Introduction

In this document, we introduce how to start planning and creating web pages. You will build and style a web page with a clickable button using HTML, CSS and Javascript. You will create a web page that provides the locations of skateboarding sites in Edinburgh, Scotland.

Planning your web pages

What do you have to think about? Let's start a plan for your webpages.

1 Goal

What is the goal of your site?

- Is it to provide information?
- To get bookings for a service or visit?
- To sell a product?
- A place for people to chat and share information?

Deciding on a clear site goal (or goals!) will help the planning decisions.

2 Your users

- Who will use your site? Can you identify a group or group of people? Maybe they are from the same area, have a shared interest, maybe they are likely to be in an age bracket like 12-16 or 25-35.
- What does your users need to achieve on your site?

It's useful to create something called a persona, that identifies who would use your site and what they need from it.

Sample persona

A persona should answer:

- Who are you?
- What's your main goal?
- What's your barrier (or pain point) to achieving this goal?

Image of a sample persona

This persona is for Alex, a software engineer, tasked with developing a weather app.. There are 6 areas to consider:

- Demographics – age, job, location
- Summary – 2 or 3 sentences about the person – what they like to do, their job.
- Needs and goals – what weather information do they need?
- Motivations – why do they need the weather info.?
- Activities – what are Alex's activities that relate to the weather info.? For example, she likes water sports like surfing.
- Pain points – this tells us what her problems are – what stops her achieving her needs and goals? These pain points are the problems that your website should solve. In this case, the

problem is she doesn't have one place to find accurate weather conditions, so that she can plan her outings to enjoy water sports.

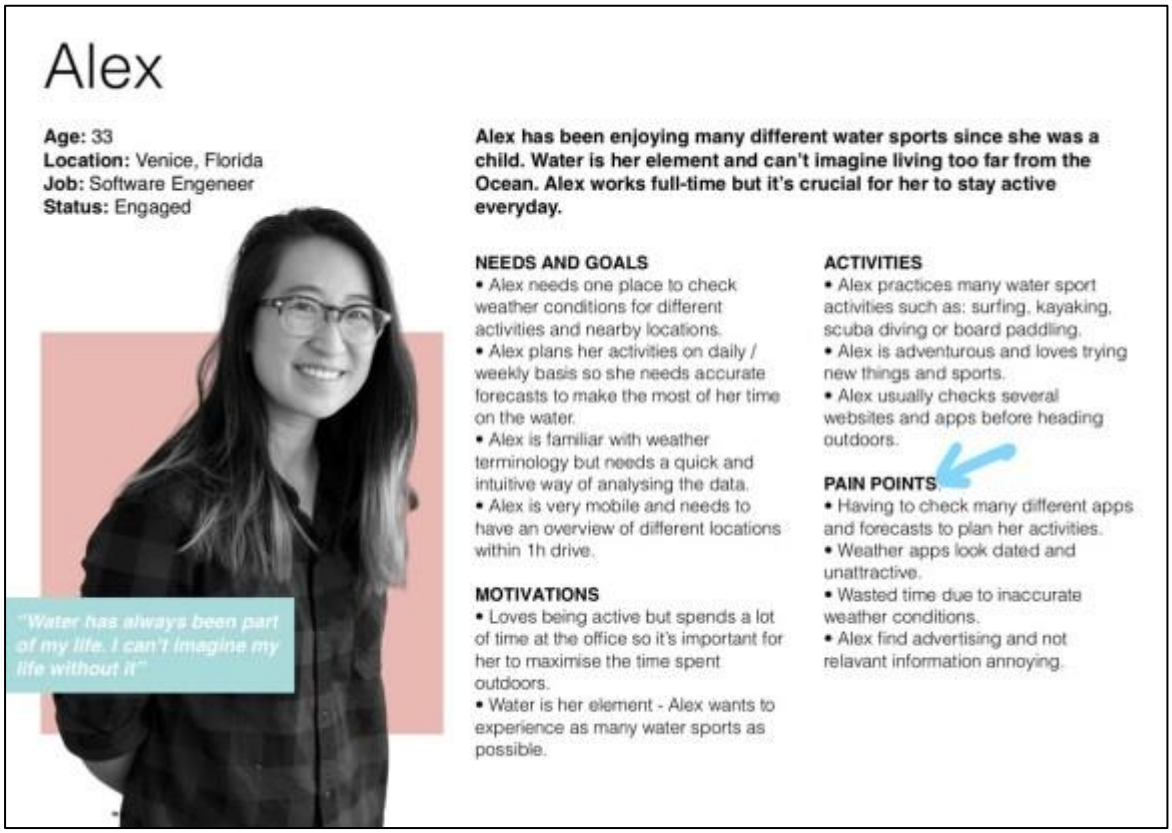


Figure 1. A user persona of Alex, a software engineer

Practical activity: create your own persona

Have a go! We have given you two blank personas below – try talking to people you know who you think might use our skate site. Find out more about them, using the section titles as a guide.

