Tutorial 4

Editor – Brackets

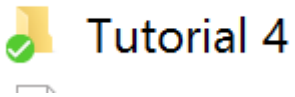
Goals

Create a website showcasing the following techniques

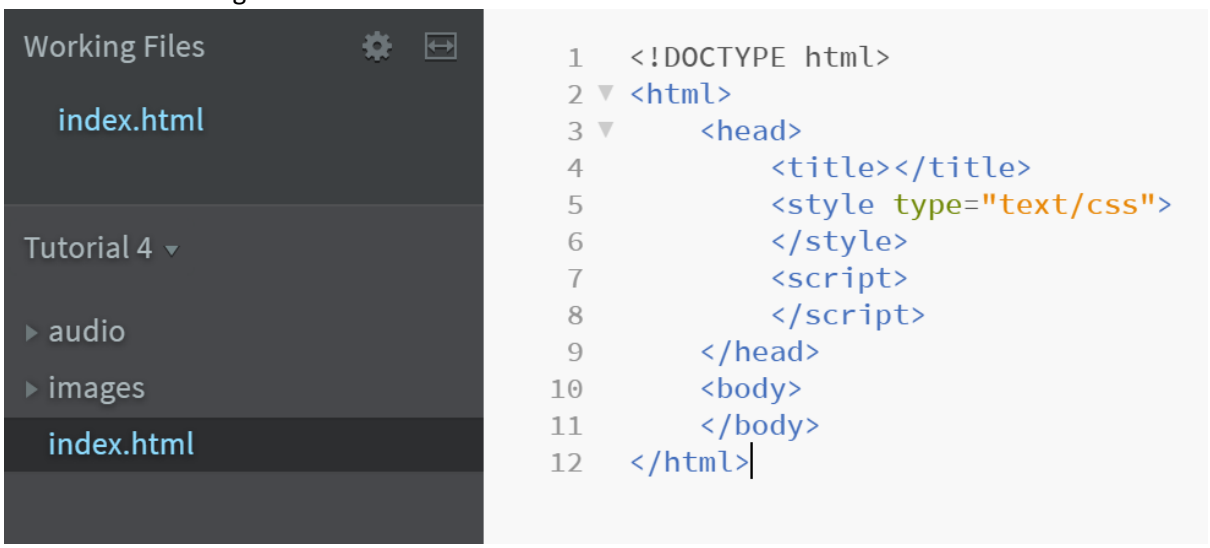
- Dealing with sound, making areas on an image produce sound
- Dealing with external helper files to get sound working within a canvas.

Website

- Create a folder on the desktop called tutorial 4



- Open Brackets
- Create the following structure and index.html file

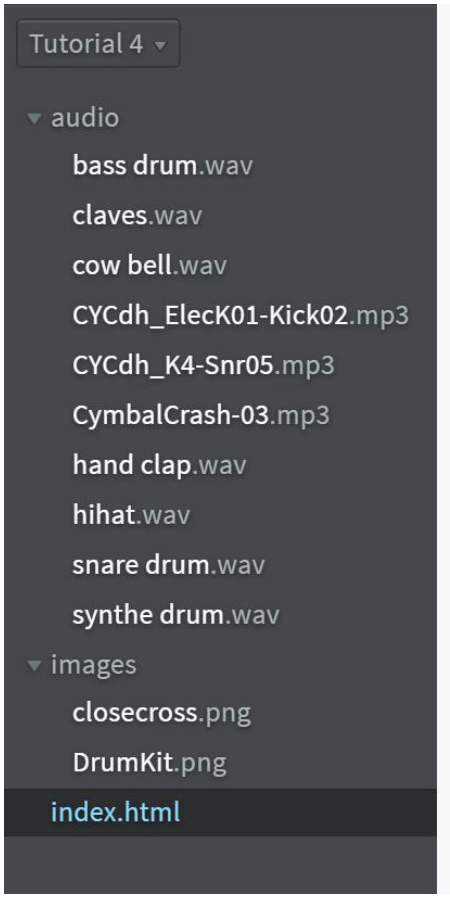


-

Index Page

Open up index.html, we will be working on this to create the page.

Download the resource files from L@G. Put them in the appropriate folders and remove the extraneous files. You should have the following setup.



