learntoprogram.tv

LAB 5: Create a Mobile HTML5 site that Uses Gestures

In this lab, you will build a site that uses mobile gestures to behave more like a native application on mobile browsers. You will learn to:

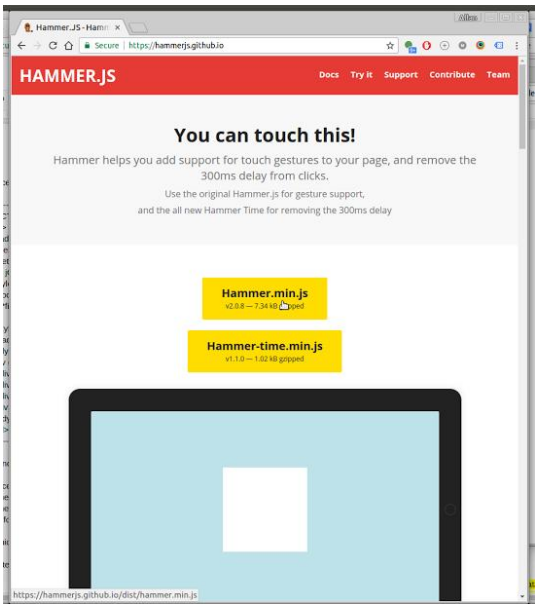
1. dynamically set the height of the content.
2. respond to tap gestures on a mobile device.
3. respond to left and right swipe gestures on a mobile device.

1. Once again, create an HTML page with the jQuery Mobile aesthetic. This time, assign an id of “main” to the main content div. You will use this id to set the height of the div.

```
<!DOCTYPE html>
<html>
  <head>
    <title>LAB 5</title>
    <meta name="viewport" …..
    <!-
- jQuery Mobile include tags here --
>
    <style>
      #footer{
        /*fix footer position rules*
/
      }
    </style>
  </head>
  <body>
    <div data-role="page">
      <div data-role="header">
        <h1>Gestures With Hammer.JS<
/h1>
      </div>
      <div data-
role="main" class="ui-content">
      </div> <!-- main -->
      <div data-
role="footer" id="footer"><h1>Lab 5<
/h1></div>
    </div>
  </body>
</html>
```

First, let’s add the Hammer.JS library to the HTML document.

2. Since there is no official CDN for Hammer.JS, go ahead and download the source for Hammer.JS. Go to the Hammer website at https://hammerjs.github.io/ and click the links to hammer.min.js and hammer-time.min.js. Copy and paste the code into new files named hammer.min.js and hammer-time.min.js in the folder with your lab5 HTML.



3. Add Javascript tags to include Hammer and Hammer-time in your HTML.

```
<script src="....
<script src="hammer.min.js"></script
>
<script src="hammer-
time.min.js"></script>
<style>
```

Before implementing gestures, you will need to explicitly set the height of the main content to the full height of the screen, allowing the Hammer library to pick up gestures across the screen.

4. Create a new script tag in the just after the Hammer include tags. Inside the new script tag, assign a function to window.onload.

```
1.    <script src="hammer-
   time.min.js"></script>
2.    <script >
3.      window.onload = function(){}
4.    </script>
5.    <style>
```

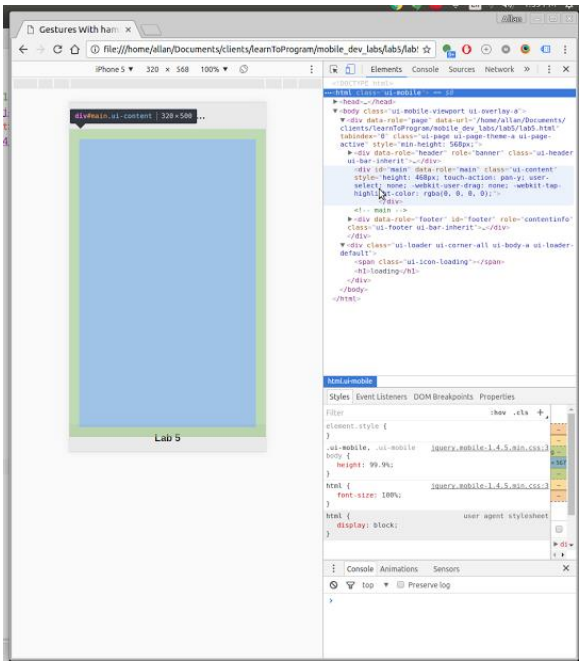
5. Inside the ‘window.onload’ function, detect the height of the screen using window.innerHeight and save it to a variable. Subtract 100 from the height to account for the height of the header and the footer.

```
1.
      <script src="hammer-
   time.min.js"></script>
2.    <script >
3.      window.onload = function(){
4.        var h = window.innerHeight -
   100;
5.      }
6.    </script>
7.    <style>
```

6. Next, set the height of the main content div to the height of the screen.

```
1.   <script src="hammer-
   time.min.js"></script>
2.      <script >
3.        window.onload = function(){
4.          var h = window.innerHeight -
    100;
5.          document.getElementById("mai
   n").style.height = h+"px";
6.        }
7.      </script>
8.      <style>
```

7. To test that this works, you can load the page in your browser, open a console, and hover over the “main” div in the elements tree. In the browser, the main content section should be highlighted, and should cover the full height of the screen from the header to the footer.