How to complete:

- Blank section at top – give some basic info., such as name, age, location and what they do (job, student, volunteering).
- Interests – what do they like to do?
- Goals – these should be related to your site… such as what information do they need regarding skateboarding websites
- Challenges – what do they want to get out of your stakeboarding site? Do they want information? Top locations? Advice? Images?
- Technology – what features should your website include? E.g. a button, links, menus, tiles, slides, videos, etc.

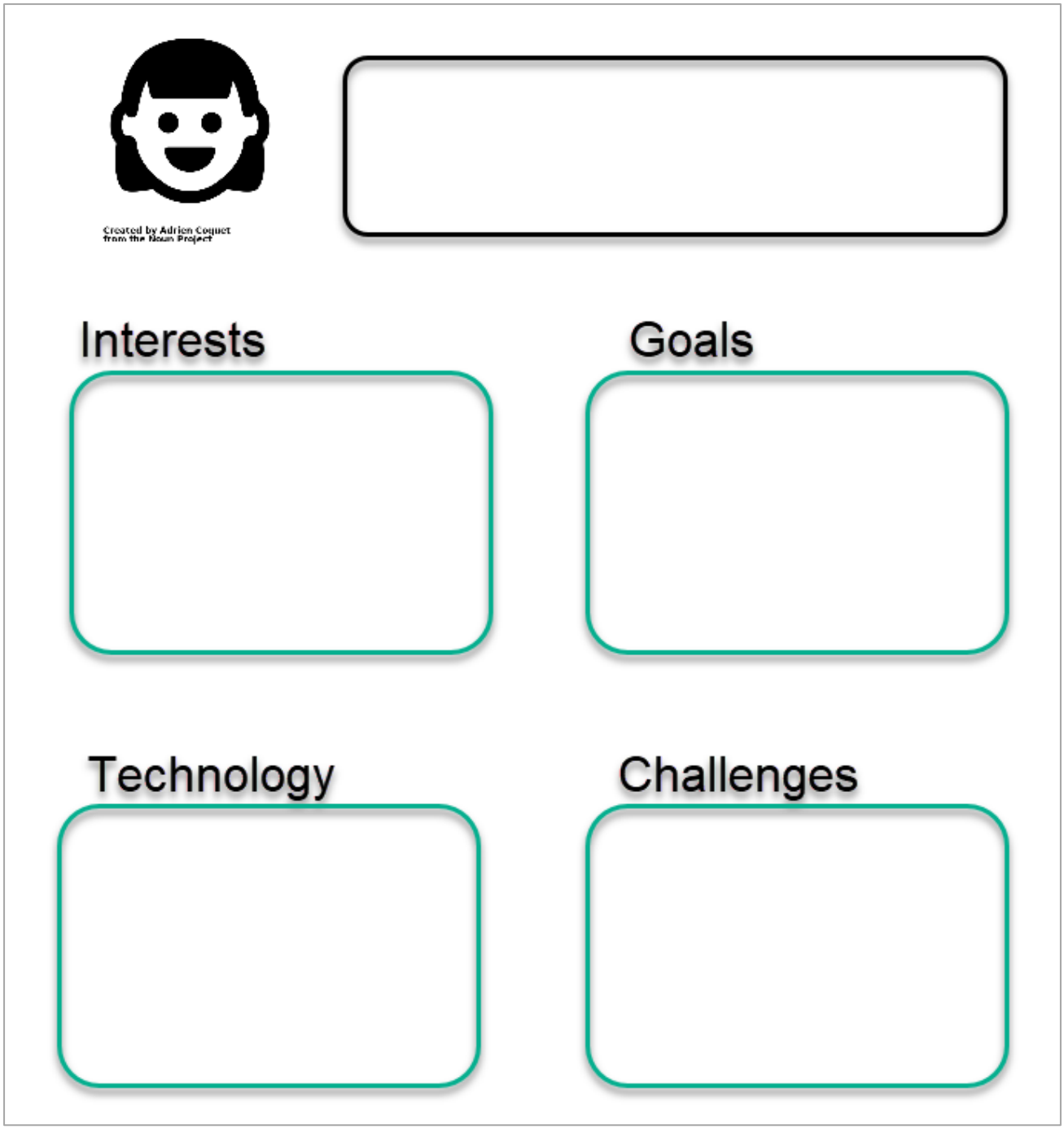


Figure 2. Create your own persona

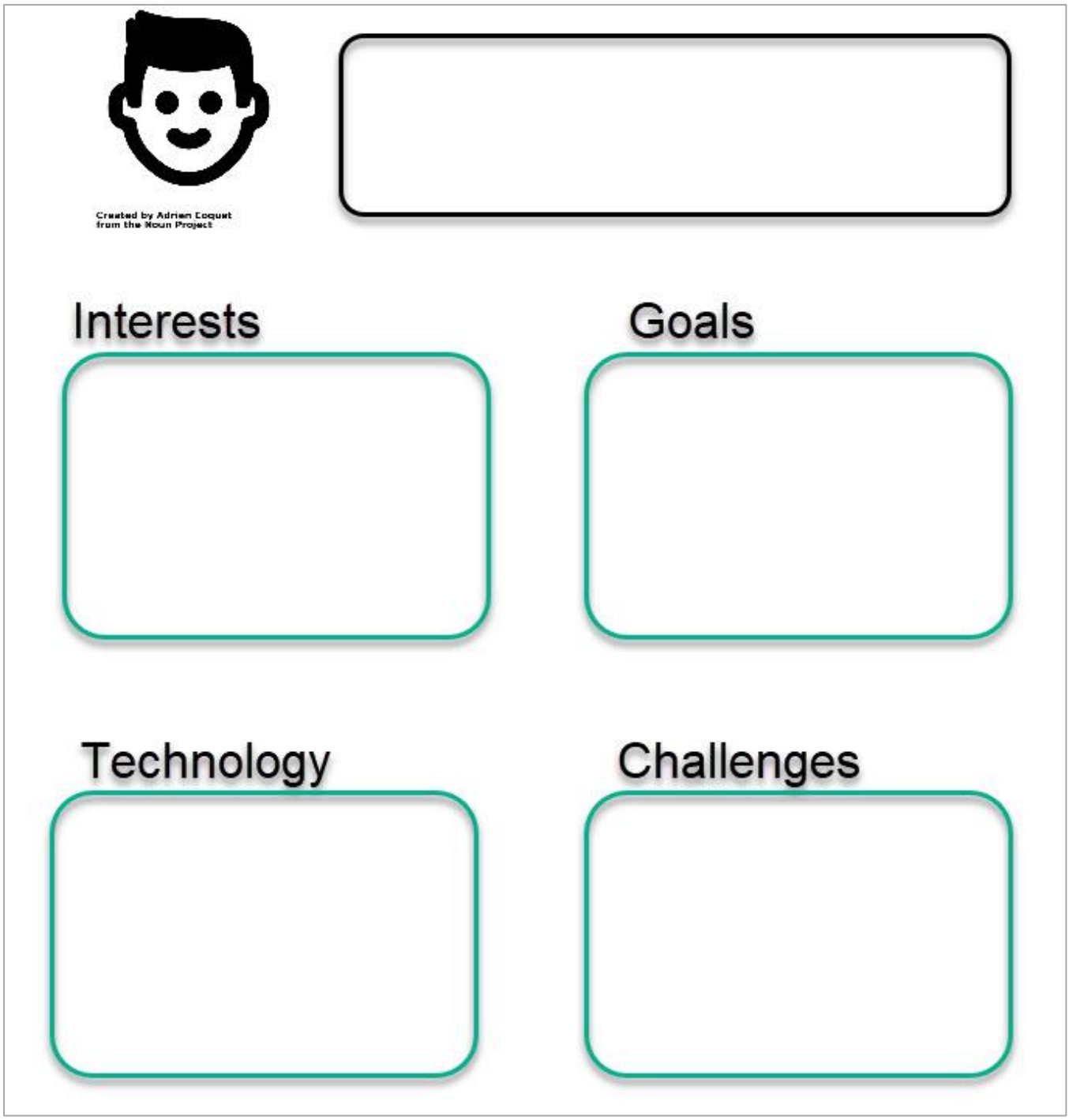


Figure 3. Create your own persona

3 Requirements

Before you start coding your website, you need a list of everything your site needs to do and achieve. The personas will help you create this list of requirements. Here is an example list:

- User can search or browse skate site locations.
- User can view locations on a map.
- Works on mobile phones.
- Works on tablets.
- Accessible – The Web is fundamentally designed to work for all people, whatever their hardware, software, language, location, or ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability. You can gind out more information on web accessibility in the W3C Accessibility fundamentals.

4 Research other sites

There are lots of website examples, popular types of website include:

- eCommerce website.
- Business website.
- Blog website.
- Portfolio website.
- Event website.
- Personal website.
- Membership website.
- Nonprofit website

Looking at other sites is helpful for two reasons:

You can figure out what appeals to you visually

Have a look at sites that you use and consider the style of them. What fonts/colours/style of images do you like? Try and find some sites that are popular with your typical users and get some ideas.

There are lots of sites to help you choose fonts and colour palettes – have a play:

- https://www.fontspace.com/font-generator
- https://coolors.co/
- https://www.canva.com/colors/color-palette-generator/

How are similar sites organised and what do they offer

What pages do similar sites have? What features do they offer, for example; links to social media, the ability to comment, or to save favourite items?

For this project, maybe these sites could be useful to look at:

- https://skateboardscotland.com/
- https://tonyhawk.com/
- https://activeforlife.com/skateboarding-101/
- https://www.skateboardershq.com/

5 Plan your site

Create a site plan – what pages will you need. For this exercise let's just create one page. If you expand the website in the future, you might need several pages, for example:

- Home page
- About page
- Contact us
- Find a skate spot
- Getting started with skateboarding

6 Source your data

Pull together a list of all the skate sites that you plan to share. What information will you need to share? For a start; site name, location, opening hours, booking info., cost, and anything else you think will be helpful.

