Australian **Computing** Academy

DT Challenge JavaScript

Cookie Clicker

1. Showing the cookie
2. Clicking the cookie
3. Selecting the cookie
4. Counting cookies
5. Extensions: Decorating the cookie

The Australian Digital Technologies Challenges is an initiative of, and funded by the Australian Government Department of Education and Training (https://www.education.gov.au/).

SHOWING THE COOKIE

1.1. A cookie web page

1.1.1. What is a web page?

You've probably used lots and lots of web pages already, but what are they exactly?

A web page is made up of three different things: HTML, CSS and JavaScript.

We'll get to CSS and JavaScript later. For now let's explore HTML.

1.1.2. What is HTML?

Hyper Text Markup Language, or HTML, is the language of a web page. It's *Hyper Text* because it is more fancy than just text. And it's a *Markup Language* because it uses a language to add information to plain old text.

Here's an example of a very simple HTML page.

```
<p>Hello, World!</p>
```

Hello, World!

Notice the `<p>` and `</p>`? The `<p>` is called an opening paragraph tag and the `</p>` is called a closing paragraph tag. Together they're called a `p` element or a paragraph element.

1.1.3. Problem: Hello, World!

It's time to write some HTML!

Using the paragraph tags write the following text:

```
Hello, World!
```

> **Hint**
>
> A paragraph element is made up of an opening tag `<p>` then some text and then a closing tag `</p>`.

Testing

- [ ] Checking there is an `p` heading element.
- [ ] Checking the `p` paragraph is the only element in the document.
- [ ] Checking the text inside the `p` (paragraph) element.
- [ ] Checking your document visually against the expected image.

1.2. Cookie heading

1.2.1. Adding headings

Web pages aren't just made of paragraphs of text, they also have headings!

Headings come in six different sizes. The biggest heading is the `h1` element and the smallest is the `h6` element.

Here's an example of a web page with headings:

```
<h1>The headings web page</h1>
<p>
  What is a heading? Well, the title above this paragraph is a heading. It's a h1 to
</p>
<h2>All the types of headings</h2>
<p>
  The types of headings are h1, h2, h3, h4, h5, and h6 in order of largest to smalles
</p>
<h2>Use headings to title sections</h2>
<p>
  Headings are used to help show a reader the sections and subsections of a web page.
</p>
<h3>What about sub-sub sections?</h3>
<p>
  You can use smaller and smaller headings for each section title. Like the heading a
</p>
```

The headings web page

What is a heading? Well, the title above this paragraph is a heading. It's a h1 to be precise.

All the types of headings

The types of headings are h1, h2, h3, h4, h5, and h6 in order of largest to smallest.

Use headings to title sections

Headings are used to help show a reader the sections and subsections of a web page.

What about sub-sub sections?

You can use smaller and smaller headings for each section title. Like the heading above, which is a

1.2.2. Problem: Australian Cookie Academy

The Australian Cookie Academy is opening a new online bakery! They want to get people excited before the online store launches.

Create a web page with an `h1` element that contains the following text:

```
Australian Cookie Academy
```

And an `h2` element with this text inside:

```
Online store coming soon!
```

> **Headings**
>
> A heading element is made up of an opening tag `<h1>` then some text and then a closing tag `</h1>`.
>
> You can use any number from 1-6 to represent the level of the heading. 1 is used for the most important headings!

Testing

- [ ] Checking there is an `h1` heading element.
- [ ] Checking the `h1` heading is the first element in the document.
- [ ] Checking the text inside the `h1` heading element.
- [ ] Checking the capitalisation and punctuation in the `h1` heading element.
- [ ] Checking there is an `h2` heading element.
- [ ] Checking the `h2` heading is the second element in the document.
- [ ] Checking the text inside the `h2` heading element.
- [ ] Checking the capitalisation and punctuation in the `h2` heading element.
- [ ] Checking your web page visually against the expected image.

1.2.3. Problem: Fancy Cookie Club

The Fancy Cookie Club is a club for fans of rare and gourmet cookies! Of course, they need a website so everyone can share the joy of fancy cookies!

Create a web page with an `h1` element that contains the following text:

```
Fancy Cookie Club
```

And an `h2` element with this text inside:

```
Fans of rare and gourmet cookies welcome!
```

> **Headings up!**
>
> Remember a heading element is made up of an opening tag e.g. `<h1>` then some text and then a closing tag `</h1>`.

Testing

- [ ] Checking there is an `h1` heading element.
- [ ] Checking the `h1` heading is the first element in the document.
- [ ] Checking the text in the heading 1 element is correct.
- [ ] Checking there is an `h2` heading element.
- [ ] Checking the `h2` heading is the second element in the document.
- [ ] Checking the text in the heading 2 element is correct.
- [ ] Checking your web page visually against the expected image.

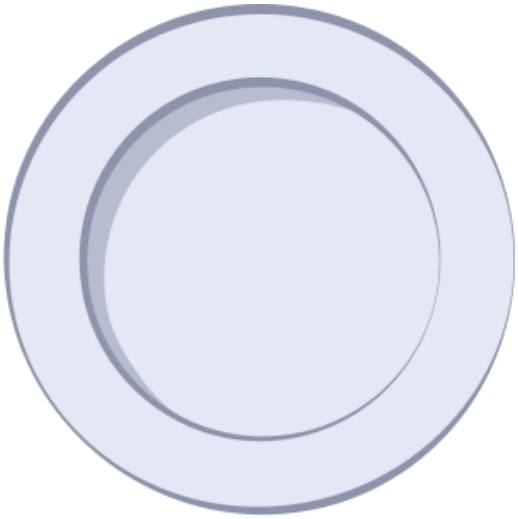
1.3. Cookie picture

1.3.1. Adding pictures

Of course websites aren't all just words, they have pictures too! To add pictures we use an `<img>` tag.

But how does the HTML know which image to use? We use something called the `src` (source) attribute. It looks like this:

```
<img src="plate.png">
```



The `src` attribute tells the `img` tag which image to use. There are lots of other attributes too. They all follow the pattern `attribute="value"`.

No closing tag

Some elements, such as `img` don't need a closing tag because you can't put any text or elements inside of them.

1.3.2. Problem: G-img-erbread

After spending all day baking and decorating you would like to share your wonderful gingerbread creation with the internet.

Your web page has a heading and you've already made a picture of your gingerbread person and saved it as **`gingerbread.png`**. Now you just need to add the image below the heading to finish your webpage!

> **Use the src**
>
> You can use the `<img>` tag without a closing tag.
>
> You'll need to use the `src` attribute to set the picture.

You'll need

index.html

```
<h1>My gingerbread person</h1>
```

Testing

- [ ] Checking there is an `img` element.
- [ ] Checking that the `h1` heading is still the first element in the document.
- [ ] Checking the `img` is after the heading element in the document.
- [ ] Checking that the image has a `src` attribute.
- [ ] Checking that the image has the correct `src` attribute.
- [ ] Checking your web page visually against the expected image.

1.3.3. Problem: Cookie img-ination

You've just had a brilliant dream that you need to share with the world!

Your web page has a heading and you've already made a picture of your dream and saved it as **`cookie-dream.jpg`**. Now you just need to add the image below the heading to finish your webpage!

> **Img in the HTML**
>
> Remember to use the `img` element and `src` attribute to create the picture using HTML!

You'll need

index.html

```
<h1>My cookie dream</h1>
```

Testing

- [ ] Checking there is an `img` element.
- [ ] Checking that the `h1` heading is still the first element in the document.
- [ ] Checking the `img` is after the heading element in the document.
- [ ] Checking that the image has a `src` attribute.
- [ ] Checking that the image has the correct `src` attribute.
- [ ] Checking your web page visually against the expected image.

1.4. Cookie Clicker - step 0

1.4.1. Problem: Project - step 0: No cookies

It's time to start making the cookie clicker game!

When we start our game the player will have no cookies. So we'd like you to put the following on your web page

A `h1` heading that says:

```
No Cookies
```

And then an `img` with the picture:

```
crumbs.png
```

> **Picture this**
>
> Remember, you can create a picture with the using a `src` attribute in the `img` element.

Testing

- [ ] Checking there is an `h1` heading element.
- [ ] Checking the `h1` heading is the first element in the document.
- [ ] Checking the text in the `h1` heading element says 'No cookies'.
- [ ] Checking there is an `img` element.
- [ ] Checking the `img` is after the heading element in the document.
- [ ] Checking that the image has a `src` attribute.
- [ ] Checking that the image has the value 'crumbs.png' in the `src` attribute.
- [ ] Checking your web page visually against the expected image.

CLICKING THE COOKIE

2.1. Interactive cookie

2.1.1. What is JavaScript?

HTML is just for *marking up* plain text information. JavaScript on the other hand can make calculations and make pages interactive (like with a cookie clicker)!

We store JavaScript in a separate file, and then import that file into our HTML page.

> **That's a mouthful**
>
> JavaScript is a lot to type and a lot to say. It is often just called JS for short.

2.1.2. Adding Javascript to a webpage

To add JavaScript to a web page is easy! We use a `<script>` tag with a `src` attribute that gives the name of the JS file.

But what can we put in the JS file? The quickest and easiest thing to do is `alert('Your text here')` which will pop up a message on your web page.

```
<script src="main.js"></script>
```

main.js

```
alert('Hello there!');
```

Try this code in your browser to see the results

2.1.3. Problem: Hello, JavaScript!

We've written the JavaScript for you. But it's not in the HTML page yet!

Use the `<script></script>` element and the `src` attribute to include the JavaScript file `"hello.js"`.

You'll need

index.html

```
<p>You can use a script element to load JavaScript</p>
```

hello.js

```
alert('Hello, this is JavaScript!');
```

Testing

- [ ] Checking you created a `script` element.
- [ ] Checking you used the `src` attribute in the `script` tag.
- [ ] Checking you used the `src` attribute to reference `"hello.js"`.
- [ ] Checking you made the alert say `'Hello, this is JavaScript!'`.

2.1.4. Problem: Hello, Cookies!

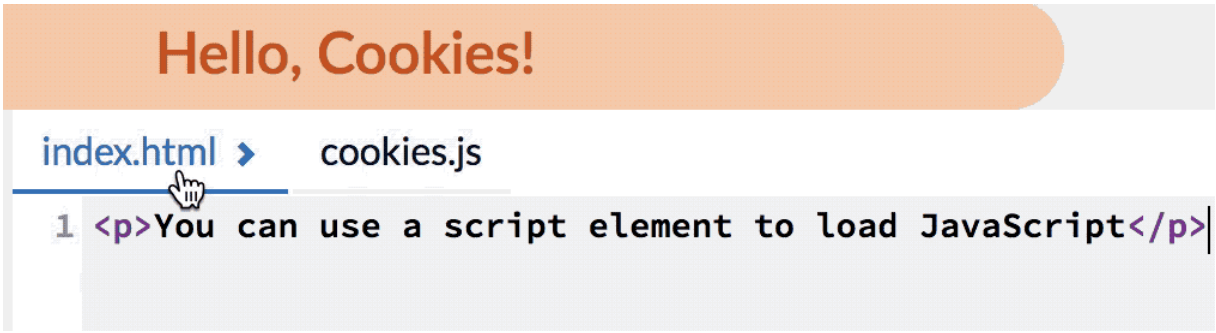
This time you need to add the JavaScript code in addition to the `script` element!