Next, we load up the resources and load the image file of the drum kit, at the moment, the audio will play upon the load of the page, but when we click on the drum image there is nothing there to activate other sounds.

Use the following code

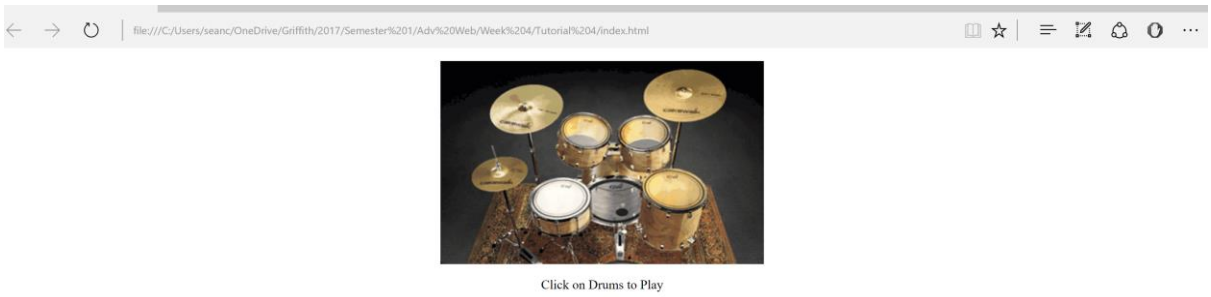
Div

```
<body>
    <audio id="Bassdrum" autoplay="false" defaultMuted="true" src="audio/bass%20drum.wav" ></audio>
    <audio id="Snare" autoplay="false" defaultMuted="true" src="audio/snare%20drum.wav" ></audio>
    <p><img id="Drums" src="images/DrumKit.png" class="drumSize" alt="dk" onclick="handleClick(event)"></p>
    <p>Click on Drums to Play </p>
</body>
```

Add the following css

```
<style type="text/css">
    .drumSize{width: 400px; height: 250px; display: block; margin: 0 auto;}
    p{text-align: center;}
</style>
```

Save and test the page, you should see the following

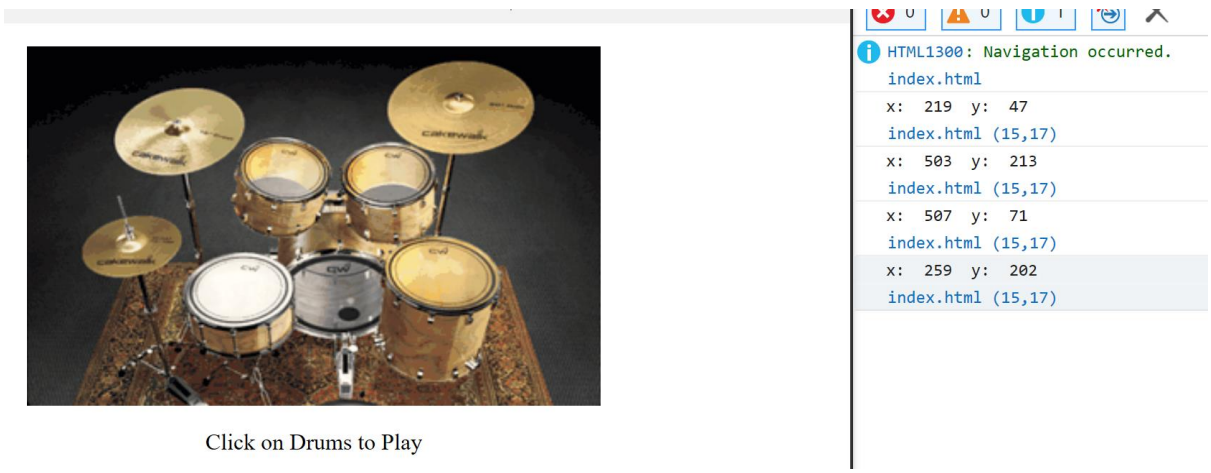


Now that we have the drum kit loaded, we will write some code to demonstrate onclick event.

