ISCG6420 Semester 1 2024

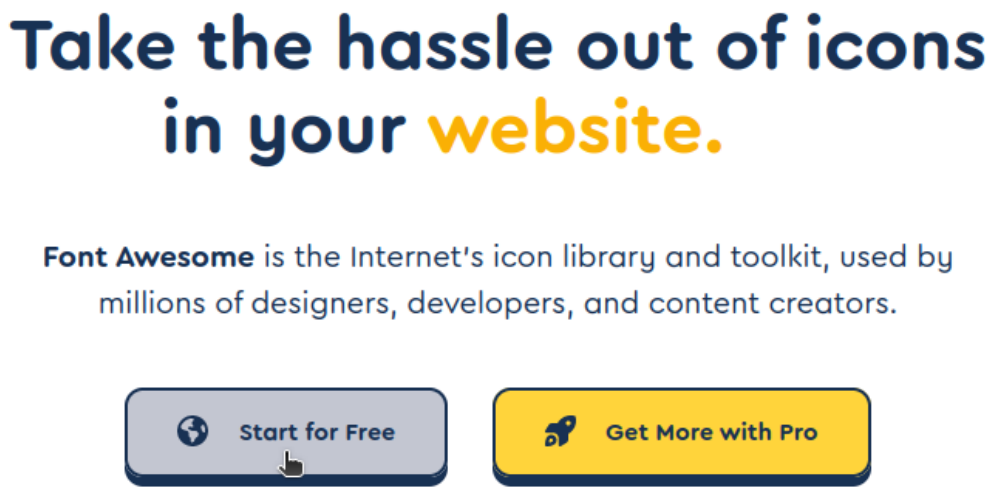
Exercise: Manipulating Animations with JS

This exercise will take you through adding replay functionality to your CSS advertisement animation. It will also utilise font-based icon provider FontAwesome to deliver website icons from a Content Delivery Network (CDN).

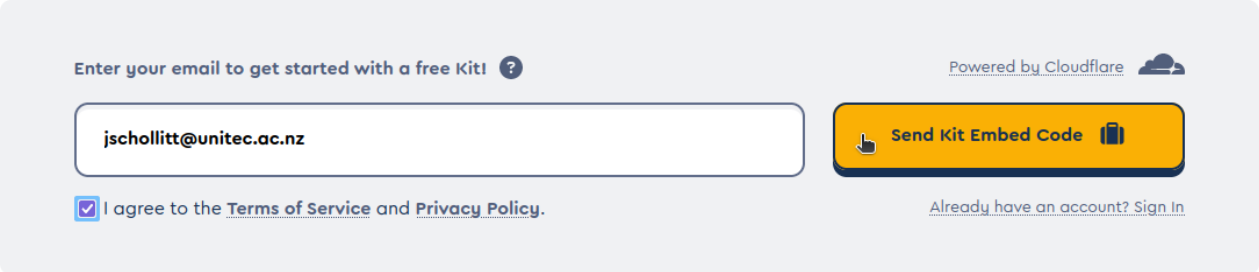
Get FontAwesome access

Font Awesome (FA) is a highly utilised icon delivery system for websites and applications. It uses customised fonts to provide lightweight, yet scalable icons to user interfaces. Using FA in a website is easy to do – simply add a script with a link to a FA CDN and add classes to our HTML elements.

Open a web browser and go to : https://fontawesome.com/. Click the "Start for Free" button to begin the account creation process.



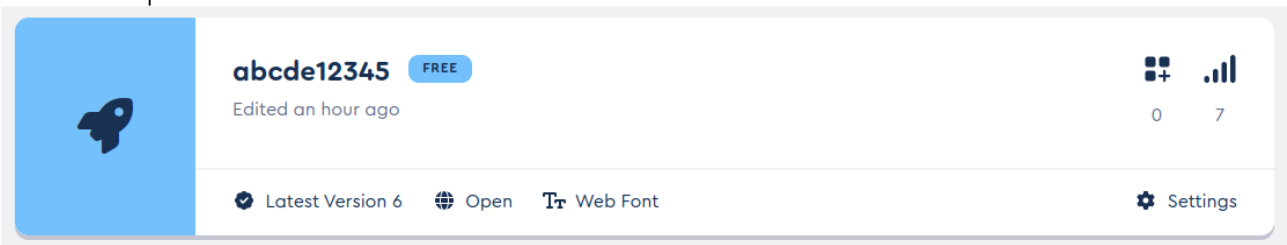
Enter an email address you wish to use for the signup process, tick the terms of service agreement (after reading the policies), and click the "Send Kit Embed Code" button