Inside of the HTML reference `"cookies.js"` using a `script` element.

Inside of **`cookies.js`** copy and paste the following JS code:

```
alert('Cookies! My favourite!');
```

> **Add your code to the `cookies.js` file!**
>
> On the workspace to the right where you have been writing code, there is another tab, called **`cookies.js`**.
>
> You will need to add the **`alert`** code there!



You'll need

index.html

```
<p>You can use a script element to load JavaScript</p>
```

Testing

- [ ] Checking you created a `script` element.
- [ ] Checking you used the `src` attribute in the `script` tag.
- [ ] Checking you used the `src` attribute to reference `"cookies.js"`.
- [ ] Checking you made the alert say `'Cookies! My favourite!'`.

2.2. More interactive cookies

2.2.1. Hello, this is function

```
<script src="main.js"></script>
```

main.js

```
alert('Yo!');
```

Try this code in your browser to see the results

The `alert` in the example we showed is called a *function*.

Notice the parentheses `()` in `alert('Yo!')`? These tell JavaScript to *call* the function.

Calling a function just means to do what functions says. In this case, show a pop up with message we gave it!

2.2.2. More functions!

There are lots of different functions you can use.

Here are a couple of examples of some interesting functions you can use:

- `alert('Hello!')`
- `prompt('What is your name?')`
- `confirm('Do you want to continue?')`

Try substituting the functions into the code below:

main.js

```
// Paste in a function here on the next line!
```

```
<p>Try out functions on this page!</p>
<script src="main.js"></script>
```

Try out functions on this page!

2.2.3. Problem: Your feedback is valuable

Mega Cookie Corp wants us to create an online suggestions box.

We've already created a static web page, but to be interactive this page will need JavaScript too.

Mega Cookie Corp would like you to ask their customers the following question:

```
How can we serve you better?
```

And then they would like you to reassure the customer that they have been heard and are working to improve thanks to their suggestion by telling them:

```
You have been heard.
```

And then:

```
We will work to improve thanks to your suggestion.
```

> **Hint**
>
> You'll need to *call* three *functions* (one per line).
>
> Which functions out of `alert`, `prompt` and `confirm` do you think you'll need?

We don't actually do anything with the customer feedback for now, but don't worry. Mega Cookie Corp has heard you and will work to improve thanks to your suggestion!

You'll need

index.html

```
<h1>Mega Cookie Corp</h1>
<h2>Customer suggestions box</h2>
<script src="main.js"></script>
```

Testing

- [ ] Checking that you called the prompt function.
- [ ] Checking that your prompt asks `How can we serve you better?`
- [ ] Checking for the *first* `alert` function.
- [ ] Checking that the first `alert` says `You have been heard.`
- [ ] Checking you called `alert` function a *second* time.
- [ ] Checking the second `alert` says `We will work to improve thanks to your suggestion.`
- [ ] Testing that you didn't call any more functions.

2.3. Clicking cookies

2.3.1. Clicking the cookie

We know how to show a picture of a cookie but how do we know when someone has clicked it?

We're going to learn about JavaScript *events* which will tell us when our cookie is clicked!

2.3.2. What are events?

Events allow JavaScript programs to run code in response to something happening in the web browser. For example when you click a button, press a key, swipe an item in a list or when the web browser loads an image.

In JavaScript the name of an event starts with 'on'. For example, `onClick`, `onLoad` and `onKeyDown`.

Here's an example of an `onClick` event for a button.

```
<button onClick="alert('Ouch, that hurts!')">Click Me!</button>
```

Click Me!

Try making the button say something else when you click it.

2.3.3. Events in images

Events aren't just for buttons, though. We can make lots of things trigger events.

Here's an example where clicking on an image triggers an event.

```
<img src="macaron.png" onClick="alert('Om nom nom!')">
```

2.3.4. Problem: Taiyaki Tapper

It's finally time to make a cookie clicker. Well, kind of.

Due to a mixup at the dessert factory we're going to be using taiyaki (https://en.wikipedia.org/wiki/Taiyaki) instead of cookies! Taiyaki is a Japanese dessert which is a fish shaped waffle with a sweet filling, such as red bean.

The first time (and every time) you click the taiyaki it should say:

```
You clicked me for the first time!
```

The taiyaki should say the same thing every time afterwards too. Sadly, lots of fish don't have very good long term memories and this taiyaki is no exception.

Did I mention the taiyaki should say the same thing every time?

> **A function-ing fish**
>
> Use an `alert` function inside of the `img` element's `onClick` attribute to make the taiyaki say something when clicked.

You'll need

index.html

```
<h1>Taiyaki Tapper</h1>
<img src="taiyaki.png">
```

Testing

- [ ] Checking you have a taiyaki `img`.
- [ ] Checking you set the `onClick` attribute in the `img` element.
- [ ] Checking that `alert` was inside of the `onClick` attribute.
- [ ] Checking that you *called* the `alert` function when the taiyaki was clicked.
- [ ] Checking that you made the `alert` say `'You clicked me for the first time!'` when the taiyaki was clicked.

2.4. More events with cookies

2.4.1. Lots of kinds of events

There are lots of different kinds of events (https://developer.mozilla.org/en-US/docs/Web/Events).

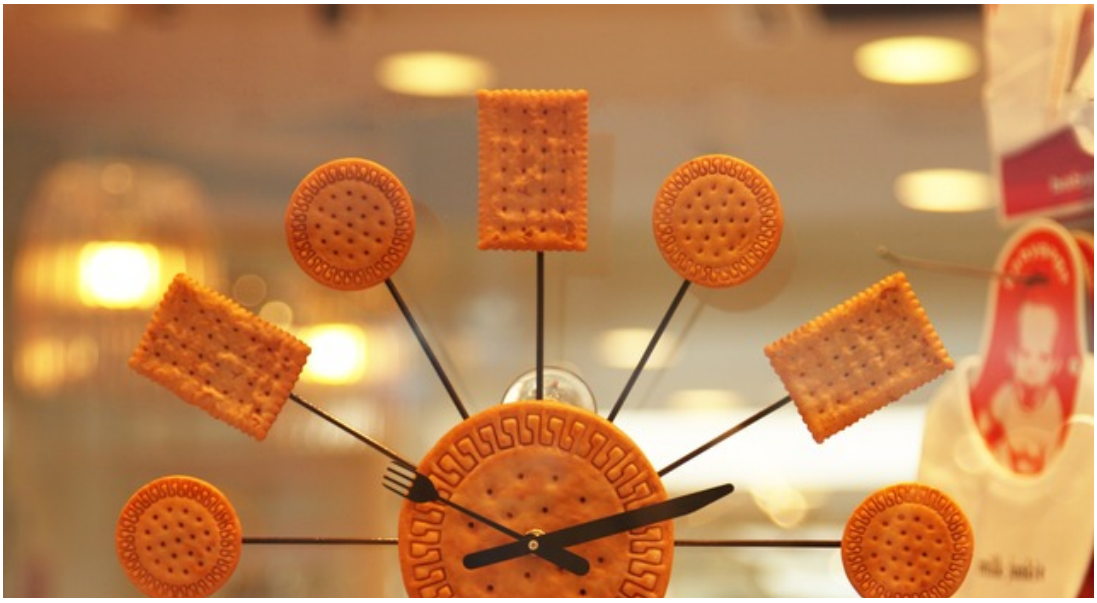
Here's a list of the ones that are the most interesting for an `img` element:

- `onClick`
- `onDblClick`
- `onContextMenu` this one is tricky!
- `onMouseOver`
- `onMouseLeave`
- `onWheel` hint: this wheel is often found on a mouse
- `onLoad`

Replace the `onClick` event below with the events from the list above.

See if you can figure out how to trigger the message for all of them. Some of them are pretty tricky!

```
<!-- Replace 'onClick' with an event from the list above -->
<img src="clock.jpg" onClick="alert('You triggered the event!')">
```



2.4.2. Confirm: Are you alert at this prompt?

Remember, there's more to JavaScript than alerts!

Try these different examples of events on buttons.

```
<button onClick="alert('I am alert!')">Alert Me!</button>
<button onClick="prompt('So prompt! Your thoughts?')">Prompt Me!</button>
<button onClick="confirm('Confirm this confirmation?')">Confirm Me!</button>
```

Alert Me! Prompt Me! Confirm Me!

Try making the button say something else when you click it.

2.4.3. Problem: Don't touch the bikkies!

It's ANZAC Day (https://en.wikipedia.org/wiki/Anzac_Day) and your Nan has just baked a fresh batch of ANZAC biscuits (https://en.wikipedia.org/wiki/Anzac_biscuit)! She's recruited you to make sure no one eats them until they've cooled.

Alert people about the hot bikkies when they first see the biscuits by using the **load** event:

```
Hey! Those bikkies are still hot!
```

And if they grab a biscuit using the **click** event you should pop up the following message to **confirm** they really want the cookie:

```
Are you sure? You might burn your mouth!
```

> **An eventful day**
>
> You can add as many event attributes to an element as you need!
>
> In this case you'll need the `onLoad` and `onClick` attributes.

You'll need

index.html

```
<img src="anzac-bikkies.png">
```

Testing

- [ ] Checking you have a gingerbread `img`.
- [ ] Checking you set the `onLoad` attribute in the `img` element.
- [ ] Checking that an `alert` happened when the gingerbread cookies were loaded.
- [ ] Checking that you made the `alert` say `'Hey! Those cookies are still hot!'` when the gingerbread cookies were loaded.
- [ ] Checking you set the `onClick` attribute in the `img` element.
- [ ] Checking that you `confirm`ed when the gingerbread cookies were clicked.
- [ ] Checking that you `confirm`ed by asking `'Are you sure? You might burn your mouth!'` when the gingerbread cookies were clicked.

2.5. Function-ing cookies

2.5.1. Doing more with events

If we want to do more than just one thing in an event we can *define* a *function*!

Just like calling the `alert` function we can call our own function. For example, the `why` function we've *defined* below.

main.js

```
function why() {
  prompt('Why did you click me?');
  alert('That makes sense.');
}
```

```
<button onClick="why()">Will you click Me?</button>
<script src="main.js"></script>
```

Try this code in your browser to see the results

To define a function, we write the word `function` then the name of the function, then parentheses `()` and then inside of the squiggly brackets `{}` we write the code we'd like to use.

2.5.2. You call that a function?

Let's go over using functions one more time.