Javascript

```
<script>
    function handleClick(event)
    {
        // find out what was clicked
        var windowClickX = event.clientX;
        var windowClickY = event.clientY;
        console.log('x: ',windowClickX,' y: ',windowClickY);
    }
</script>
```

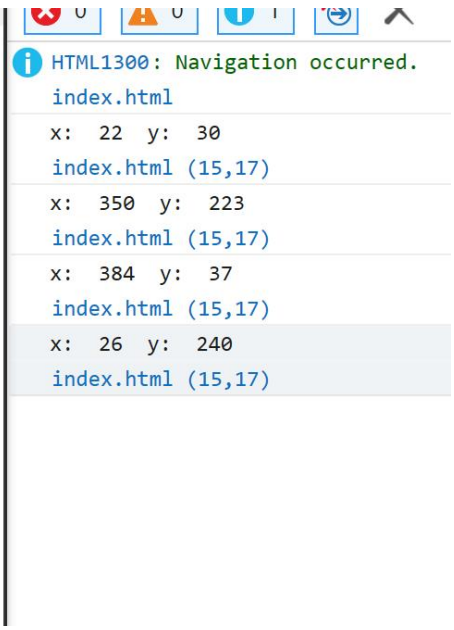
So, with the console open, test the page, you should be able to see the following



This is good in that we can now view what is occurring when we click, but what happens when we shrink the window and test it, click in the corners of the image after you have refreshed the page and shrunk the window.

You will see the following

Click on Drums to Play



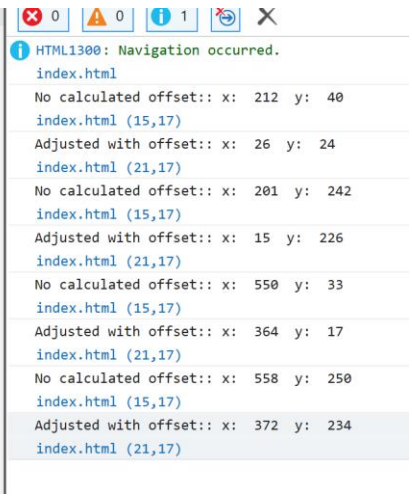
In this case, the window was shrunk to the point that there is basically no white space around the image at all. Notice how different the console has recorded the X value. This is because there is an element to the page that is called offset. The offset is a variable that can contain differing values based upon the difference between the object selected and the left edge of the screen. There is also an offset from the top of the page to the object as well. Now, we need to add code to account for this varying distance.

Javascript

```
<script>
    function handleClick(event)
    {
        // find out what was clicked
        var windowClickX = event.clientX;
        var windowClickY = event.clientY;
        console.log('No calculated offset:: x: ',windowClickX,' y: ',windowClickY);
        var drum_im = document.getElementById("Drums");
        var winOffsetX = drum_im.offsetLeft - drum_im.scrollLeft;
        var winOffsetY = drum_im.offsetTop - drum_im.scrollTop;
        var clickX = windowClickX-winOffsetX;
        var clickY = windowClickY-winOffsetY;
        console.log('Adjusted with offset:: x: ',clickX,' y: ',clickY);
    }
</script>
```

To test this new code, re-center the page, and reload, then click in all 4 corners of the image. You should see something similar to the following. Remember that the numbers will be different based upon the screen resolution of the device this is being tested on.

Click on Drums to Play



Now that we have the ability to calculate the offset, we will split the image in two, this will then allow us to click on either the left hand side or right hand side of the image to be able to load and play the audio source.

Use the following code:

```
<script>
   function handleClick(event)
   {
       // find out what was clicked
       var windowClickX = event.clientX;
       var windowClickY = event.clientY;
       console.log('No calculated offset:: x: ',windowClickX,' y: ',windowClickY);
       var drum_im = document.getElementById("Drums");
       var winOffsetX = drum_im.offsetLeft - drum_im.scrollLeft;
       var winOffsetY = drum_im.offsetTop - drum_im.scrollTop;
       var clickX = windowClickX-winOffsetX;
       var clickY = windowClickY-winOffsetY;
       console.log('Adjusted with offset:: x: ',clickX,' y: ',clickY);
       // match that to a drum and call function to play its noise
       if(clickX > 200)
       {
           document.getElementById("Snare").play();
       }
       else
       {
           document.getElementById("Bassdrum").play();
       }
   }
</script>
```

Save and test, you should be hearing differing sounds from each side of the image.

Next we need to split up the image from left and right sides, we are going to be changing the code to ensure that there is precise areas on the image in which if you click, a differing sound is played due.