You will be sent an email with a link to confirm your address. Follow the link from the email to return to FontAwesome with a created account. If prompted for personal details, simply skip this step by using the link below the form to continue.

Add FontAwesome to our webpage

Navigate to the “Your Kits” page from the top nav bar. Half way down the page will be a kit with a unique name. Click on the kit:



In this kit will be a code box with an HTML <script> element. Copy this element and add it to the head of your website HTML.

Update Project code

Using your preffered code editor, open your project for Week 3 exercise: Create CSS animated advertisement. In your advertisement webpage, add the copied <script> element to the end of the <head> section:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Week 3</title>
    <link …
    />
    <link rel="stylesheet" href="./week3.css" />

    <script src="https://kit.fontawesome.com/abcde12345.js" crossorigin="anonymous"></script>

  </head>
```

We can now add icons to our webpage. To see the icons available, see the Icons page on the FontAwesome website: https://fontawesome.com/search

Note: we only have access to non-pro icons with a free account.

Create a Replay button

To add a FA icon to our webpage web can copy code snippets from the FA website. For this exercise we want an icon that will represent a replay action. Clicking on any icon from the FA search page will reveal the details of the icon, including design variations and code snippets for different technologies. We will be using the “circle-play solid” icon: https://fontawesome.com/icons/circle-play?f=classic&s=solid

Copy the HTML code snippet.

In your advertisement webpage, find the div with ID: “adWrapper”. Inside this div is a div for scene 1 of the advertisement. Create a second scene div below it and create an element inside with ID: “adReplay”.

```
<div id="adWrapper">
    <div id="adScene1">…
    </div>
    <div id="adScene2">
        <div id="adReplay">

        </div>
    </div>
</div>
```

On the blank line inside “adReplay” paste the code snippet for the FA circle-play icon.

```
<div id="adScene2">
  <div id="adReplay">
    <i class="fa-solid fa-circle-play"></i>
  </div>
</div>
```

Style the Icon

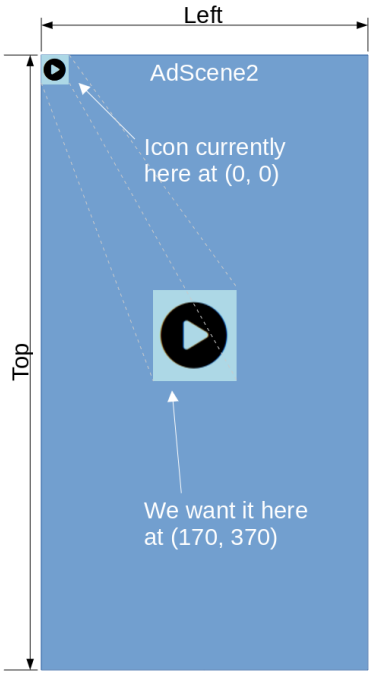
If we view our website in a web browser we can see our new icon. After the ad animation the icon appears but is small and in the top-left corner of our advertisement. To make it appear correctly we need to add some CSS.

Create a style tag for the ID “adReplay” and add the following style attributes:

```
#adReplay {
    position: absolute;
    left: 170px;
    top: 370px;
    font-size: 60px;
}
```

The position: absolute will allow us to set pixel positioning relative to the parent element. This means the top and left attributes will be relative to the “adScene2” position. We also want to increase the size of the icon. Since it is a font we can do this with font-size.

These styles will apply changes like this:



Integrating with the animation sequence

We want the replay icon to appear at the end of the scene 1 animation, and to stay on screen once all animations have completed. To do this we can make an animation for the replay scene. This scene will stay invisible for the duration of scene 1, then become visible.

Note: If we want an element style to stay after an animation, we need to set that style as the default for the element, and use the animation sequence to change it for the duration of the animation. Once the animation completes the element styles will return to default.

Create a style for ID “adScene2”.