Here's where we **define** the function:

main.js

```
function joke() {
  alert('Double Double?');
  alert('Click Click!');
}
```

Here's where we *call* it, - in this case if someone double clicks the button:

```
<button onDblClick="joke()">Double Click?</button>
<script src="main.js"></script>
```

Here's what the result is:

Try this code in your browser to see the results

2.5.3. Problem: This message will self-destruct...

Someone has been playing pranks down at the fortune cookie factory! They've made the cookies harder to open and they've stuffed them full of multiple messages! We need you to make a website to open these cookies remotely.

Since these cookies are so tough you'll need to use the **double click** event on the cookie image.

We've already written the `openCookie` *function* in `"fortune-cookie.js"` which contains the message. You just have to *call* the function when the `img` is **double clicked**.

> **Click click**
>
> The attribute for detecting a **double click** event is `onDblClick`.

You'll need

fortune-cookie.js

```
function openCookie() {
  alert('Your mission, if you choose to accept it is to open this fortune cookie.')
  confirm('Oh, you\'ve already done it?');
  alert('This message will self destruct in 3...');
  alert('2...');
  alert('1...');
  alert('CRUNCH!!!');
}
```

index.html

```
<img src="fortune-cookie.jpg">

<script src="fortune-cookie.js"></script>
```

Testing

- [ ] Checking you have a fortune cookie `img`.
- [ ] Checking you set the `onDblClick` attribute in the `img` element.
- [ ] Checking that `openCookie` was inside of the `onDblClick` attribute.
- [ ] Checking that the `openCookie` function was called.

2.6. Cookie Clicker - step 1

2.6.1. Problem: Project - step 1: Choc Click Cookie

Now that we know how to use events let's make our cookie clickable!

Create a `h1` heading with this text:

```
Click this cookie!
```

Then below the heading, have an `img` element with the `src`:

```
cookie.png
```

And when the image is **clicked** then use `alert` to show the following two messages:

```
Congratulations!
```

```
You clicked the cookie.
```

We've started the `cookieCongrats` function in `main.js` to get you started.

> **Here's my function, so call me on click**
>
> You can show *two* messages by *defining* your own function inside `"main.js"` and *calling* it inside of the `onClick` attribute.
>
> Make sure you use a `script` element with the `src` attribute so your web page knows where to find `"main.js"`.

You'll need

main.js

```
function cookieCongrats() {

}
```

Testing

- [ ] Checking there is a `h1` heading element with the correct text.
- [ ] Checking that there is an `img`with the correct `src` attribute.
- [ ] Checking your document visually against the expected image.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that an `alert` happened when the cookie was clicked.
- [ ] Checking that you made the `alert` say `'Congratulations!'` when the cookie was clicked.
- [ ] Checking that a *second* `alert` happened when the cookie was clicked.
- [ ] Checking that you made the *second* `alert` say `'You clicked the cookie.'` when the cookie was clicked.

SELECTING THE COOKIE

3.1. Grab the cookie

3.1.1. Grabbing the cookie

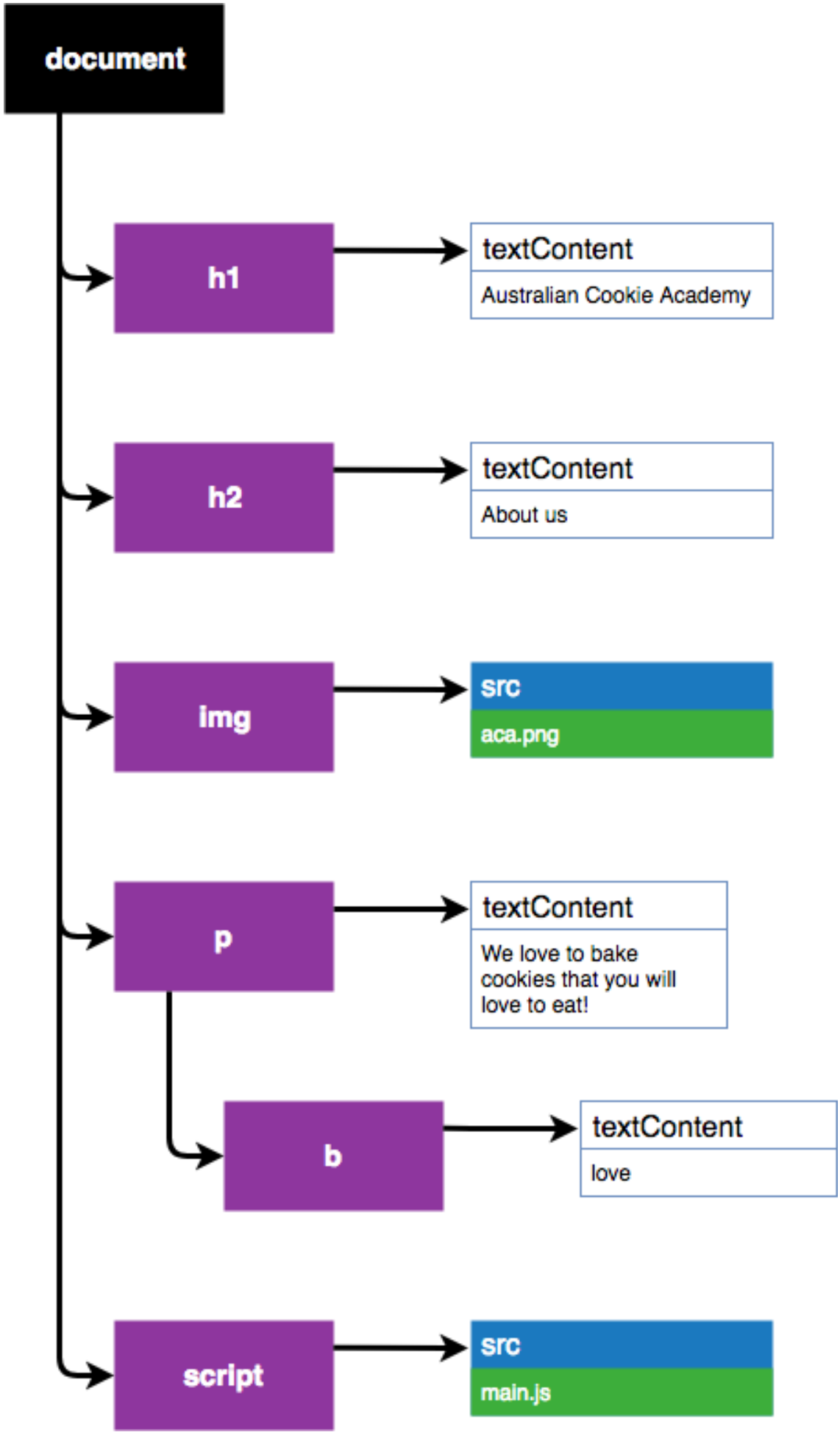
To make the cookie clicker we need to be able to change the elements on the page. For example update the number of cookies or make the cookie animate when it's being clicked.

To do this we need to be able to *select* the cookie from our web page. And to do that we need to learn more about JavaScript and a new concept called the *DOM*.

3.1.2. What is the DOM?

The DOM (Document Object Model) is an interface for JavaScript to interact with the HTML on our web page.

It mirrors the structure of the HTML on the page.



A diagram representing the DOM version of a web page

It lets us add, remove, read or change the elements on our webpage.

For example, you can use the DOM to change the `src` of the `img` to a different picture or add an element with new `textContent`.

3.1.3. Selecting tags

When interacting with the DOM with Javascript we use the `document` object.

To do anything with a DOM element we first have to *select* it. We can do that with `document.querySelector('tag')`.

Try selecting different elements of this web page by changing the name of the tag from `h1` to another tag name in the following JavaScript code.

```
<script src="highlight.js"></script>
<h1>Fancy Cookie Club</h1>
<h2>This week's cookie</h2>
<h3>The Macaron</h3>
<img src="macaron.png">
<script src="main.js"></script>
```

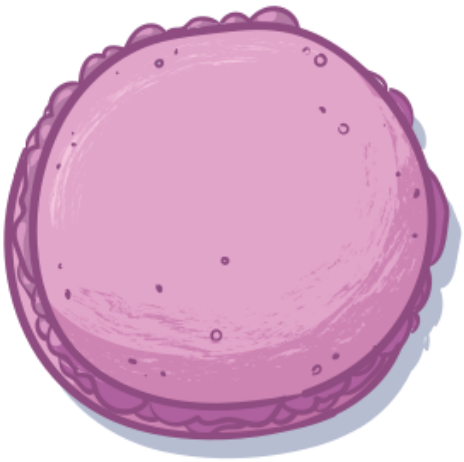
main.js

```
let heading = document.querySelector('h1');
```

Fancy Cookie Club

This week's cookie

The Macaron



3.1.4. Problem: DOMinostein

A Dominostein (https://en.wikipedia.org/wiki/Dominostein) is a German sweet named after domino tiles (https://en.wikipedia.org/wiki/Dominoes). It has many layers, one of which is called Lebkuchen (https://en.wikipedia.org/wiki/Lebkuchen), a type of cookie similar to gingerbread.

Dominsoteine with 3 layers

Since we're interested in cookies here at the Austrian Cookie Academy, please select the part of the Dominostein which is a type of cookie, the element which contains `lebkuchen`.

We'd also like you to put the element you selected into a variable called `cookie` so we can use it later.

> **How do we select?**
>
> Remember we can use `document.querySelector('')` with the name of the element inside of the quotes.
>
> You can use `let variableName = ... ;` to put them element into a variable.

You'll need

index.html

```
<script src="highlightSelection.js"></script>

<div>
  <i>marzipan</i>
  <u>sour cherry</u>
  <b>lebkuchen</b>
</div>

<script src="select.js"></script>
```

Testing

- [ ] Checking the `cookie` variable contains the lebkuchen layer.

3.1.5. Problem: Cookie-ing the books

Mega Cookie Corp are up to their old tricks again! They've hidden some not so savoury information in their web page by making it *small*!

Select the `small` element and store it in a variable called **`finePrint`** in the web page and expose Mega Cookie Corp's trickery!

> **More DOM selection**
>
> You'll need to use **`document.`**`querySelector` like in the previous problems!