To start with, we need to determine the areas that need to be modified. So, let's examine the image we are working with.

Each section of the drumkit can be split up into boxes of which can then have the click area tracked.

This is a prime example of why the use of console.log and handle click is so important and useful for implementing precision in programming.

First we determine the areas:

Each of the red boxes are going to be needed as a clickable area.

To start with, we need to add all the audio files as before, instead of loading 2 audio files we will put in the code to load 8 audio files.

Div code

```
<body>
    <audio id="Bassdrum" autoplay="false" defaultMuted="true" src="audio/bass%20drum.wav" ></audio>
    <audio id="Snare" autoplay="false" defaultMuted="true" src="audio/snare%20drum.wav" ></audio>
    <audio id="Claves" autoplay="false" defaultMuted="true" src="audio/claves.wav" ></audio>
    <audio id="Cow" autoplay="false" defaultMuted="true" src="audio/cow%20bell.wav" ></audio>
    <audio id="Kick" autoplay="false" defaultMuted="true" src="audio/CYCdh_ElecK01-Kick02.mp3" ></audio>
    <audio id="Snr" autoplay="false" defaultMuted="true" src="audio/CYCdh_K4-Snr05.mp3" ></audio>
    <audio id="Crash" autoplay="false" defaultMuted="true" src="audio/CymbalCrash-03.mp3" ></audio>
    <audio id="Hihat" autoplay="false" defaultMuted="true" src="audio/hihat.wav" ></audio>
    <audio id="Synth" autoplay="false" defaultMuted="true" src="audio/synthe%20drum.wav" ></audio>
    <p><img id="Drums" src="images/DrumKit.png" class="drumSize" alt="dk" onclick="handleClick(event)"></p>
    <p>Click on Drums to Play </p>
</body>
```

If you save and test, there is no change to the current page except that in the background, the system knows of the new audio files. Next, we modify the clean up the handleclick function to use as the tool to determine the size of each box on the image.

JavaScript

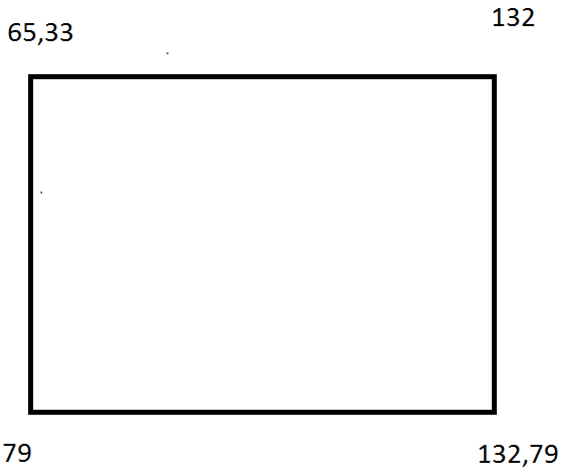
```
function handleClick(event)
{
    var windowClickX = event.clientX;
    var windowClickY = event.clientY;
    var drum_im = document.getElementById("Drums");
    var winOffsetX = drum_im.offsetLeft - drum_im.scrollLeft;
    var winOffsetY = drum_im.offsetTop - drum_im.scrollTop;
    var clickX = windowClickX-winOffsetX;
    var clickY = windowClickY-winOffsetY;
    console.log('Adjusted with offset:: x: ',clickX,' y: ',clickY);
}
```

Once this is done, we save and load up the page. With the console open, we click the top left corner and bottom right corner of each section on the drum, to give us the dimensions needed for that specific area.

Console

```
index.html
Adjusted with offset:: x:  65  y:  33
index.html (19,17)
Adjusted with offset:: x:  132  y:  79
index.html (19,17)
Adjusted with offset:: x:  42  y:  122
index.html (19,17)
Adjusted with offset:: x:  88  y:  155
index.html (19,17)
Adjusted with offset:: x:  154  y:  66
index.html (19,17)
Adjusted with offset:: x:  199  y:  102
index.html (19,17)
Adjusted with offset:: x:  119  y:  148
index.html (19,17)
Adjusted with offset:: x:  165  y:  190
```

Once we have this information, we can modify the Javascript to play the audio. This is done by using a separate function and implementing some IF statements. So, the Cymbol on the left it's top left position is 65,33 and bottom right is 132,79. Using these numbers we can then calculate the box like this:



From here we type up the following JavaScript

```
function handleClick(event)
{
    var windowClickX = event.clientX;
    var windowClickY = event.clientY;
    var drum_im = document.getElementById("Drums");
    var winOffsetX = drum_im.offsetLeft - drum_im.scrollLeft;
    var winOffsetY = drum_im.offsetTop - drum_im.scrollTop;
    var clickX = windowClickX-winOffsetX;
    var clickY = windowClickY-winOffsetY;
    console.log('Adjusted with offset:: x: ',clickX,' y: ',clickY);
    playAudio(clickX,clickY);
}
function playAudio(x,y)
{
    /*Crash*/
    if ((x>= 65) && (x<=132))
        {
            if ((y>=33) && (y<=79))
                {
                    document.getElementById("Crash").play();
                }
        }
    /*Bass Drum*/
    /* Snare */
    /*Claves*/
    /*Cow*/
    /*Kick*/
    /*Snr*/
    /*HiHat*/
    /*Synth*/
}
```

The /**/ are comments and in the playAudio function are bookmarks for where the code will go for each element of the drum.

Notice the modification to the handleClick function. We are feeding the x and y co-ordinates into the playAudio function. Once we have them there, we check to see if the clicks occur in the correct logical box.

Now, we add the elements one at a time, with testing. This way, if there are any mistakes, we will only have one area to look at.

Either fill in the X and Y co-ordinates yourself, or use the following

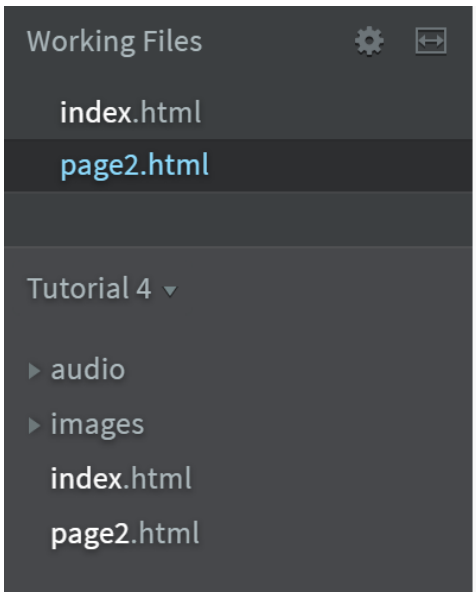
```
function playAudio(x,y)
{
    /*Crash*/
    if ((x>= 65) && (x<=132))
        {
            if ((y>=33) && (y<=79))
                {
                    document.getElementById("Crash").play();
                }
        }
    /*Bass Drum*/
   if ((x>= 187) && (x<=237))
        {
            if ((y>=146) && (y<=199))
                {
                    document.getElementById("Bassdrum").play();
                }
        }
    /* Snare */
     if ((x>= 187) && (x<=237))
        {
            if ((y>=146) && (y<=199))
                {
                    document.getElementById("Bassdrum").play();
                }
        }
    /*Claves*/
    if ((x>= 51) && (x<= 85))
        {
            if ((y>= 120) && (y<= 155))
                {
                    document.getElementById("Claves").play();
                }
        }
    /*Cow*/
    if ((x>= 155) && (x<= 197))
        {
            if ((y>= 67) && (y<= 104))
                {
                    document.getElementById("Cow").play();
                }
        }
```

```
        }
    /*Kick*/
     if ((x>=121 ) && (x<=171))
        {
            if ((y>=151) && (y<=193))
                {
                    document.getElementById("Kick").play();
                }
        }
    /*Snr*/
    if ((x>= 226) && (x<=269))
        {
            if ((y>=74) && (y<=109))
                {
                    document.getElementById("Snr").play();
                }
        }
    /*HiHat*/
      if ((x>= 263) && (x<=336))
        {
            if ((y>=21) && (y<=64))
                {
                    document.getElementById("Hihat").play();
                }
        }
    /*Synth*/
     if ((x>= 258) && (x<=320))
        {
            if ((y>=137) && (y<=179))
                {
                    document.getElementById("Synth").play();
                }
        }
}
```

Test, each element on the drum kit, should make a different sound.