```
#adScene2 {
    visibility: visible;
    animation-name: scene2;
    animation-duration: 5s;
}
```

This sets the scene default to visible, and applies an animation that runs for 5 seconds. We can make the scene invisible for the duration of the animation (while scene 1 is animating), and then appear afterwards.

Create keyframes for “scene2”.

```
@keyframes scene2 {
    0% {
        visibility: hidden;
    }

    99% {
        visibility: hidden;
    }

    100% {
        visibility: visible;
    }
}
```

JavaScript

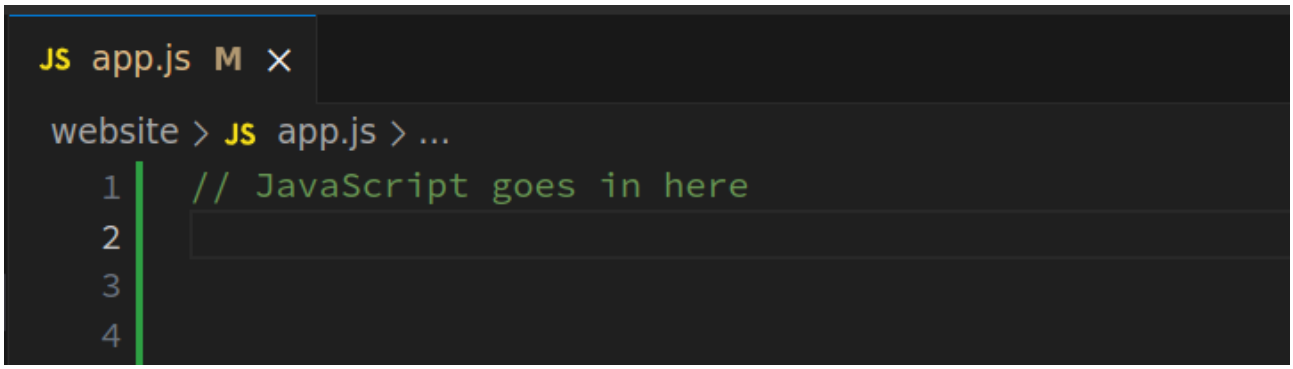
Refreshing the website shows the advertisement running as before, but now our replay icon appears after the animation, in the centre. The final step is to make the icon do something when clicked.

In the HTML, add an onclick event attribute to the <i> element:

```
<div id="adScene2">
  <div id="adReplay">
    <i class="fa-solid fa-circle-play" onclick="adReplay()"></i>
  </div>
</div>
```

Here we have supplied the onclick attribute with the name of a JavaScript function. This will then call that function when the element is clicked on the webpage.

Next, create a new file in the same directory as the HTML file. Call it “app.js” and open it for editing:



Create a new function with the name “adReplay”:

```
function adReplay() {

}
```

Create a block-level variable (let or const) and set its value to the element with ID: “adWrapper”:

```
function adReplay() {
    let ad = document.getElementById("adWrapper");
}
```

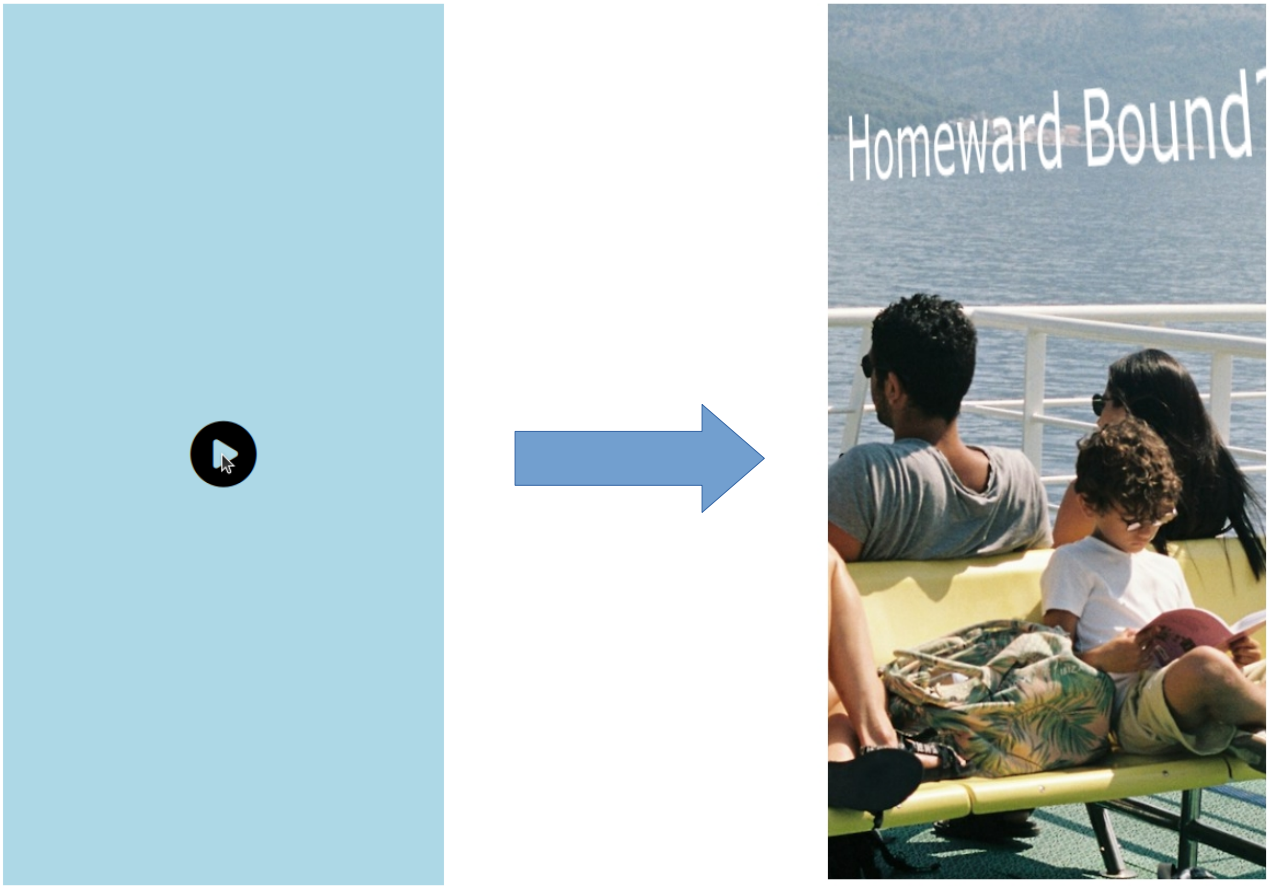
Create another variable and set its value to be a copy of the element created above:

```
function adReplay() {
    let ad = document.getElementById("adWrapper");
    let copy = ad.cloneNode(true);
}
```

Finally, let's replace the first element with the copy:

```
function adReplay() {
    let ad = document.getElementById("adWrapper");
    let copy = ad.cloneNode(true);
    ad.replaceWith(copy);
}
```

This function makes use of the interaction between element changes in the DOM and CSS animation timers. By replacing the adWrapper element with a clone, we are effectively resetting the timers for the animations and allowing the user to see a rerun of the advertisement.



Using this technique there are many ways we can use DOM manipulation to enhance our animations.
For more complex animations, JavaScript can be used for spawning and deleting elements instead of applying timers to every element and trying to orchestrate elements with delays and keyframe sequencing.

Further exercises

Try using the learning outcomes from this exercise in other parts of your websites.

- Add Font Awesome icons to elements in your navigation
- Replace button text with icons where the action can be described visually (Save, Print, Share, Social Media, etc)
- Add content and libraries from Content Delivery Networks to add functionality to your website (jQuery, AJAX, image hosting, etc)

Wrapping up

Save your work, commit and push your changes to your git repository.

Exercise complete.