You'll need

index.html

```
<script src="highlightSelection.js"></script>

<h1>Mega Cookie Corp</h1>

<p>
  These scrumptious cookies are lovingly handmade <small>by our robots</small> and ta
</p>

<p>
  <button>Buy a cookie</button>
</p>

<img src="cookies.jpg" onClick="alert('Thanks for purchasing :D Have a mega nice day!

<script src="select.js"></script>
```

Testing

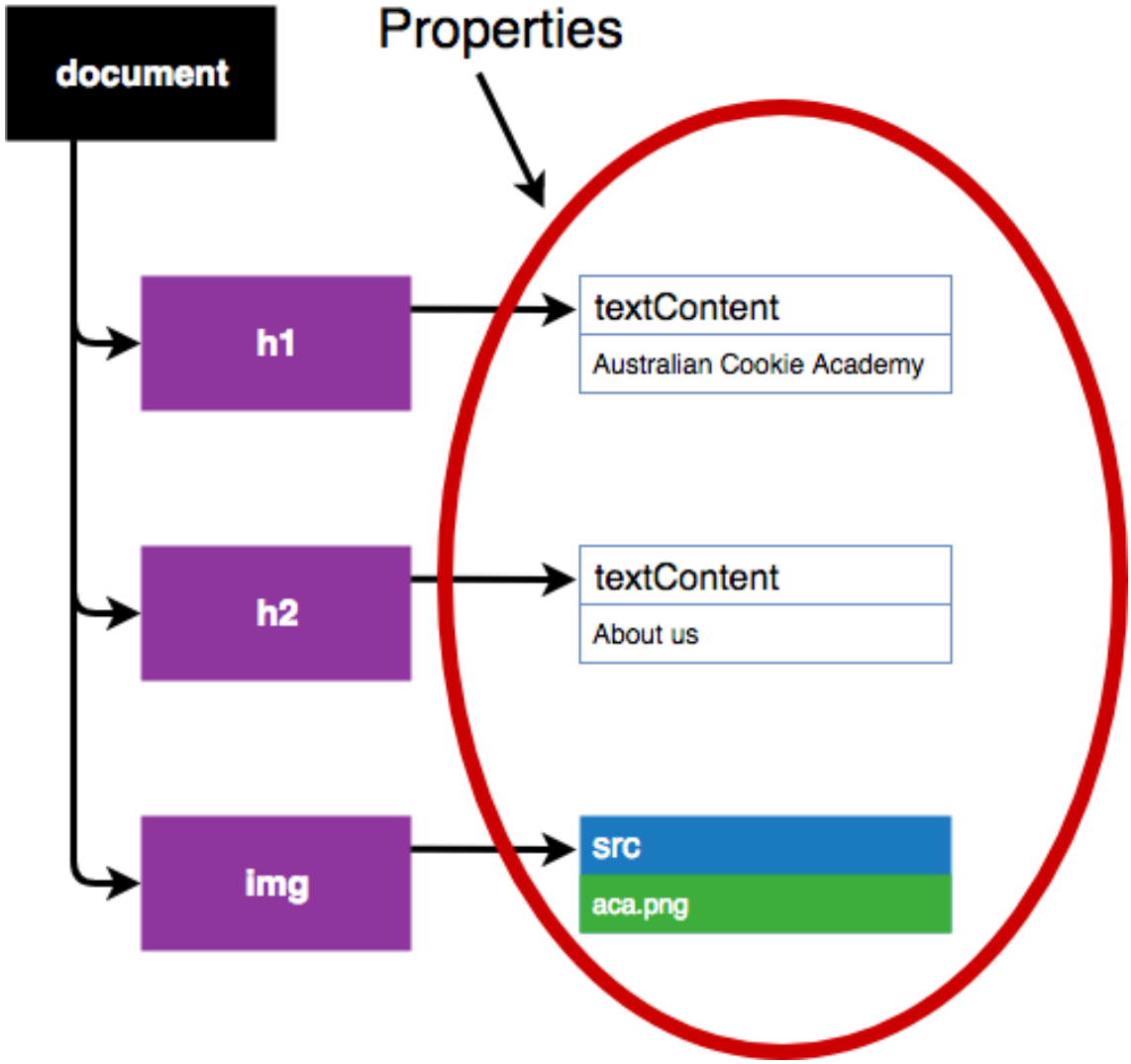
- [ ] Checking the **`finePrint`** variable contains the `small` element.

3.2. Change the cookie picture

3.2.1. Changing the cookie picture

Now that we know when the cookie has been clicked on we can do some really cool things! Like transforming the cookie into an even better cookie!

How do we do that? By using *properties* of the DOM elements we've selected in the previous problems.



A diagram showing the `textContent` and `src` as examples of DOM properties

3.2.2. The src property

Since we know how to select DOM elements we can now start do things with them!

We can use the `src` *property* of the DOM object. Of course, we have to make sure it's an `img` element. If we use another kind of element, nothing will happen!

```
<h1>Who ate my scotch fingers?</h1>
<img src="scotch-fingers.png">
<script src="main.js"></script>
```

main.js

```
let image = document.querySelector('img');
image.src = 'crumbs.png';
```

Who ate my scotch fingers?

Notice the `.src` after our `img` selection? This is how we access the `src` *property* of the DOM object. Changing the value of this property changes which image our `img` tag uses.

3.2.3. Problem: Chocolate src

The cookies we've baked are rather plain. Let's make them shine by drizzling some chocolate on them!

First you'll need to select the image with `document.querySelector`. Then once you've done that you'll need to change the image's `src` to the better and more chocolate covered picture!

This delicious chocolate covered cookie is called `"butter-cookie-choc.png"`

Changing the image source

Once you have a variable which contains an `img` element you can *assign* a different value to the variable's `.src` attribute.

You'll need

index.html

```
<img src="butter-cookie.png">

<script src="main.js"></script>
```

Testing

- [ ] Checking you selected the `img` element.

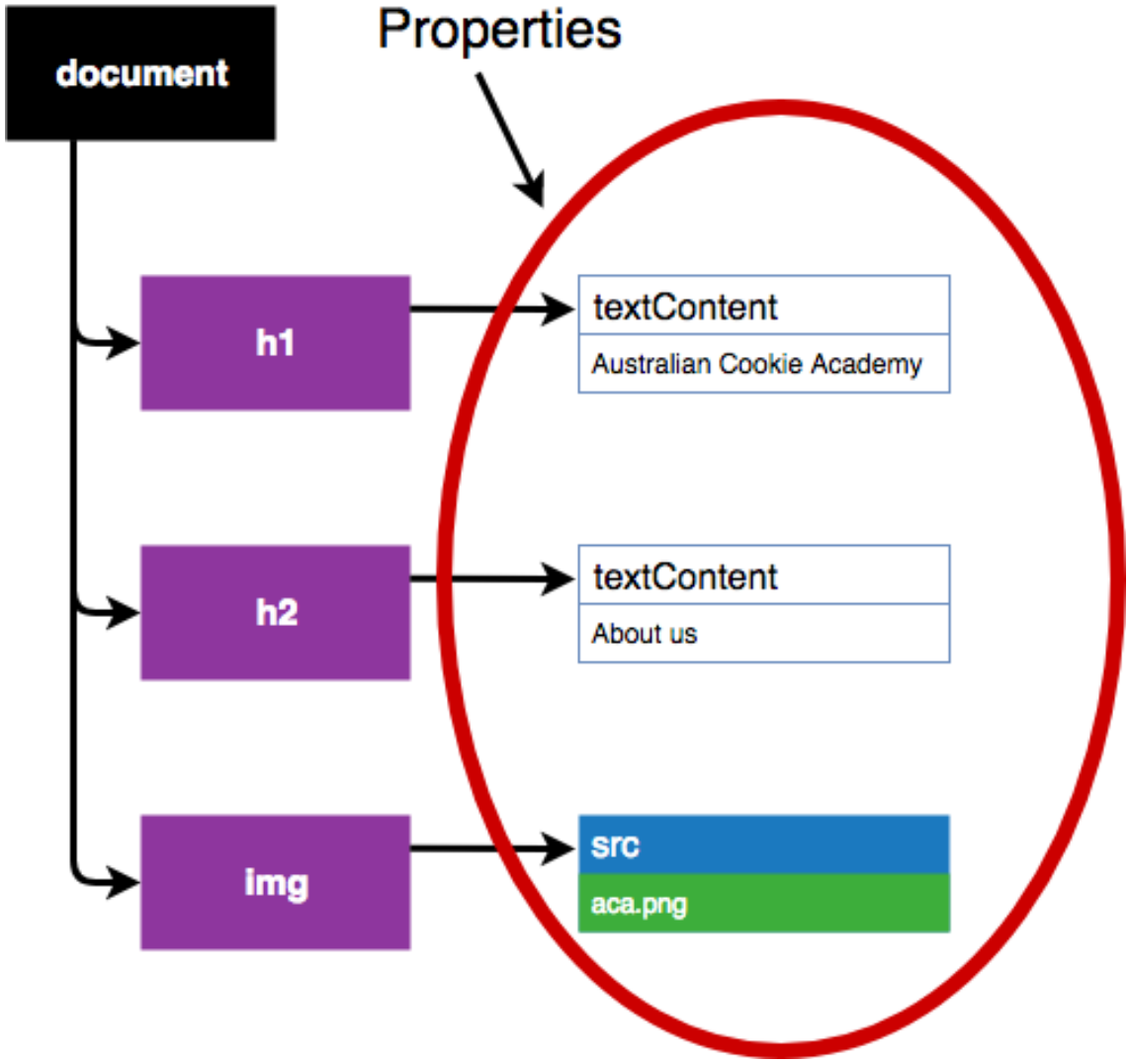
- [ ] Checking that you changed the image's `src` attribute.
- [ ] Checking that you changed the image's `src` attribute to `"butter-coookie-choc.png"`.

3.3. Change the cookie heading

3.3.1. Changing the cookie heading

The other thing we might want to do in our cookie game is change the heading.

Just like with the image we can change the heading by using a *property* of a DOM element.



A diagram showing the `textContent` and `src` as examples of DOM properties

3.3.2. The textContent property

Another thing we can do with DOM elements is change their text.

```
<h1>Fancy Cookie Club</h1>
<h2>This week's cookie</h2>
<h3>A boring cookie</h3>
<img src="choc-chip.png">
<script src="main.js"></script>
```

main.js

```
let heading = document.querySelector('h3');
heading.textContent = 'A classic quintessential choc-chip cookie!';
```

Fancy Cookie Club

This week's cookie

A classic quintessential choc-chip cookie!

Notice the `.textContent` after our heading selection? This is how we access the `textContent` *property* of the DOM object.

It doesn't matter which object you select, as long as it has text then you can change it!

Try this

Can you think of a different description for the cookie shown?

How would you change the title of the cookie club?

3.3.3. Problem: A sometimes food?

The Cookie Monster thinks cookies are a sometimes food! This is disastrous for Mega Cookie Corp's profits!

Mega Cookie Corp has managed to get access to the JavaScript on Cookie Monster's blog. Use JS to change the article's title from **cookies are a sometimes food** to:

```
C is for Cookie. That is good enough for me.
```

> **Changing text**
>
> Once you've selected an element you can change its content by assigning a different string to the element's `textContent`.