External Audio helper

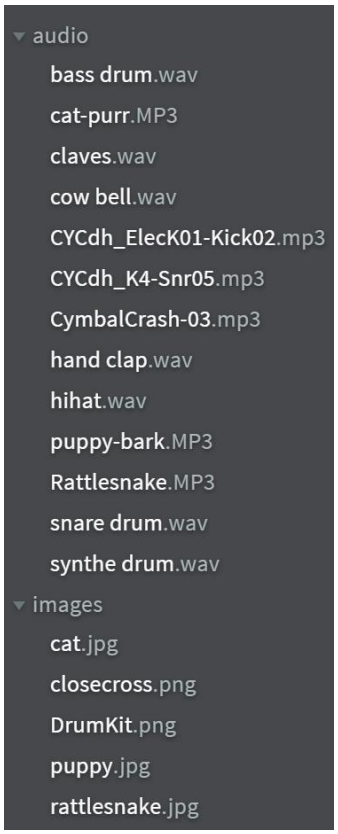
Start a new page Called page2.html




```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <script>
        </script>
    </head>
    <body>
    </body>
</html>
```

Download the resources from here: https://1drv.ms/u/s!AskBHXkgcX44pek8iXTZfy7Q3O5ORA

Extract them into the tutorial 4 folder.



Notice the new files: cat-purr.mp3, puppy-bark.mp3, Rattlesnake.mp3, cat.jpg, puppy.jpg and rattlesnake.jpg

These are the files we will be manipulating.

To start with, we will create a canvas and style it.

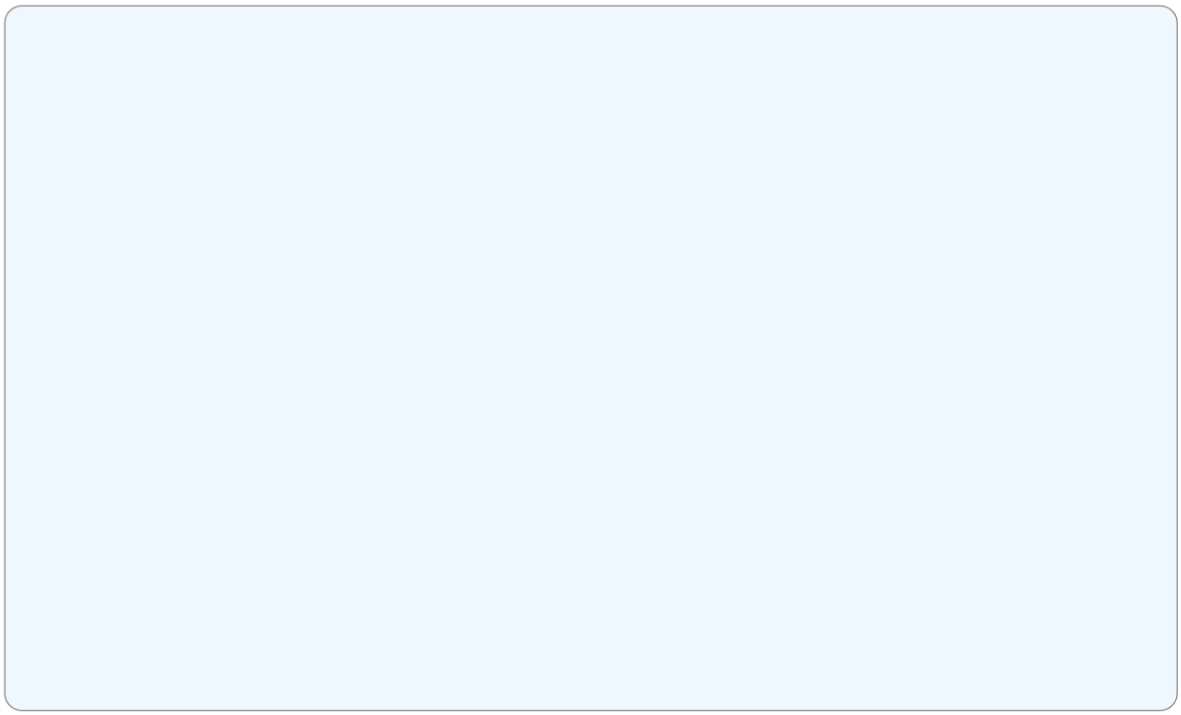
Div

```
<body>
    <canvas id="myCanvas" width="1000" height="600" onclick="handleClick(event)" class="colCanvas"></canvas>
</body>
```

Css

```
<style type="text/css">
    .colCanvas{background-color: aliceblue; border: 1px gray solid; border-radius: 15px;}
</style>
```

Result



Now, we are going to add the three images, these images will respond to the click event eventually, but for now, we'll get them to load into the page. Each will have a separate function for drawing and there will be an additional function to load each of the draw functions.