8. Inside the ‘window.onload’ function, after the height is set, make a variable called main, and assign to it the main content div.

Initiate the Hammer.JS library, passing in the ‘main’ variable.

```
        document.getElementById("main").style.height = h+"px";

    var main = document.getElementById('main');
    var hammertime = new Hammer(main);
  }
</script>
```

9. Use the ‘Hammer.on’ method to create an event that responds to both tap and press.

```

document.getElementById("main").style.height = h+"px";

        var main = document.getElementById('main');
        var hammertime = new Hammer(main);
        hammertime.on("tap press", function(){});
      }
    </script>
```

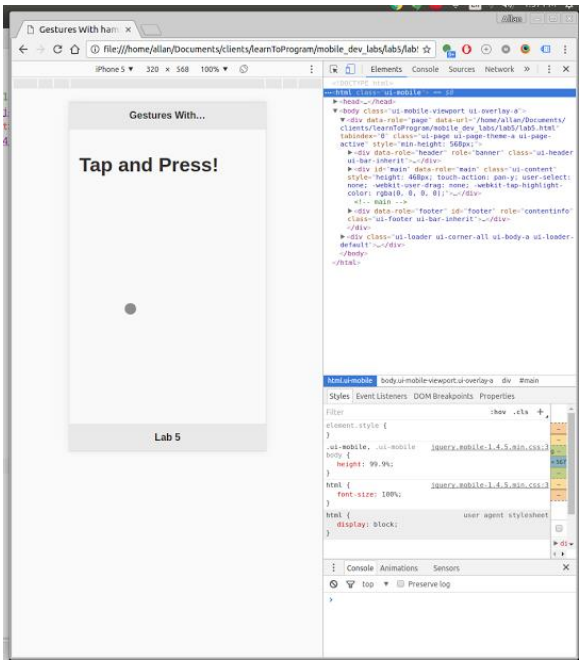
10. Inside the tap/press function, Set the innerHTML of the main content div to a header with a message of your choice.

```
 document.getElementById("main").style.height = h+"px";

  var main = document.getElementById('main');
  var hammertime = new Hammer(main);

  hammertime.on("tap press", function(){
    main.innerHTML = "<h1>Tap and Press!</h1>";
  });
}
t;/script>
```

11. To test out your new feature, open your HTML file in the chrome emulator and simulate a tap or press by clicking on the screen. You should see your message appear on the screen.



Challenge yourself:

- Enter a different message on the screen depending on whether the screen is tapped or pressed.

12. Let’s add a swipe event. Use the ‘Hammer.on’ function to create an event that responds to swipes in the left and right directions.

```

hammertime.on("tap press", function(event){
        main.innerHTML = "<h1>Tap and Press!</h1>";
      });
```

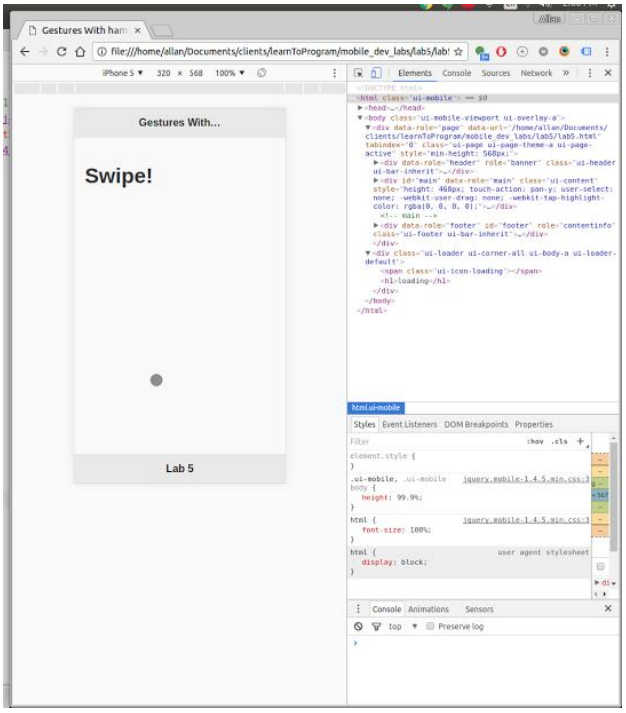
```
4.        hammertime.on(“swipe_left swip
    e_right",function(){});
5.      }
```

14. Inside the on swipe function, set the main innerHTML to a header with a message of your choice.

```
1.   hammertime.on("tap press", function
    (event){
2.            main.innerHTML = "<h1>Tap
    and Press!</h1>";
3.          });
4.          hammertime.on("swipeleft swi
    peright",function(){
5.  main.innerHTML = "<h1>Swipe!</h1>";

6.          });
7.        }
```

15. To test out your new feature, open your HTML file in the chrome emulator. Quickly click, drag and release the mouse in the left or right direction to simulate a swipe. You should see your message appear on the screen.



Challenge Yourself:

- Replace your swipe message with a message that responds to a pan event.