You'll need

index.html

```
<h1><img src="cookie.png">Cookie Monster's Blog</h1>
<article>
  <h2>Cookies are a sometimes food</h2>
  <p>Om nom nom nom nom!</p>
</article>

<script src="main.js"></script>
```

Testing

- [ ] Checking that you selected the blog's title element.
- [ ] Checking that you changed the text of the blog's title.
- [ ] Checking that you changed the blog title to `C is for Cookie. That is good enough for me.`

3.4. Click to change the cookie

3.4.1. Changing the cookie on click

We've learnt how to do something when we click the cookie and how to change the cookie picture and text.

So far we've only used DOM properties and events to do these things separately. But they can be combined so that a user can change the web page just by clicking or moving their mouse!

3.4.2. Events and DOM properties

Here's an example of how you can use DOM properties and events together:

```
<h1>This is my almond cookie.</h1>
<h2>It's meant for Chinese New Year!</h2>
<img src="chinese-almond.png"
  onMouseEnter="warning()"
  onMouseDown="eat()"
  onMouseUp="busted()">
<script src="main.js"></script>
```

main.js

```
let heading = document.querySelector('h1');
let image = document.querySelector('img');

function warning() {
  heading.textContent = 'Don\'t eat my almond cookie.';
}

function eat() {
  image.src = 'crumbs.png';
}

function busted() {
  heading.textContent = 'You ate my almond cookie!';
}
```

This is my almond cookie.

It's meant for Chinese New Year!

3.4.3. Problem: Double choc click cookie

Cookie Monster has forgiven you for changing his blog using JavaScript. But now you are friends you would like to use JS to make the cookie on his blog edible!

When the cookie is **clicked** change the cookie picture to:

> `cookie-bite.png`

When the cookie is **double clicked** change the picture to:

> `crumbs.png`

And also when the cookie is **double clicked** change the blog's subheading (the `h2` element) to:

> `Om nom nom nom!`

> **O is for onClick**
>
> In case you've forgotten, `onClick` and `onDblClick` are the names of the event attributes you'll need for this problem.

You'll need

index.html

```
<h1>
  <img src="cookie-full.png">
  Cookie Monster's Blog
</h1>
<h2>Welcome to me blog!</h2>
<article>
  <h3>Cookie friends</h3>
  <p>Sometimes me think, "What is friend?" and then me say, "Friend is someone to sha
</article>

<script src="main.js"></script>
```

Testing

- [ ] Checking there is a `h1` heading element with the correct text.
- [ ] Checking there is a `h2` heading element with the correct text.
- [ ] Checking that there is an `img` with the correct `src` attribute.
- [ ] Checking your document visually against the expected image.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `img` element's `src` changed when it was clicked.
- [ ] Checking that the `h1` element's text did **not** change when the cookie was double clicked.
- [ ] Checking that the `h2` element's `textContent` changed when the cookie was clicked.
- [ ] Checking that the `img` element's `src` changed when it was clicked.

3.5. Cookie Clicker - step 2

3.5.1. Problem: Project - step 2: Awww crumbs

We now know how to change elements on our page using the DOM. So let's use that knowledge to make our game more interactive!

We'll start off with the page we had for the 'No cookies' problem and then update the page when the cookie is clicked.

Start with a `h1` heading with this title:

```
No Cookies
```

Then below that have an image element with this picture:

```
crumbs.png
```

> **Be lazy!**
>
> You can get a quick start on this problem by copying your solution to the 'No cookies' problem!

And when the image is clicked...

Change to this heading:

```
You clicked and got a cookie!
```

And this image:

```
cookie.png
```

> **Write the script**
>
> You can change the image `src` inside of a function you define in the `"main.js"` file and *calling* it inside of the `onClick` attribute.
>
> Make sure you use a `script` element with the `src` attribute so your web page knows where to find `"main.js"`.

Testing

- [ ] Checking there is a `h1` heading element with the correct text.
- [ ] Checking that there is an `img` with the correct `src` attribute.
- [ ] Checking your document visually against the expected image.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `h1` element's `textContent` changed when the cookie was clicked.
- [ ] Checking that the `img` element's `src` changed when it was clicked.

COUNTING COOKIES

4.1. Input cookies

4.1.1. Variables

We've used *variables* to store DOM elements in previous modules. But what *are* variables?

Variables allow us to label different kinds of **data** and read or modify that data. For example, selecting DOM elements and changing their `src`.

We can also use variables to accept *user input*. Like this example where we store a `prompt` inside of the `bestCookie` variable:

```
let bestCookie = prompt('What is your favourite cookie?');
```

We'll see in the next few slides how to use that variable.

4.1.2. Using variables

Here's an example of asking for some data, putting it in a variable and then using it to update a web page.

```
<p>Click this <span>cookie</span></p>
<img src="biscotti.png" onClick="changeName()">
<script src="main.js"></script>
```

main.js

```
let clickThis = document.querySelector('span');

function changeName() {
  let answer = prompt('This isn\'t a cookie. But it is a...');
  clickThis.textContent = answer;
}
```

Try this code in your browser to see the results

4.1.3. Problem: Fortune Cookie

Mega Cookie Corp is ramping up production on fortune cookies. They need to create a website to see what the fortunes look like in a cookie.

You already have a web page with a place to put fortunes (the `h2` element) and a **`prompt`** that asks the user for the fortune to display.

All you need to do is set the `textContent` of the `h2` element to be the fortune that the user gave as an answer to the **`prompt`**.

> **Hint - break it down**
>
> Let's look at this step by step.
>
> - Find out what the new fortune is (this is done for you!)
> - Select the **`h2`** element, and save it in a variable.
> - Update the contents of that variable, and set it to be the words that are read in and saved as **`fortune`**!

You'll need

main.js

```
let fortune = prompt('Enter a fortune:');
```

index.html

```
<link rel="stylesheet" type="text/css" href="main.css">

<h1>Your Mega Cookie Corp Fortune<sup>TM</sup> is:</h1>
<img src="fortune.png">
<h2></h2>

<script src="main.js"></script>
```

main.css

```
h2 {
  position: relative;
  left: 250px;
  top: -210px;
  width: 320px;
  font-size: 16px;
}
```

Testing

- ☐ Checking that there is are `h1`, `h2` and `img`.
- ☐ Checking that the `h2` heading element has correct text when a user inputs `You will pass this test`.
- ☐ Checking that the `h2` heading element has correct text when a user inputs `If you click you will get a cookie`.
- ☐ Checking that the `h2` heading element has correct text when a user inputs a fortune that we've hidden.

4.1.4. Problem: Pick a cookie

The Fancy Cookie Club has decided that the cookie of the week is "Your favourite cookie". They don't know everyone's favourite cookie though.

Help the FCC make a website that asks the user for their favourite cookie and then updates the cookie of the week page with their cookie.

We've already provided you with code that prompts the user and updates the image.

You should update the `h3` subheading element's `textContent` with the name of the cookie.

And also use the `onError` event to of the `img` to change the image of any missing cookie to:

```
plate.png
```

> **Names of cookies**
>
> Here's a list of the cookie names you can enter to get a picture:
>
> - anzac
> - biscotti
> - butter-cookie-choc
> - butter-cookie
> - chinese-almond
> - choc-chip
> - choc-digestive
> - choc-pretzel
> - fortune
> - gingerbread
> - macaron
> - raspberry-ice
> - scotch-finger
> - strawberry-sugar
> - stroopwafel
> - taiyaki

You'll need

main.js

```
let img = document.querySelector('img');
let heading = document.querySelector('h3');

let bestCookie = prompt('What\'s your favourite cookie?');

img.src = bestCookie + '.png';
```

index.html

```
<h1>Fancy Cookie Club</h1>
<h2>Cookie of the week</h2>
<h3>Your favourite</h3>
<img src="plate.png">

<script src="main.js"></script>
```

Testing

- [ ] Checking that there is are `h1`, `h2` and `h3` heading elements and an `img`.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `img` has the correct `src` when a user inputs `anzac`.
- [ ] Checking that the `h3` heading element has correct text when a user inputs `anzac`.
- [ ] Checking that the `img` has the correct `src` when a user inputs `choc-pretzel`.
- [ ] Checking that the `h3` heading element has correct text when a user inputs `choc-pretzel`.
- [ ] Checking that the `img` has the correct `src` when a user inputs a cookie with no picture.
- [ ] Checking that the `h3` heading element has correct text when a user inputs `oatmeal-raisin`.

4.2. Numbers of cookies

4.2.1. Numbers

So far the variables we've used have been `DOM elements` and `strings`. But what about **numbers**?

You can get a number from a user by using the following JS code:

```
let cookies = parseInt(prompt('How many cookies are in the cookie jar?'));
```

The **parseInt** function turns the answer from a `string` to a **number** (*integer*). We'll see why this is important later.

4.2.2. Adding numbers

You can never have enough cookies! This example uses the + *operator* to reveal how many more cookies we really wish we had.

```
<p>I have <i>0</i> cookies but I wish I had <b>2</b> cookies</p>
<script src="main.js"></script>
```

main.js

```
let cookies = parseInt(prompt('How many cookies do you have?'));

let have = document.querySelector('i');
have.textContent = cookies;

let want = document.querySelector('b');
want.textContent = cookies+2;
```

Try this code in your browser to see the results

4.2.3. When is a number not a number?

You'll notice if you remove **parseInt** from the previous slide's example that you get some funny behaviour!

You'll see something like:

```
I have 3 cookies but I wish I had 32 cookies.
```

instead of

```
I have 3 cookies but I wish I had 5 cookies.
```

This is because a number is not a number if it is a `string`. JavaScript can't know whether the user entered text or a number, so you have to tell it to expect a number by using **parseInt**.

So when you don't use **parseInt** JS will just add the number `2` on the end of the string `1`. Just like you might add the letter `'s'` on the end of a word to pluralise it.

4.2.4. Problem: Stroopwafel+3

Australian Cookie Academy has started selling Stroopwafels (https://en.wikipedia.org/wiki/Stroopwafel). Everybody loves these thin, crispy and caramelly waffles. The trouble is they always end up heart broken because they buy too few!

A heart breaking Stroopwafel order

The ACA really wants its customers to be satisfied so they want you to update the website to add 3 Stroopwafels to the customer's order.

The ordering system is mostly set up. You have to:

- find the place where the number ordered is being read in, and make sure to use `parseInt` to make it a number; and
- find the right place to add the extra three Stroopwafels to the customer's order.

You'll need

main.js

```
let stroopwafels = prompt('How many Stroopwafels would you like?');

let bought = document.querySelector('i');
bought.textContent = stroopwafels;

let received = document.querySelector('b');
received.textContent = stroopwafels;
```

index.html

```
<h1>Australian Cookie Academy</h1>
<h2>Stroopwafel store</h2>
<p>You have bought <i>0</i> Stroopwafels</p>
<p>But you will receive <b>0</b> Stroopwafels!</p>
<img src="stroopwafel.png">

<script src="main.js"></script>
```

Testing

- [ ] Checking that there is are `h1` and `h2` heading elements, `i` and `b` elements and an `img`.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `i` has the correct text when a user inputs `2`.
- [ ] Checking that the `b` has the correct text when a user inputs `2`.
- [ ] Checking that the `i` has the correct text when a user inputs `15`.
- [ ] Checking that the `b` has the correct text when a user inputs `15`.
- [ ] Checking that the `i` has the correct text when a user inputs a hidden number of stroopwafels.
- [ ] Checking that the `b` has the correct text when a user inputs a hidden number of stroopwafels.