Even with the JavaScript written, there will need to be a small modification to the div code. This will be the onload function. This function will be called on the body tag, so when the page is loading up, it knows that it needs to run the draw functions as well.

Javascript

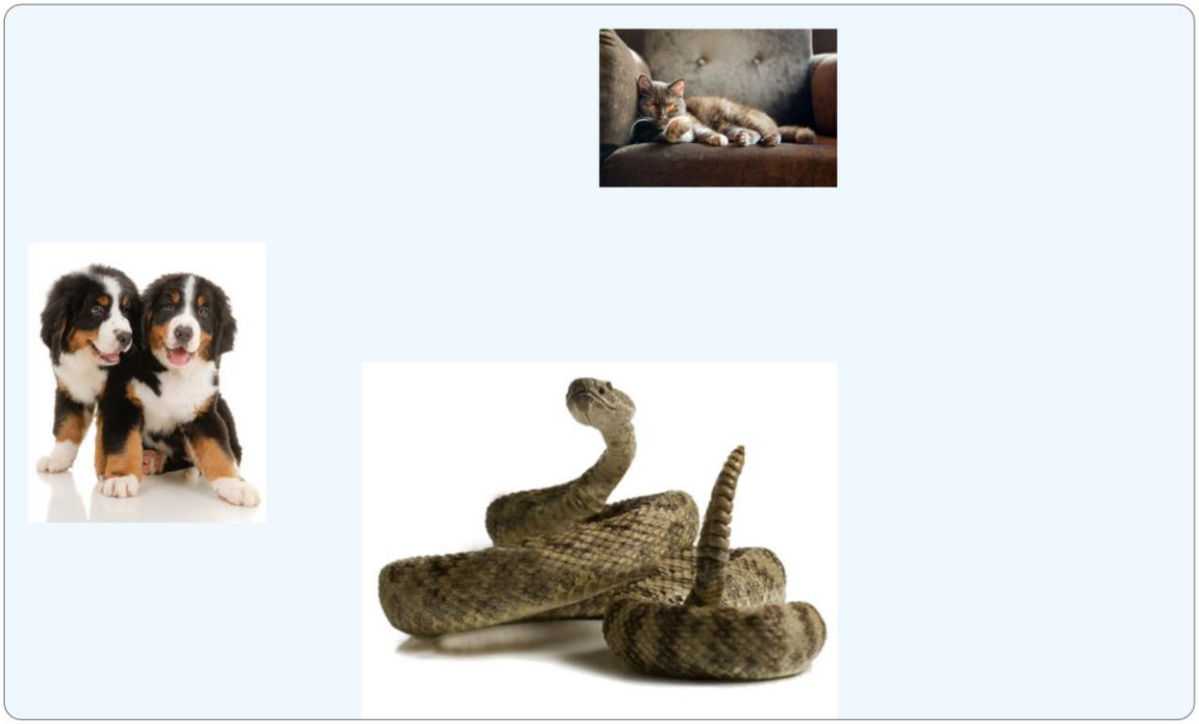
```
<script>
    function drawCat()
    {
            var ctx = document.getElementById("myCanvas").getContext('2d');
            var cat = new Image();
            cat.src = "images/cat.jpg";
            cat.onload = function()
            {
                ctx.drawImage(cat,500,20);
            }
    }
    function drawPuppy()
    {
            var ctx = document.getElementById("myCanvas").getContext('2d');
            var puppy = new Image();
            puppy.src = "images/puppy.jpg";
            puppy.onload = function()
            {
                ctx.drawImage(puppy,20,200);
            }
    }
    function drawSnake()
    {
            var ctx = document.getElementById("myCanvas").getContext('2d');
            var snake = new Image();
            snake.src = "images/rattlesnake.jpg";
            snake.onload = function()
            {
                ctx.drawImage(snake,300,300);
            }
    }

    function loadResources()
    {
        drawCat();
        drawPuppy();
        drawSnake();
    }
</script>
```

