HTML & CSS
SAVING YOUR WORK

Track your progess by ticking off the boxes below:

Introduction: It's important to **save** your work regularly, so you don't lose any of the brilliant ideas you've come up with. Let's have a look at where to save your code and various other things you might need (like images) so that the browser can find them correctly.

Previewing the changes you make

When you make changes in the **text editor**, you have to **save** them before you can **preview** them in the browser. Once you've saved them, you have to **refresh** the browser. If you're on **Windows**, often you can use a keyboard shortcut like **CTRL+S** to save the file you're editing, and many browsers will allow you to refresh the page with **CTRL+R** shortcut or **F5** key (if using **Internet Explorer**). On the **Mac** these shortcuts are similar, **⌘+S** to save document you're editing and **⌘+R** to refresh the page in the browser.

Editing pages as you go

But what if you want to try something really quickly without switching between the editor and the browser? If you have used developer tools before, then forgive us explaining how to do it again, but if not, prepare to be amazed!

1. Go to your **web browser**. ☐
2. Right click anything interesting, and then click **Inspect element**. A panel (let's call it the **developer** panel) will open up. It will show you the page's code as the same time as showing you the page.
3. Move your **mouse** over different pieces of code. The corresponding things on the page will be highlighted, so you can see which bit does what. ☐
4. On the left, the panel shows you the **HTML structure** of your document. If you click on any **element**, on the right you will see which bits of **CSS** this element uses. This is useful when you want to focus on one thing without having to read all the CSS you've written. ☐
5. **Double-clicking** on the **HTML** or **single-clicking** on **CSS** allows you to edit the code. Any changes are applied immediately, so you can clearly see their effect. ☐

Track your progess by ticking off the boxes below:

6. The catch is that these changes aren't actually **saved** – they are only kept until you **refresh** the page. If you now go back to your document you can see that your code hasn't changed. To **save** the changes, you have to either **type them in** or **copy them** from the **developer panel**. ☐

Saving things in the right place

When your site needs only one file – the **HTML document** – then your job is easy. You can move it anywhere, put it on a memory stick, move it inside **directories** to organise it, etc. and it still works perfectly. Once you start **linking** to other pages, including pictures, or audio clips, videos and the like, you need to remember a few simple rules.

Case #1 – all files live in the same directory

Let's say you have your **page.html** on the desktop, and it includes a picture of a kitty, like this: ☐

```
<img src="kitty.jpg" alt="A kitty playing with some wool.">
```

The browser will look for the **kitty.jpg** file in the same directory as the page, in our case the desktop (desktop is just a directory like any other, only a little bit special).

Case #2 – going into different directories

What if we wanted to copy our site onto a memory stick? We would have to move both files and make sure they live in the same directory on the stick. Maybe we can make it easier by putting them in their own directory and move that. So our file structure now looks like this: ☐

```
my_kitty_site
|
\- page.html
|
\ kitty.jpg
```

Track your progess by ticking off the boxes below:

Let's tidy it up a bit more. What if later we want to include more **images**? Let's put them in their own **directory** so it's easier to find them.

```
my_kitty_site
|
\- page.html
|
\ - images
      |
      \ kitty.jpg
      |
      \ kitty_and_puppy.jpg
```

Now we we open up the page in the **browser**, the image doesn't show up! This is because the browser can't find it. Let's help it by adding what we call a **path** – a route through the **directories** it has to take to find the files, starting at the **page**. In our case we would have to go into **images directory** before we get to the picture, like this:

```
<img src="images/kitty.jpg" alt="A kitty playing with some wool.">
```

The images/ bit before the name of the kitty image file tells the browser to go into the **images directory** first. **Save**, **refresh**, and the kitty is back!

Case #3 - coming out of directories

What if you fancied tidying up your **file structure** some more and put all your **HTML documents** in their own **directories**, like we did with images?

Track your progess by ticking off the boxes below:

Let's say you created the following: ☐

```
my_kitty_site
|
\ - pages
|     |
|     \ - page.html
|
\ - images
      |
      \ kitty.jpg
      |
      \ kitty_and_puppy.jpg
```

Again, when we open up the **page.html** our images don't work. This is because the browser looks for files in relation to the **page** you're looking at. When we said:

```
<img src="images/kitty.jpg" alt="A kitty playing with some wool.">
```

We meant: "from **page.html** go into directory called **images** where you will find **kitty.jpg**". But there are no images inside pages directory, where **page.html** lives! What we **wanted to say** was: "from **page.html** go **one directory up**, then go into directory called **images** where you will find **kitty.jpg**". So how can we say it?

```
<img src="../images/kitty.jpg" alt="A kitty playing with some
wool.">
```

☐

../ means "**go one directory up**". The value of the whole **src attribute** can then be broken down into these parts:

../ – go one directory up **images/** – go into the images directory kitty.jpg - and grab **kitty.jpg file**

Things to try

A). **Organise** your files however you like. Try to figure out how to make sure the **browser** can find them. ☐

B). What if you wanted the browser to look for a file **two directories up**? How would you describe that? ☐