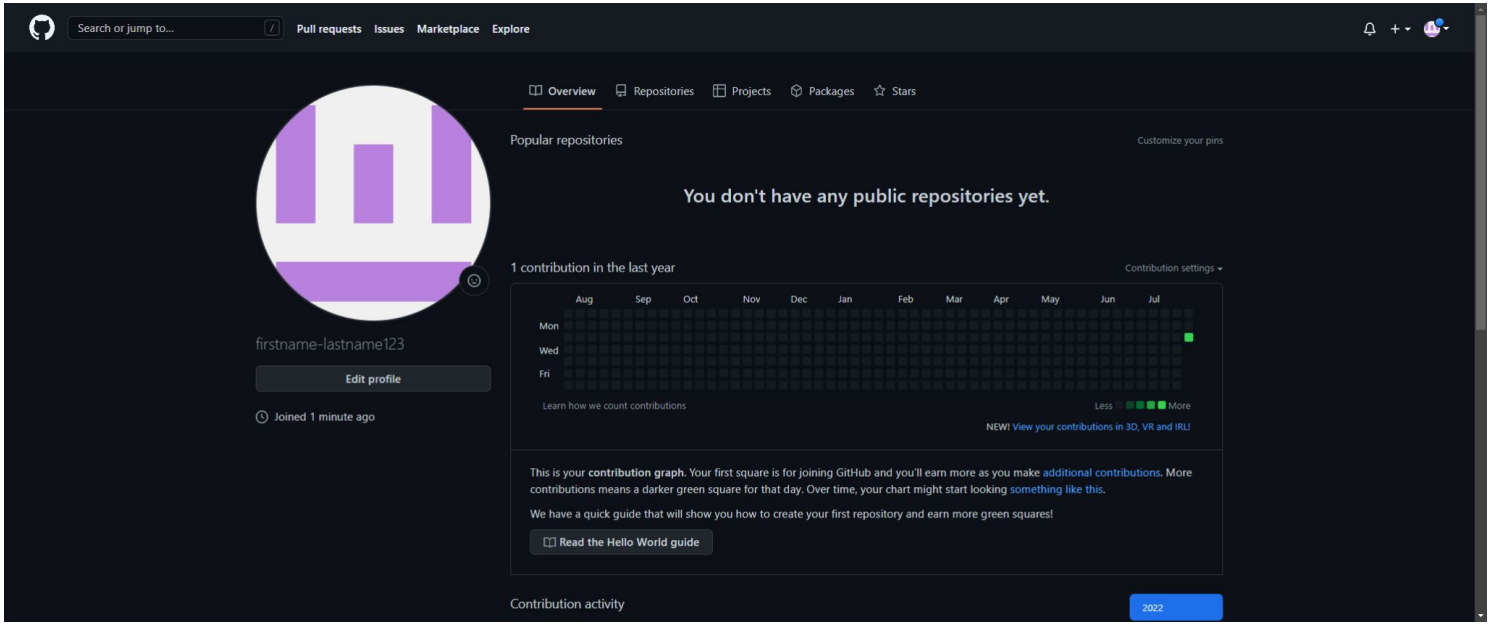
Making a Github Account

4. Enter the code that gets sent to your email to finalize setting up your account



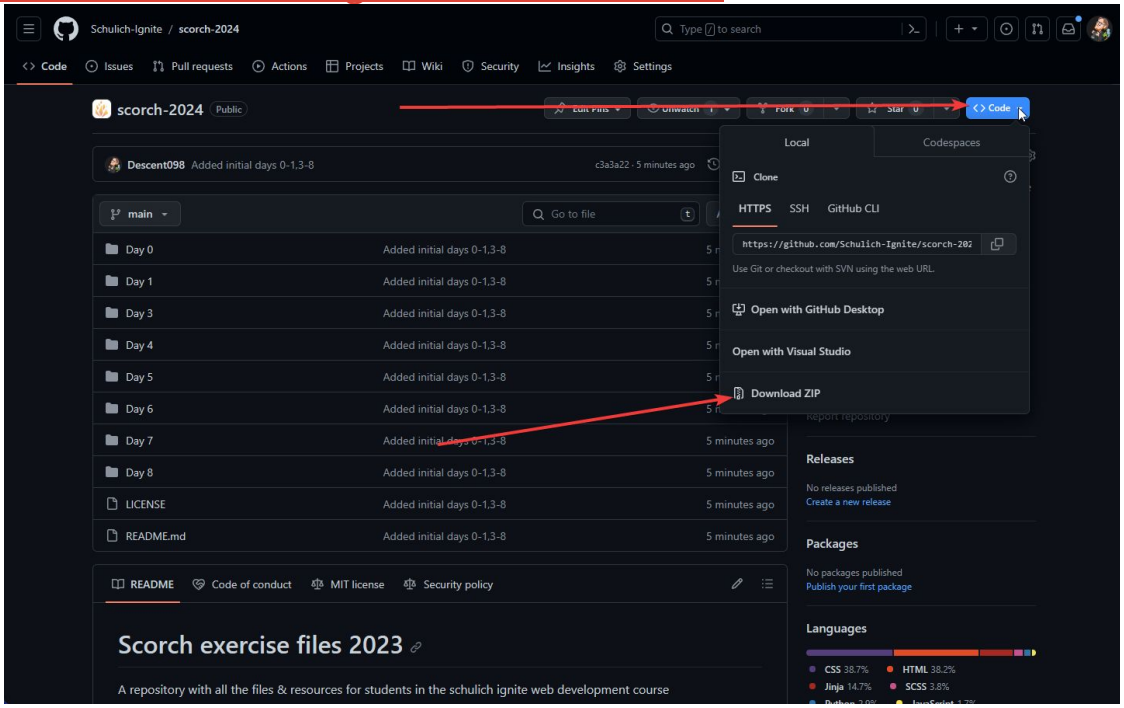
Making a Github Account

5. You should have a page available at https://github.com/<username> that looks like this:



Our Code for Sessions

You can download our code for our sessions here: https://github.com/Schulich-Ignite/scorch-2024



Extract The Files you just downloaded

You will get a .zip file in your downloads.

You should be able to open it like a normal folder and drag the contents wherever is convenient for you. We will be using these files a lot so make sure you put them somewhere that is accessible

Check today's files

If you go to the day 0 folder there will be a file called test.html in the exercises folder you should be able to double-click and open it with the new browser you installed. It should look like this:

Opening .md files

Any .md file that is inside the exercise folders can be more easily read by opening them in VS code and using:

```
    Ctrl + shift + v
Or
    Cmd + shift + v
```

To preview them

Troubleshooting

Had problems setting anything up?

We will go into a quick **breakout room** to make sure everyone has it working!

Or if everything works, have a chat with your mentor :)

Time to play some games!

You will now be sent to a **breakout room** with your **mentor**

- Introduction activity (see next slide)
- Ask any questions you may have
- Play some games!

Icebreaker activity with your group!

Time for introductions! With your group answer the following questions:

- What's your name?
- What is your favorite smell?
- Name something about programming that interests you

Games to play

Here are some game ideas, but feel free to play whatever you like, or just chat!

- Two truths and one lie
- Trivia
- Among Us
- Skribbl.io
  - https://skribbl.io/

Before we go...

Remember to join Discord!
https://discord.gg/TnxBHQTU

See you soon!

- 6:00pm-7:30pm MST October 20th, 2023
- Every Friday

Thanks for coming!