4.2.5. Problem: Sharing is caring

Fancy Cookie Club is having an inter-club meetup event with branches of the Basic Biscuit Bunch all over the world! To help with their meetings they need a website to count how many cookies and biscuits will be at a meetup.

Write JS code to first `prompt` `'How many cookies will the FCC bring?'` and then `prompt` `'How many biscuits will the B3 bring?'`.

Then put the number of cookies into the `i` tag, the number of biscuits into the `b` tag and the number of cookies+biscuits into the `span` tag.

> **Adding variables**
>
> Adding two variables together is just like adding a number to a variable. Just put a `+` sign between them!
>
> Make sure that both variables are numbers though. Otherwise you might not get the answers you expected!

You'll need

index.html

```
<h1>FCC and B3 meetup</h1>
<h2>There will be <span>0</span> biscuits and cookies</h2>
<p>The Fancy Cookie Club will bring <i>0</i> cookies</p>
<p>The Basic Biscuit Bunch will bring <b>0</b> biscuits</p>

<script src="main.js"></script>
```

Testing

- [ ] Checking that there is are `h1` and `h2` heading elements, `i`, `b` and `span` elements.
- [ ] Checking that you have exactly two `prompt`s.
- [ ] Checking that you have the cookie `prompt` text correct.
- [ ] Checking that you have the biscuit `prompt` text correct.
- [ ] Checking that the `i` has the correct text when a user inputs `2` cookies.
- [ ] Checking that the `b` has the correct text when a user inputs `3` biscuits.
- [ ] Checking that the `span` has the correct text when a user inputs `3` biscuits.
- [ ] Checking that the page has the correct text when a user inputs a hidden number of biscuits and cookies.
- [ ] Checking that the page has the correct text when a user inputs different hidden number of biscuits and cookies.
- [ ] Checking that the page has the correct text when a user inputs another set of hidden number of biscuits and cookies.

4.3. Counting clicks

4.3.1. Counting - just another way to add 1

We've seen how to use numbers, and how to add. Now we can combine those two elements to count!

You can add one in JavaScript by setting a variable, and then adding one to it:

```
let cookieCount = 0;
cookieCount = cookieCount + 1
```

That's a lot of typing just to add one (or *increment*) to a value, though! Don't worry, there's a shorter way!

The first line sets up a variable called `cookieCount` and sets it to value `0`.

The second line updates the value of `cookieCount` to whatever it was (in this case, `0`) plus one more!

The shorthand to **add one** to a variable is using `++` like this:

```
let cookieCount = 0;
cookieCount++
```

That means the same thing!

4.3.2. Counting events

So now that we know how to count and do things when events happen, how do we count a number of events?

It's easy! We just put the counting code (the line with the `++`) inside of the event function!

```
<p>Move the mouse over the cookie to poke it</p>
<img onMouseOver="poke()" src="cookie.png">
<p>You have poked the cookie <span>0</span> times</p>
<script src="main.js"></script>
```

main.js

```
let pokes = 0;
function poke() {
  pokes++;
  let counter = document.querySelector('span');
  counter.textContent = pokes;
}
```

Move the mouse over the cookie to poke it

You have poked the cookie 0 times

4.3.3. Problem: SnackChat

SnackChat is an app that allows you to take a photo and send it to a random friend just by tapping on the photo.

SnackChat is the tastiest new app in the app store!

SnackChat wants to release a new feature that allows you to send your photo to a random friend **every single time** you tap the photo.

They need you to count and update the `span` element with the number of friends you've sent a Snack.

Don't worry about writing the function to send a photo to a random friend every time you tap. SnackChat will add that functionality once you're done.

> **Hint**
>
> A good first step is to set up a variable to count how many friends you've sent Snacks to. It will be 0 to start with!
>
> Remember to define a variable *outside* of the count function!

You'll need

count.js

```
function count() {
  alert('Snap sent to a random friend');
}
```

index.html

```
<h1>SnackChat</h1>
<p><span>0</span> friends Snacked</p>
<img src="snap.jpg" onClick="count()">

<script src="count.js"></script>
```

Testing

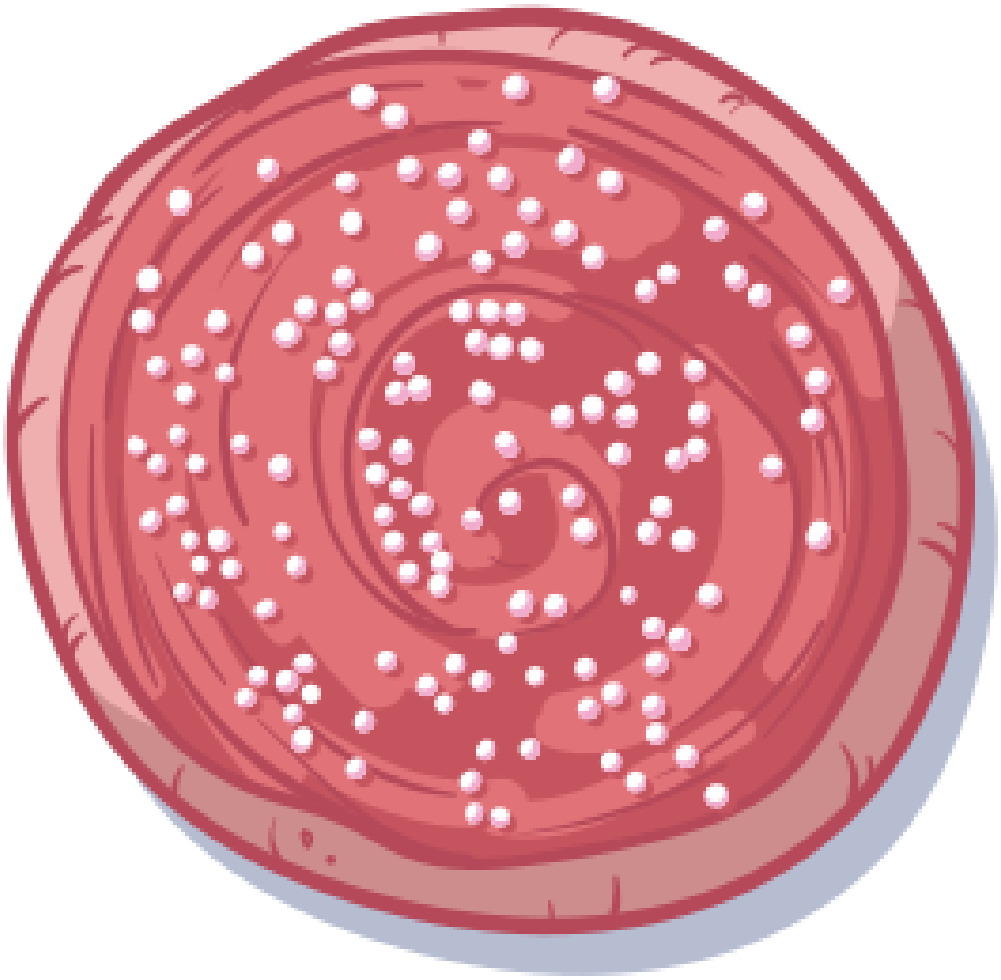
- [ ] Checking there is a `p` element with the correct text.
- [ ] Checking there is a `span` element with the correct text.

- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `span` element's `textContent` changed when the image was clicked.
- [ ] Checking that the `p` element's `textContent` changed when the image was clicked.
- [ ] Checking that the `span` element's `textContent` changed when the image was clicked many times.
- [ ] Checking that the `p` element's `textContent` changed when the image was clicked many times.

4.3.4. Problem: Hundreds and Thousands

Mega Cookie Corp is conducting the annual quality control inspection of their strawberry sugar cookies. To ~~avoid taxes~~ allow employees more flexibility, Mega Cookie Corp would like to ~~fire their full time employees~~ move their counting process into the sharing economy.

Mega Cookie Corp will crowd source local freelancers to **click** on each of the sprinkles to count them and ensure number of sprinkles is within acceptable limits.



A standard MCC cookie should have no more than 120 sprinkles.

Write some JavaScript code that defines a `count()` function that updates the `span` tag with the number of sprinkles counted so far.

You'll need

index.html

```
<h1>Sprinkle Quality Assurance</h1>
<p><span>0</span> sprinkles counted</p>
<img src="strawberry-sugar.png" onClick="count()">

<script src="count.js"></script>
```

Testing

- [ ] Checking there is a `p` element with the correct text.
- [ ] Checking there is a `span` element with the correct text.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `span` element's `textContent` changed when the image was clicked.
- [ ] Checking that the `p` element's `textContent` changed when the image was clicked.
- [ ] Checking that the `span` element's `textContent` changed when the image was clicked many times.
- [ ] Checking that the `p` element's `textContent` changed when the image was clicked many times.

4.4. Cookie Clicker - step 3

4.4.1. Problem: Complete Cookie Clicker

We now know how to change elements on our page using the DOM. So let's use that knowledge to make our game more interactive!

We'll start off with the page we has for the 'No cookies' problem and then update the page when the cookie is clicked.

Start with a `h1` heading with this content:

```
<i>No</i> Cookies
```

Then below that have an image element with this picture:

```
crumbs.png
```

And when the image is clicked...

Change the to this image:

```
cookie.png
```

And change the `i` element to the number of cookies.

For example, the heading should contain this content after 1 click:

```
<i>1</i> Cookies
```

And be like this after 23 clicks:

```
<i>23</i> Cookies
```

> **Be lazy!**
> You can get a quick start on this problem by copying your solution to the 'Awwww crumbs' problem!

Testing

- [ ] Checking there is a `h1` heading element with the correct text.
- [ ] Checking there is a `i` element with the correct text.
- [ ] Checking that there is an `img`with the correct `src` attribute.
- [ ] Checking your document visually against the expected image.
- [ ] Checking you used the `script` element with the correct `src` attribute.
- [ ] Checking that the `img` element's `src` changed when it was clicked.
- [ ] Checking that the `h1` element's `textContent` changed when the cookie was clicked.
- [ ] Checking that the `h1` element's `textContent` changed when the cookie was clicked many times.

5

EXTENSIONS: DECORATING THE COOKIE

5.1. Cookie extensions

5.1.1. Warmer, Chewier, Tastier

This last module shows you how to make your cookie clicker even cooler!

You can use the playground on the next slide to combine all of the extensions or even try out some ideas of your own!

5.1.2. Problem: Cookie Clicker Playground

Create your ultimate cookie clicker here!