Name	Address1	Address2	Postcode	Cost	Open
Saughton Skatepark	Balgreen Road		EH11 3AX	Free	24/7
Portobello Skatepark	Treverlen Park	Duddingston Road	EH15 1NF	Free	24/7

Practical: Skate site

To try this out, we are going to create a web page to provide information on places to skateboard or pump tracks in your area.

Let's get started!

We will be creating our content and customising it in Noteable JupyterLab.

1. Login to Noteable.
2. Select Schools Web Development and select Start at the start screen, as shown in the image below:

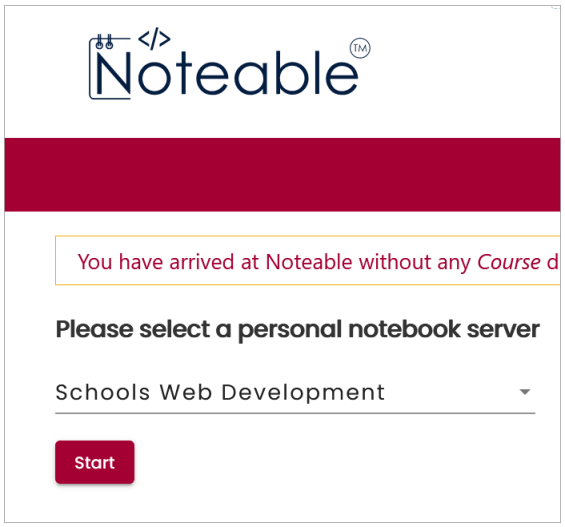


Figure 4. Log into Noteable

3. Select Upload Files – the arrow underneath the menu at the top left:

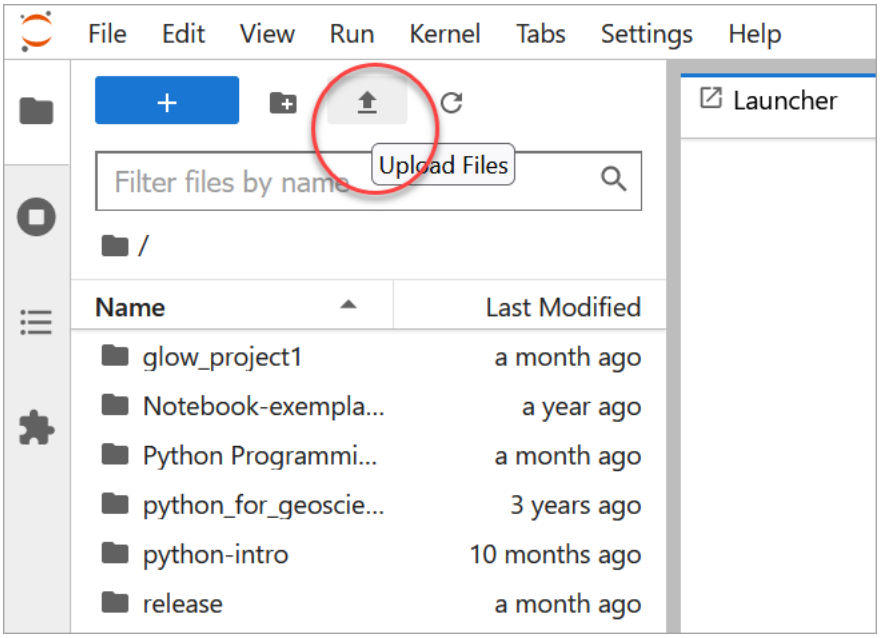


Figure 5. Upload Files in Noteable

4. Find the zip file **glow_project2.zip.**
5. Select it.
6. Select Open – this is the option in Windows – you may have a different option if you are using an ipad or macbook.
7. You should see the file appear in your list on the left of Noteable Jupyterlab.
8. Right click on it and select **Extract Archive:**