Div

```
<body onload="loadResources()">
    <canvas id="myCanvas" width="1000" height="600" onclick="handleClick(event)" class="colCanvas"></canvas>
</body>
```

Result



Now that the resources have been loaded, we need to connect to the external audio helper. We are going to be using the createJS sounds library to help this happen.

http://www.createjs.com/#!/SoundJS

This linking is done via a script call, do this above the current set of scripts

```
</style>
<script src="http://code.createjs.com/soundjs-0.6.2.min.js"></script>
<script>
    function drawCat()
```

This code access the js library stored on createjs website. A handy way to access libraries if you don't want to download them. Now, we want to load up our audio files. We need them to be ready for playing as soon as the images are loaded, so from here we'll make a few JavaScript changes and a HTML change

We also need to assign some global variables to identify the sounds.

```
<script>
    var soundIDCat = "cat";
    var soundIDPuppy = "puppy";
    var soundIDSnake = "snake";

    function drawCat()
```

These variables allow us to manipulate within any function, as they are created outside of a function they are classified as global variables.

Next, we need to create a function to load the audio files.

```
function loadSound()
{
    createjs.Sound.registerSound("audio/cat-purr.mp3", soundIDCat);
    createjs.Sound.registerSound("audio/puppy-bark.mp3", soundIDPuppy);
    createjs.Sound.registerSound("audio/Rattlesnake.mp3", soundIDSnake);
}
```

Now that we have a function that does this, we can add it to the loadResources function

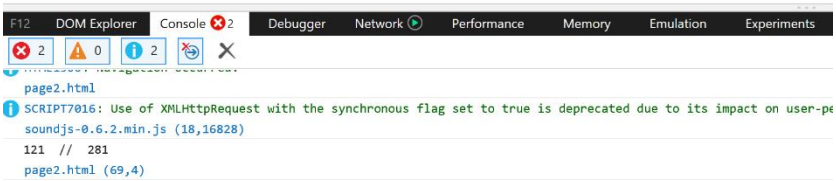
```
    function loadResources()
    {
        loadSound();
        drawCat();
        drawPuppy();
        drawSnake();
    }
</script>
```

Save all files and test the page, it should load with no errors. If there are any errors, check for typos in your code.

Everything is currently in position for loading the audio except for the actual clicking itself. Next we need to add the click ability, so we now write the handleclick function

```
function handleClick(e)
{
      posX = e.clientX;
      posY = e.clientY;
      var mcObj = document.getElementById("myCanvas");
      var winOffSetX = mcObj.offsetLeft - mcObj.scrollLeft; //calculate offset
      var winOffSetY = mcObj.offsetTop - mcObj.scrollTop; //calculate offset
      posX = posX - winOffSetX;
      posY = posY - winOffSetY;
      console.log(posX,' // ',posY);
      activateAudio();
}
```

If you save and test it, you should get an error on the activateAudio() function not existing but there will be numbers indicating x and y positioning.



Now, as we add the functions activateAudio() and playSound() these functions will allow us to make the audio play, and by being structured enough, easier to understand.

```
function activateAudio()
{
      if((posX>=25) && (posX <= 215))
      {
          if ((posY>=203) && (posY <=431))
          {
              playSound('puppy');
          }
      } // end puppy click
      if ((posX>=505) && (posX <=695))
      {
          if ((posY>=23) && (posY <= 151))
          {
              playSound("cat");
          }
      }// end of cat click
      if ((posX>=305) && (posX<=699))
      {
          if ((posY>=307) && (posY <=595))
          {
              playSound("snake");
          }
      }//end of snake click
   }

function playSound(animalName)
{
      if (animalName == 'puppy')
      {
          createjs.Sound.play(soundIDPuppy);
      }
      if (animalName == 'cat')
      {
          createjs.Sound.play(soundIDCat);
      }
      if (animalName == 'snake')
      {
          createjs.Sound.play(soundIDSnake);
      }
}
```

Once completed, save and test. Notice how we can play multiple audio streams at once.

Backup

This concludes tutorial. Save your work to USB, student drive, Onedrive, Dropbox or zip and email the folder to yourself.