Testing

- [ ] **This is a playground question. There is no right or wrong!**

5.2. More cookie heading changes

5.2.1. What is a string template?

At the moment our cookie clicker always says we have `'No cookies'` no matter how much we click! We know how to change the text with JavaScript, but how can we have a number that changes? String templates!

A string template looks like something like this:

```
`Hello, my favourite cookie is ${cookie}.`
```

Notice that the quotes used are not single quotes like a normal string, but use *backticks* (`) which can usually be found next to the **1** key on your keyboard.

The value inside of `${}` is a variable that will be put inside of the string template.

5.2.2. Using a string template

How do we actually use a variable inside of a string template? Well first we need to create a variable, then we can use it in a string template.

Take a look at this example:

```
<p>I don't have any cookies yet</p>
<script src="main.js"></script>
```

main.js

```
let cookies = prompt('How many cookies would you like?');
let eat = prompt('How many cookies will you eat now?');
let counter = document.querySelector('p');
counter.textContent = `I have ${cookies} cookies.
And I will eat ${eat} of the ${cookies} cookies right now.`;
```

Try this code in your browser to see the results

5.2.3. Problem: A recipe for disaster!

A long lost ANZAC biscuit recipe has been found in the Australian Cookie Academy's archives. Unfortunately several crucial details at in the recipe have faded or have cookie dough stuck to them!

ACA wants you to help them by making a website to crowdsource what the missing details could be.

Here is the recipe with underscores replacing the unknown text:

```
________ ANZAC Biscuit Recipe

Ingredients:
 * 1 cup oats
 * 3/4 cup flaked coconut
 * 1 cup flour
 * 1 ________ baking soda
 * 1 cup brown sugar
 * 1/2 cup butter
 * 1 tablespoon golden syrup

Steps:
  1. Mix oats, flour, sugar and coconut together.

  2. In a small saucepan over low heat, ________ the syrup and butter together. Mix t

  3. Add butter mixture to the dry ingredients. Roll into balls, place on baking pape

  4. Bake at 175 degrees Celcius for 18 to 20 minutes.
```

The ACA has already created `prompts` for each crowd member to fill in the blanks. You just need to take those answers and fill in the blanks and then put the text inside of the `pre` element.

> **Filling in the blanks**
>
> You can put variables inside a string using *string templates*.
>
> Just copy and paste the recipe above into your program and add quotes around it. Make sure you replace the underscores with the template variables!
>
> Don't forget that string templates use backticks `` instead of quotes `''`!

You'll need

main.js

```
let adjective = prompt('Adjective:');
let measurement = prompt('Measurement:');
let verb = prompt('Verb:');
let utensil = prompt('Utensil:');
```

index.html

```
<h1>The long lost ANZAC recipe</h1>
<img src="anzac.png">
<pre></pre>
<script src="main.js"></script>
```

Testing

- [ ] Checking that you have all 4 prompts.

- [ ] Checking that you added text to the `pre` element.
- [ ] Checking the recipe with placeholder responses.
- [ ] Checking the recipe with real responses.

5.3. Cookie decoration

5.3.1. CSS

You may remember at the start of this course we mentioned CSS as one of the three languages making up a website. CSS (Cascading Style Sheets) are used to make web pages look good (or bad if you would like).

To add CSS to a page we add a `<link>` tag and a CSS file. The CSS file itself contains *CSS selectors* that say which element is being styled. And *CSS properties* which state what the page should look like.

For example, this page styles a heading:

```
<link rel="stylesheet" type="text/css" href="main.css">
<h1>Choc-Orange Cookies</h1>
```

main.css

```
h1 {
  color: orange;
  font-family: Tahoma, sans-serif;
}
```

As you can see in the example

- `h1` is an example of a *selector*
- We put opening (`{`) and closing (`}`) brackets, the properties and their values go in here
- `color`, `font-family` etc are example of *properties*
- After each attribute we put a colon (`:`)
- Then the *value* for each attribute
- At the end of each line we put a semi-colon (`;`)

5.3.2. More CSS properties

There are **a lot** of things you can do with CSS. Here is a list of all of the CSS properties (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference) and a list of colour keywords (https://developer.mozilla.org/en-US/docs/Web/CSS/color_value#Color_keywords).

Here's an example that uses some of them:

```
<link rel="stylesheet" type="text/css" href="main.css">
<h1>My stylish cookie</h1>
<img src="stylish-cookie.jpg">
```

main.css

```css
body {
  width: 600px;
  background-color: rebeccapurple;
  text-align: center;
}
h1 {
  color: white;
  font-family: Tahoma, sans-serif;
  text-shadow: 2px 2px sienna;
}
img {
  height: 120px;
  border: 6px dashed thistle;
  border-radius: 120px;
}
```

Styling the page itself

You can use the **body** selector to style the whole page, including the background.

5.3.3. Problem: Stylish Pretzel

Mega Cookie Corp is releasing a new product line: Loaded Pretzels™ They have a web page, but it doesn't match the fun, treat yourself nature of the Loaded Pretzels™ brand.

Mega Cookie Corp wants the marketing they've created to be clear, so first you need to:

- Center all of the text and the image on the page (using the `body` selector and the `text-align` property)
- Make the paragraph (`p` element) text bigger

Unfortunately Mega Cookie Corp doesn't know how to measure fun. So you can style the page however you would like! Just make sure you:

- Colour the background (using the `body` selector)
- Change the font of the headings and the paragraph (using this property/value `font-family: Comic Sans MS;` – you can use any font's name as a value)
- Colour the main heading (`h1` element)
- Colour the subheading (`h2` element) a different colour to the main heading

You'll need

index.html

```
<!-- This element helps display the ™ character correctly. It's very important to Meg
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<link rel="stylesheet" type="text/css" href="main.css">

<h1>Loaded Pretzels™</h1>
<h2>Treat yourself!</h2>
<img src="loaded-pretzel.png">
<p>
  A snack that has it all!
  Salt, sweet, crunch and chew.
  Treat yourself and load our Loaded Pretzels™ into your face!
</p>
```

Testing

- [ ] Checking that you centered the text.
- [ ] Checking that you made the `p` text bigger.
- [ ] Checking that you changed the background colour.
- [ ] Checking that you changed the font family of the headings and paragraph text.
- [ ] Checking that you changed the `h1` element's colour.
- [ ] Checking that you changed the `h2` element's colour.

5.4. Cookie decorating events

5.4.1. Styling DOM

We already know how to add style with CSS. But what if we want the style to change during an event? We have to use JavaScript and `element.style` to modify the style of a DOM element.

For example, on this web page we can change the colour of the text based on the user's input. You'll have to make sure the colour you answer with is a valid CSS colour! (https://developer.mozilla.org/en-US/docs/Web/CSS/color_value#Color_keywords)

```
<h1>Colourful Cookie</h1>
<script src="main.js"></script>
```

main.js

```
let heading = document.querySelector('h1');
let colour = prompt('What colour is your cookie?');
heading.style.color = colour;
```

Try this code in your browser to see the results

The `.style` gets the style of the DOM element we selected (a h1 tag in our case). The `.color` gets the colour of the text in the h1 tag. Then using the equals sign (=) we assign the colour to the heading. It's just like `.textContent` or `.src` properties!.

American Spelling

Notice how colour is spelled without the letter 'u' sometimes? JavaScript uses American spelling for all of its names, for example, the words color (colour) or capaitilize (capitalise).

But the variables contain a 'u' because we got to choose their names and we're Australian!

5.4.2. A stylish event

In addition to styling the page using `prompt`s you can use events to change the style of your page.

For example, we can use an `onClick` event to style a page's background:

```
<h1>Pick one!</h1>
<img src="blue-macaron.png" onClick="blue()">
<img src="green-macaron.png" onClick="green()">
<script src="main.js"></script>
```

main.js

```
let page = document.querySelector('body');

function blue() {
  page.style.backgroundImage = 'url(blue-macaron.png)';
}

function green() {
  page.style.backgroundImage = 'url(green-macaron.png)';
}
```

Pick one!

So much style!

As you saw above it's not just colour that you can change the style of. There's a whole list of things you can change (https://developer.mozilla.org/en-US/docs/Web/CSS/Reference).

There's also a guide (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference) on what values you can use in each of properties. Just make sure you put the values in quotes when you use them in your JS!

5.4.3. Problem: Hidden Cookies

Cookie Monster was so happy with the easter egg (https://en.wikipedia.org/wiki/Easter_egg_(media)) you added to his blog that he wants more!

You've already added an easter egg that takes a bite out of the cookie in Cookie Monster's Blog's heading. But it's time to make this easter egg really exciting!

The cookie in the heading should do the following `onClick`:

- Change to the image with a bite taken out (you've already got the code for this)
- Change the `article`'s `backgroundColor` to Cookie Monster's favourite colour, `dodgerblue`
- Change the `article`'s `backgroundImage` to `'url(cookie-full.png)'`
- Change the `color` of the text in the `article` to `snow`

You'll need

index.html

```
<h1>
  <img src="cookie-full.png" onClick="bite()">
  Cookie Monster's Blog
</h1>
<h2>Welcome to me blog!</h2>
<article>
  <h3>Cookie Bunny is Coming to Town</h3>
  <p>Me not Cookie Monster. Me guest blog writer, Cookie Bunny.</p>
  <p>Me hide cookies for all the girls and boys to find!</p>
</article>

<script src="easter-egg.js"></script>
```

easter-egg.js

```
let cookie = document.querySelector('img');

function bite() {
  cookie.src = 'cookie-bite.png';
}
```

Testing

- [ ] Checking that the style of the blog doesn't change before the cookie is clicked.
- [ ] Checking that you changed the cookie image on click.
- [ ] Checking that you changed the background colour to `dodgerblue`.
- [ ] Checking that you changed the background image to cookies.
- [ ] Checking that you changed the text colour to `snow`.

5.5. Cookie animation

5.5.1. Transformation styles

One of the more interesting style properties is called `transform`. Using `transform` you can `scale()`, `translate()` or `rotate()` DOM elements.

Here's an example of enlarging, moving down and to the left, and slightly turning an image:

```
<h1>Click the hand!</h1>
<div>
  <img src="hand.png" onClick="dunk()">
</div>
<div>
  <img src="milk.png">
</div>
<script src="main.js"></script>
```

main.js

```
let hand = document.querySelector('img');

function dunk() {
  hand.style.transform = 'scale(1.2) translate(-20px, 120px) rotate(20deg)';
}
```

5.5.2. CSS Transitions

Now that we know how to change the style of the page when we interact with it we can do some animation! By using *CSS transitions* we can animate a DOM element any time it is changed with JavaScript.