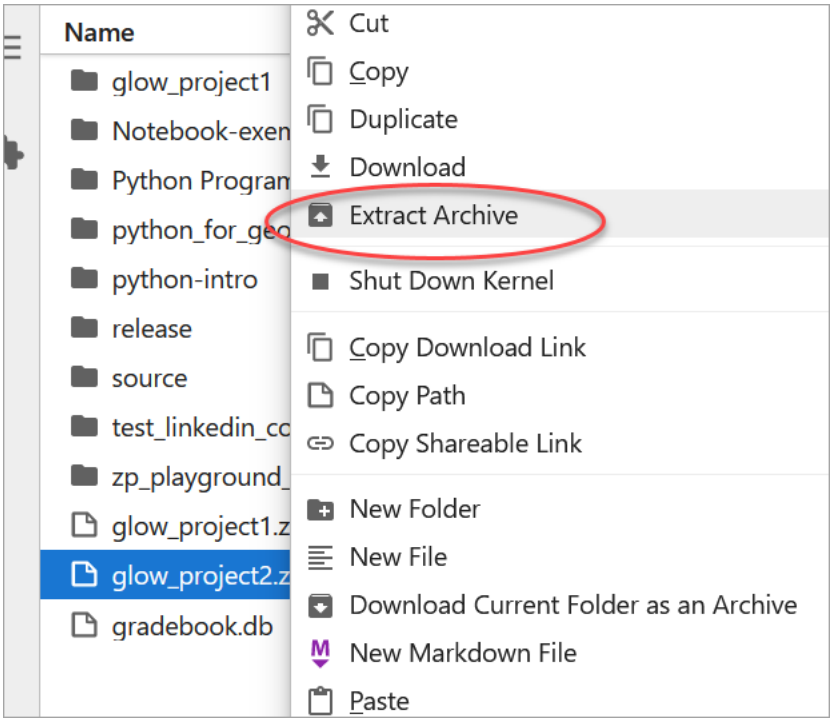


Figure 6. Extract Archive

9. Now you will see a folder, **glow_project2,** in your list.
10. Select the folder to open it.
11. Let's have a look at the contents – the image below shows what you should see in the folder:

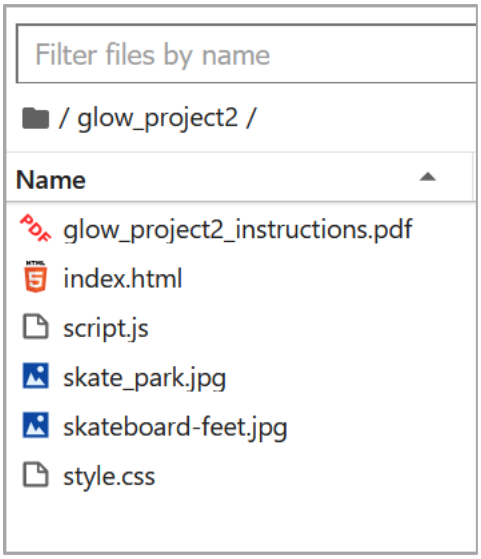


Figure 7. GLOW Project 2

- Instructions.pdf – a file you can open and view in Noteable JupyterLab with the same content as this word document.
- Index.html – where you input your HTML code.
- Script.js – where you input your Javascript.
- Skate_park.jpg ad skateboard-feet.jpg – images you will add to your HTML.
- Style.css – where you will add styling information.

Let's go ahead and create our page.

Add HTML

HTML Basics

First let's familiarise ourselves with some HTML basics.

HTML Elements

Elements are the skeleton of your page. A heading is an element. Or a paragraph. Or an image. You will use several elements in this project.

Elements have a start and end tag. The tags are between angle brackets. The content goes in between the tags. Notice the end tag has a forward slash before the tag name.

Heading 1 example

> **<h1>**My first website**</h1>**

Paragraph example

> ** **Welcome to my site. Here I share my portfolio, work experience and examples of my writing.** **

HTML Attributes

An attribute gives more information about the HTML element. The attribute is contained within the element tags.

Let's look at the image element, with some attributes added:

> ****

Breaking this down…

Image element

> **<img>** - IMG is the element you use to add an image.

SRC and ALT attributes

> **src="skateboard-feet.jpg"** – SRC is the attribute that tells us which image to add and where to find i, where the source of the image is.
>
> **alt="Image of feet on a skateboard with a sunset or sunrise" –** ALT is the attribute for alternative text. So if our image does not display on the page, users will be able to see text that describes it.

Learn more about HTML

W3C Schools is a great site for learning HTML and other languages:
https://www.w3schools.com/html/default.asp

Open the Index.html file

1. Right click on the file index.html,
2. Select Open With,
3. Then Editor.

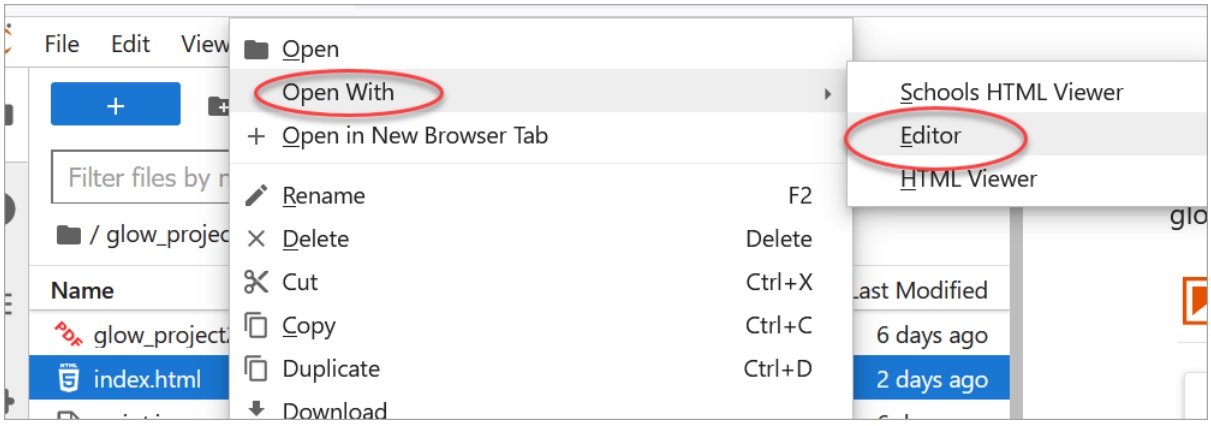


Figure 8. Opening an HTML file with the editor in Noteable Jupyterlab.