Here's an example where we animate a misleading heading fading and unfading when we mouse over it:

```
<link rel="stylesheet" type="text/css" href="main.css">
<h1 onmouseenter="fade()" onmouseleave="unfade()">
  Hover over me to get a cookie!
</h1>
<script src="main.js"></script>
```

main.css

```
h1 {
  transition: opacity 1s;
}
```

main.js

```
function fade() {
  let heading = document.querySelector('h1');
  heading.style.opacity = 0;
}

function unfade() {
  let heading = document.querySelector('h1');
  heading.style.opacity = 1;
}
```

As you can see above, to add a CSS transition we just use the `transition` property.

The value of the `transition` property is the CSS property you would like to animate. In the example we use `opacity` as the animated property.

Then, after the animated property, we put the amount of time you would like the animation to take. In the example it is 1 second which is represented as `1s` in CSS.

5.5.3. Transform transition

Transitions and transformations are cool. But what's even cooler? Both at once!

```
<link rel="stylesheet" type="text/css" href="main.css">
<h1>Click and hold the hand!</h1>
<div>
  <img src="hand.png" onMouseDown="dunk()" onMouseUp="hover()">
</div>
<div>
  <img src="milk.png">
</div>
<script src="main.js"></script>
```

main.css

```css
img {
  transition: transform 1s;
}
```

main.js

```js
let hand = document.querySelector('img');

function dunk() {
  hand.style.transform = 'scale(1.2) translate(-20px, 120px) rotate(20deg)';
}

function hover() {
  hand.style.transform = '';
}
```

Click and hold the hand!

5.5.4. Problem: Catch the cookie

There's been a recent uptick in cookies rolling off the benches and around the kitchen at the Australian Cookie Academy! The ACA have decided to create a simulator for catching run away cookies.

Catching the cookie has already been implemented.

You need to transform the cookie in the `startRolling` function. To do this you will need to `rotate()` the cookie `720deg` and `translate()` the cookie `1200px` to the right.

The CSS transition between the initial cookie position and the `transform`ed position should take 3 seconds.

> **Your order matters to us**
>
> You'll notice that the order in which the `translate()` and `rotate()` are used matters!

You'll need

main.css

```
img {

}
```

index.html

```
<link rel="stylesheet" type="text/css" href="main.css">

<img src="choc-digestive.png" onMouseEnter="startRolling()" onClick="caught()">

<script src="main.js"></script>
```

main.js

```
let cookie = document.querySelector('img');

function startRolling() {

}

function caught() {
  alert('Nice one, mate!');
  cookie.style.transform = '';
}
```

Testing

- [ ] Checking that the cookie starts off in the starting position.
- [ ] Checking that the cookie moves when the mouse enters.
- [ ] Checking that the cookie rolls (rotates) when the mouse enters.
- [ ] Checking that the cookie has rolled exactly 1200px in the 3 seconds after the mouse enters the cookie.
- [ ] Checking that the cookie rolled half way in half the time.
- [ ] Checking that the cookie returns back to the start when it's caught.

5.6. Cookie upgrades

5.6.1. If statements

In our cookie clicker we could click forever but except for a growing number each click is really the same as the last. To make the game more interesting let's check how many clicks have been done and then change the type of cookie being clicked.

To do this we'll need to learn about *if statements*. If statements allow us to do something, but only if a condition we choose is true about our data.

In JavaScript this looks like:

```
<script src="main.js"></script>
```

main.js

```
let cookies = parseInt(prompt('How many cookies do you have?'));
if (cookies > 1) {
  alert('Please share a cookie with me!');
}
```

Try this code in your browser to see the results

Inside of the parentheses (`()`) is the *condition*. If the condition is true then code inside of the curly braces (`{}`) will be run. In this case the computer asks for our cookies!

5.6.2. More if statements

How can we check for more conditions? Just add more if statements!

Checking for equal values

In JavaScript we use `=` for *assignment*. So what symbol can we use to check values are equal?

`==` (two equals signs) is what we use in JavaScript to check if values are equal.

In JavaScript this looks like:

```
<script src="main.js"></script>
```

main.js

```
let cookies = parseInt(prompt('How many cookies do you have?'));
if (cookies > 1) {
  alert('Please share a cookie with me!');
}
if (cookies == 0) {
  alert('Would you like me to bake a cookie?');
}
```

Try this code in your browser to see the results

5.6.3. If images

We can do more than just `alert()` inside of an `if` statement. We can change the text content or even change images!

```
<img src="plate.png">
<script src="main.js"></script>
```

main.js

```
let cookies = prompt('How many cookies are left?');
let image = document.querySelector('img');
if (cookies == 0) {
  image.src = 'crumbs.jpg';
}
if (cookies == 1) {
  image.src = 'cookie.jpg';
}
if (cookies > 1) {
  image.src = 'cookies.jpg';
}
```

Try this code in your browser to see the results

5.6.4. Problem: Custom cookies

Mega Cookie Corp are starting a new bespoke cookie bakery. This means they take their robot-hand made cookies and charge you a custom price!

MCC needs you to create a website to go on the tablets used by customers to order a cookie.

You need to create a **`prompt`** that asks `'How many dollars would you like to spend on your hand crafted cookie?'`.

Once you know how much money the customer would like to spend then if the amount of money is:

- $0 change the paragraph (**p** element) to `Sorry, we don't have any free cookies. Have this promotional plate.`
- Greater than $0 change the paragraph to `Have a selection of our recycled micro-cookies.` and change the picture to **`crumbs.png`**
- Greater than $1 change the paragraph to `Two fingers of Scottish Highland shortbread.` and the picture to **`scotch-finger.png`**
- Greater than $10 change the paragraph to `A singular roasted Californian Almond has been lovingly nestled into your cookie.` and the picture to **`chinese-almond.png`**
- Greater than $20 change the paragraph to `Freshly grated Fijian Coconut lightly sprinkled over a confit of English raspberries on an Italian biscotto.` and the picture to **`raspberry-ice.png`**
- Greater than $50 change the paragraph to `Oh! You must be a real connoisseur! Have our finest and most difficult to bake cookie.` and the picture to **`choc-chip.png`**

> **The most expensive order**
>
> The order you write the **`if`** statements is important. Customers should always end up with the most expensive cookie!

You'll need

index.html

```
<h1>Mega Cookie Corp Cookiery</h1>
<p>Please order your personal, bespoke cookie</p>
<img src="plate.png">

<script src="main.js"></script>
```

crumbs.png

Testing

- [ ] Checking that the web page initially shows an empty plate and the correct default text.
- [ ] Checking that you prompted the customer for how much money they would like to spend.
- [ ] Checking that you show the plate and correct message for $0.
- [ ] Checking that you show correct cookie and message for $2.
- [ ] Checking that you show correct cookie and message for $8 - the same as for $2.
- [ ] Checking that you show correct cookie and message for $1.
- [ ] Checking that you show correct cookie and message for $15.

- [ ] Checking that you show correct cookie and message for $27.
- [ ] Checking that you show correct cookie and message for $100.

5.7. Cookie party

5.7.1. Colour

We've already seen colours represented as *strings* like in `black`, `green` and `aliceblue`. But what if we want to represent a colour that hasn't been named yet?

One way to do this is CSS and JS is using three values called *hue*, *saturation* and *lightness*.

Lightness is the easiest to understand. It's just how bright the colour is!

Saturation is how colourful the colour is. The lower the saturation the greyer the colour will be. The higher the saturation the more vibrant the colour is!

Hue is the trickiest one. Hue is the colour of the colour. To put it another way, hue is the part of the rainbow this colour falls within. The rainbow can be imagined like a colour wheel. Starting with red at 0 going through orange, yellow, green, blue and purple and all the way back to red at 360.

5.7.2. Changing Colours

We can change the colour of an image using the CSS filter called `hue-rotate`. It spins the colours in an image around the colour wheel.

Try changing the number in this example:

```
<img src="macaron.png">
<script src="main.js"></script>
```

main.js

```
let image = document.querySelector('img');
image.style.filter = 'hue-rotate(180deg)';
```

5.7.3. Random Numbers

Changing an image to just one colour is boring. How about changing colours randomly?

We can use `Math.random()` to create a random number from 0 to 1.

But for `hue-rotate()` we need from 0 to 360. So if we multiply the random number by 360 we can get a random number from 0 to 360! And multiply we use the `*` operator.

So we end up with `Math.random() * 360` to create a random hue.

To put this number into our `hue-rotate()` function we can just use *string templates*!

In this example we use random numbers to change the image when it is clicked!

```
<img src="macaron.png" onClick="party()">
<script src="main.js"></script>
```

main.js

```
function party() {
  let hue = Math.random() * 360;
  let image = document.querySelector('img');
  image.style.filter = `hue-rotate(${hue}deg)`;
}
```

Don't forget!

String templates use backticks (``) not quotes (''). So make sure you use backticks for your string templates!

5.7.4. Problem: Mystery Macaron Hue

The Fancy Cookie Club is creating an event for macaron tasting. Each person attending gets to choose and try one of the fancy flavours from around the world.

The Fancy Cookie Club already has a website which allows you to pick a flavour. But it doesn't show you the colour of the macaron you chose.

Use `hue-rotate()` to set the colour of the macaron when you click each button.

Here is the hue of each flavour of macaron:

- Durian - `80`
- Hanza - `120`
- Maqui - `230`
- Lingonberry - `30`
- Kakadu Plum - `150`
- Taro - `330`

You'll need

index.html

```
<h1>Fancy Cookie Club</h1>
<h2>Pick a Flavour</h2>

<div>
  <button onClick="durian()">Durian</button>
  <button onClick="hanza()">Hanza</button>
  <button onClick="maqui()">Maqui</button>
  <button onClick="lingonberry()">Lingonberry</button>
  <button onClick="kakaduPlum()">Kakadu Plum</button>
  <button onClick="taro()">Taro</button>
</div>

<h3>Undecided</h3>

<img src="macaron.png">


<script src="main.js"></script>
```

main.js

```
let heading = document.querySelector('h3');

function durian() {
  heading.textContent = 'Durian';
}

function hanza() {
  heading.textContent = 'Hanza';
}

function maqui() {
  heading.textContent = 'Maqui';
}

function lingonberry() {
  heading.textContent = 'Lingonberry';
}

function kakaduPlum() {
  heading.textContent = 'Kakadu Plum';
}

function taro() {
  heading.textContent = 'Taro';
}
```

Testing

- [ ] Checking the web page initially shows the undecided flavour of macaron.
- [ ] Checking that the heading was updated when the Durian button was clicked.
- [ ] Checking that the colour of the macaron was change when the Durian button was clicked.
- [ ] Checking that the heading was set and colour was changed when the Hanza button was clicked.
- [ ] Checking that the heading was set and colour was changed when the Maqui button was clicked.
- [ ] Checking that the heading was set and colour was changed when the Lingonberry button was clicked.
- [ ] Checking that the heading was set and colour was changed when the Kakadu Plum button was clicked.
- [ ] Checking that the heading was set and colour was changed when the Taro button was clicked.