Existing HTML code

There is already some code in the HTML file. Let's have a look:

HTML element

<html lang="en"> - The HTML element states this page is HTML code, using English language. This is the opening tag – you will find a closing tag at the bottom of the index.html page.

</html> - Note the closing tag has a forward slash.

Head Element

Next, you see some code in the **<head>** element. The head contains information about the document. The user won't see any of this content of the <head> element on the web page.

<head> - Opening tag.

<link rel="stylesheet" href="./style.css" /> - This code is telling the page to look at the file style.css for style information. The file style.css is where you will put code that details what fonts and colours to use, as well as other styling information.

<script type="text/javascript" src="./script.js"></script> - This code is telling the page to look at the file script.js for Javascript. The file script.js is where you will put Javascript that will make some of your HTML interactive.

<title>home**</title>** Page title.

</head> Closing tag.

Add Body Element

We need to add the **<body>** element to our page. The body is where you input all the content that the user sees on your page.

1. Let's add the opening body tag:

   <body>

2. Now add a heading 1:

   <h1>Edinburgh Skate spots</h1>

3. And a heading 2:

   <h2>Welcome to our community site!</h2>

4. Let's add our first paragraph:

** The place to find skate spots in Edinburgh. Just select any button to get more details of a site. **

Add an image

5. Let's add in an image – we'll add in the **skateboard-feet** image you have in your folder. The SRC attribute gives the location of your image:

   ****

6. We need to add the ALT attribute. This is alternative text that will display if your user's device can't display the image:

   ****

Add a button

Next let's add in our button. Our plan is to add a button that the user can click. When they click, a text box will appear with details of the skate park.

7. First, add the button element with the text Saughton Park (this is the first skate site we will add):

   <button>Saughton Park</button>

8. Last, we want to add a class or attribute to the button. This will tell our Javascript what type of button it is and our CSS styling how to style this button– add in **class=”clickme”** in the opening tag:

   <button class="clickme">Saughton Park</button>

Add a section

Next we want to add a section – the section will have the text that appears when the user clicks on the button.

9. Add opening and closing section tags.

   <section>

   </section>

10. Let’s add an attribute, **hidden.** We want the section to be hidden until the box is clicked.

11. We add an attribute within the opening tag – with the word class, an equal sign, then the attribute in quotation marks:

   <section class="hidden">

   </section>

12. Now let’s add the text that will appear when the button is clicked. We will add a Heading 1 and paragraphs for the address and other details:

   <section class="hidden">

   <h1>Saughton Skatepark</h1>

   ** Balgreen Road, EH11 3AX **

   ** Free **

   ** Open 24/7 **

   </section>

Add link

For our section, it would be useful to add a link to a website with more details about the skate park. The structure of a link is:

Visit website

We can break this down into 4 parts:

- <a> the opening tag
- href=”the link to the website, in quotation marks”>

- The text that describes the link
- – the closing tag

13. Let's add our link as a paragraph in our section. Under the last paragraph (Open 24/7), add a paragraph tag:

    ** **

14. Then add the link, as shown here:

    Visit website

15. Finally, add a paragraph closing tag:

    ** **

16. Make sure you have a section closing tag:

    </section>

17. After your section, make sure there is a closing body tag.

    </body>

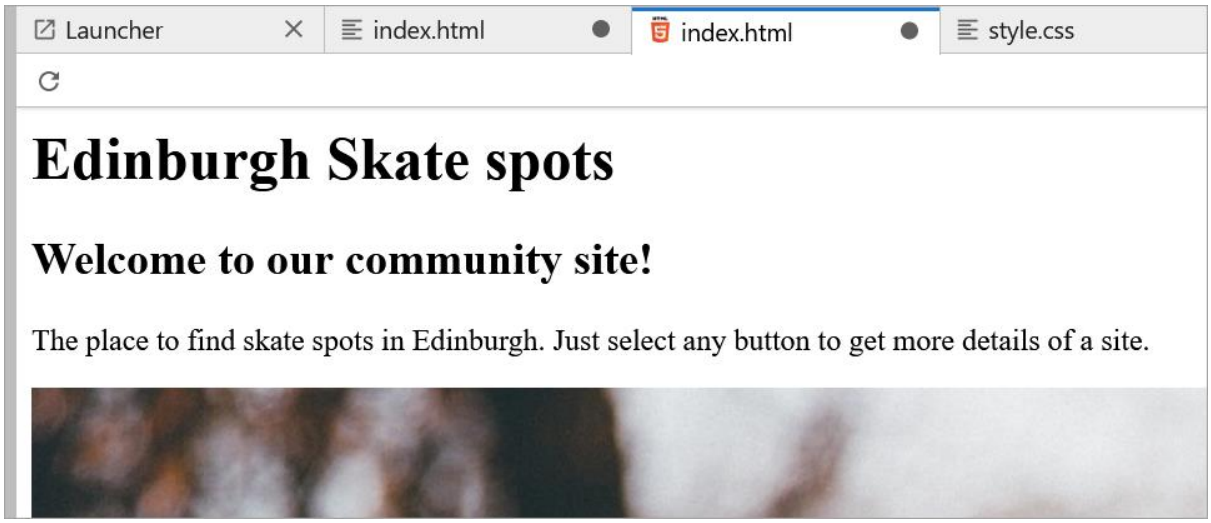
18. You should have a closing html tag:

    </html>

Check out your progress

We have added all the HTML. Have a look at it in the HTML viewer.

19. Double click Index.html to open it as a viewer (NOT an editor). You should see something similar to the image below.
20. Now let's move on and add some styling with CSS.



Add CSS

Now let's add some styling. You can use CSS (Cascading Style Sheets) to change fonts, colours, borders and lots more.

To learn more, check out the W3C site:

https://www.w3schools.com/css/default.asp#

CSS Rules

You add CSS with:

1. a Selector (this is the HTML element you want to style, for example H1).
2. a Declaration – this gives the property and the value. So in the image below:
    a. H1 is the selector – we are styling our heading 1.
    b. Color is the first property – this will change our font colour.
    c. Blue is the first value – our font will be blue.
    d. Font-size is the second property.
    e. 12px is the second value – our font will be sized at 12 pixels.

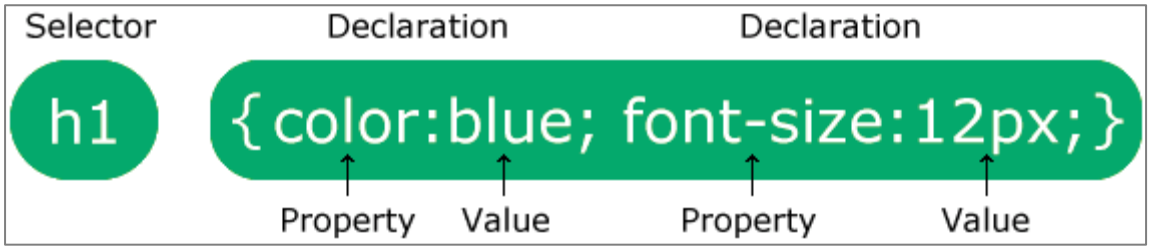


Figure 9 CSS rule, from https://www.w3schools.com/css/css_syntax.asp

Fonts

The font will always be Times New Roman by default. Let's change it to something we prefer.

You can find lots of information about choosing fonts here:

https://www.w3schools.com/css/css_font.asp

1. Let's add a different body font, Open Sans.
2. To start CSS, you first name the element, then add a pair of curly brackets (usually beside the P on a keyboard) – so we want to type:

```
body {
}
```

3. Now you input the descriptors and values between the brackets.

```
body {font-family: "Open Sans", sans-serif;

}
```

Our CSS rule is going to style our font as Open Sans. If that font is not available, it will use any available font that is sans-serif (it doesn't have strokes at the end of the letters).

4. We want to change our font size. Let's add another property and value to our rule.
5. After the semi-colon after **sans-serif,** add:

```
body {font-family: "Open Sans", sans-serif;

font-size: 16px;

}
```

6. Now let's change the font on our Heading 1 and Heading 2. Again start with the element then a pair of curly brackets:

```
h1 {

}

H2 {

}
```

7. Then add the code for font family and size:

```
h1 {

 font-family: "Proxima Nova", sans-serif;

 font-size: 30px;

}

h2 {

   font-family: "Open Sans", sans-serif;

   font-size: 20px;

}
```

Colours

There are a few ways to add a colour in CSS; colour names, HEX codes, HSL codes and other methods – find out more here: https://www.w3schools.com/css/css_colors.asp

We will use a colour name.

8. Let's give our page background a different colour.

```
body {background-color: lightblue;
```

```
}
```

Images

We added one image. We want it to adjust its size, depending on how big the user's screen is. If we set a maximum width and automatic height, it will adjust.

9. Input the img selector and curly brackets.
10. Input a max-width of 60%.
11. Input height to auto.

```
img {
  max-width: 60%;
  height: auto;
}
```

Button

Let's create a style for our button.

12. Type in button and your curly brackets.
13. We will change the background colour to cadetblue.
14. Give a font size of 16 pixels.
15. Font colour is darkslateblue.

```
button {
   background-color: cadetblue;
   font-size: 16px;
   color: darkslateblue;
}
```

Next, let's add some padding and a border.

16. Input padding of 10 pixels – this will add some space around the text.
17. Type in a border-radius of 10 pixels. This rounds the corners of our button.
18. Finally add a border to the box. 3 pixels, colour is darkblue.

```
   padding: 10px;
   border-radius: 10px;
   border: 3px solid darkblue;
}
```

Button hover

Remember that we are going to make this button clickable. We want the style to change when a user is hovering over the button. Let's create a different style for the hover state.

19. Type button:hover and your curly brackets.
20. Change the background colour to darkblue.
21. Font colour is white.

```
button:hover {
background-color: darkblue;
color: white
}
```

Section

Now we want to style our section, which will appear when our box is clicked. Once we add our Javascript, there will be 2 possible classes for the section – *hidden* or *showing.*

22. Add the following code – this ensures nothing is displayed for the hidden class – adding a dot before the word hidden tells CSS it is a class:

```
.hidden {
 display: none;
}
```

Now we will create a style for when our section is visible – so we start with the class *showing.*

23. Add the class .showing and your curly brackets.
24. Add display: block – to make sure our information text appears in a block.
25. Add font family, background and font colours as shown (#ffffff and #000000 are the Hex colour codes for white and black):

```
.showing {
   display: block;
   font-family: "Arial", sans-serif;
   background-color: #ffffff;
   color: #000000;
}
```

Now let's make some other changes to the section's style.

26. Input width of 400 pixels.
27. Add padding of 20 pixels – remember this adds space around the text within the block.

28. Add a top margin of 20 pixels. This creates a bit of space between the block and the box above it.
29. Finally, add border radius of 10 pixels, to round the corners.

```
    width: 400px;
    padding: 20px;
    margin-top: 20px;
    border-radius: 10px;
}
```

Add Javascript

Javascript is a very popular programming language. You can use it to add interactivity to your page.

Learn more about Javascript here: https://www.w3schools.com/js/default.asp

We want a user to click on our button and for the section (with the address) to appear.

1. Open your script.js file.
2. Copy and paste this code:

```
document.querySelector(".clickme").addEventListener("click", () => {
  document.querySelectorAll(".hidden").forEach((item) => {
    item.classList.toggle("showing");
 });
});
```

3. Let's break down this code to understand what it is doing:

document.querySelector(".clickme").addEventListener("click", () => {

This line is looking for the class .clickme. When it finds the class, it will add an EventListener….so that when the class is clicked, something will happen.

document.querySelectorAll(".hidden").forEach((item) => {

This line is saying whenever the class **.hidden** is found, for each item with the class…

item.classList.toggle("showing");

…it will toggle with the class **.showing.** This means it will switch to the style for the class .showing.

HTML Reference

```
<html lang="en">
  <head>
    <link rel="stylesheet" href="./style.css" />

    <title>Home</title>
  </head>
    <body>
      <h1>Edinburgh Skate spots</h1>
      <h2>Welcome to our community site!</h2>
      <p>The place to find skate spots in Edinburgh. Just select any button to get more details of a site.</p>

    <img src="skateboard-feet.jpg" alt="Image of feet on a skateboard with a sunset or sunrise"/>
      <br>
    <script type="text/javascript" src="./script.js"></script>
      <button class="clickme">Saughton Park</button>
      <section class="hidden">
        <h1>Saughton Skatepark</h1>
        <p>Balgreen Road, EH11 3AX</p>
        <p><a href="https://skateparks.skateboardscotland.com/view/saughton-skatepark/">Visit website</a></p>
        <p>Free</p>
        <p>Open 24/7</p>
        </section>
    </body>
</html>
```

CSS Reference

```
body {
  font-family: "Open Sans", sans-serif;
  font-size: 16px;
}
body {
  background-color: lightblue;
}
h1 {
  font-family: Proxima Nova, sans-serif;
  font-size: 30px;
}
h2 {
    font-family: "Open Sans", sans-serif;
    font-size: 20px;
}
img {
  max-width: 60%;
  height: auto;
}
button {
    background-color: cadetblue;
    font-size: 16px;
    color: darkslateblue;
    padding: 10px;
    border-radius: 10px;
    border: 3px solid darkblue;
}
button:hover {
    background-color: darkblue;
```

```
    color: white;
}
.hidden {
    display: none;
}
.showing {
    display: block;
    font-family: "Arial", sans-serif;
    background-color: #ffffff;
    color: #000000;
    width: 400px;
    padding: 20px;
    margin-top: 20px;
    border-radius: 10px;
}
```

Javascript reference

We want a user to click on our button and for the section (with the address) to appear.

1. Open your script.js file.
2. Copy and paste this code:

```
document.querySelector(".clickme").addEventListener("click", () => {
   document.querySelectorAll(".hidden").forEach((item) => {
      item.classList.toggle("showing");
  });
});
```

3. Let's break down this code to understand what it is doing:

document.querySelector(".clickme").addEventListener("click", () => {

This line is looking for the class .clickme. When it finds the class, it will add an EventListener….so that when the class is clicked, something will happen.

document.querySelectorAll(".hidden").forEach((item) => {

This line is saying whenever the class **.hidden** is found, for each item with the class…

item.classList.toggle("showing");

…it will toggle with the class **.showing.** This means it will switch to the style for the class .showing.