Hands-On Projects

Hands-On Project 10-1

In this project, you'll add support for touch events to a puzzle app in which users drag tiles to form a picture.

1. In your text editor, open the **script.js** file from the HandsOnProject 10-1 folder in the JS_Week_Two folder, add your name and today's date where indicated in the comment section, and then save the file.

2. In the `setUpPage()` function, below the statement that sets an event listener for the `mousedown` event, enter the following statement:

```
puzzlePieces[i].addEventListener("touchstart", startDrag, false);
```

3. Within the `startDrag()` function, after the statement `event.preventDefault()`, add an `if` statement that checks if `event.type` is not equal to `mousedown`, and if so executes the following two statements:

```
this.addEventListener("touchmove", moveDrag, false);
this.addEventListener("touchend", removeTouchListener, false);
```

4. Below the code you entered in the previous step, enclose the code creating event listeners for the `mousemove` and `mouseup` events in an `else` statement. Your `if/else` statement should match the following:

```
if (event.type !== "mousedown") {
this.addEventListener("touchmove", moveDrag, false);
this.addEventListener("touchend", removeTouchListener,
false);
} else {
this.addEventListener("mousemove", moveDrag, false);
this.addEventListener("mouseup", removeDragListener,
false);
}
```

5. Within the `getCoords()` function, below the statement `var coords = []`, add an `if` statement that checks both `event.targetTouches` and `event.targetTouches .length` for truthy values, and if they are present, executes the following statements:

```
var thisTouch = event.targetTouches[0];
coords[0] = thisTouch.clientX;
coords[1] = thisTouch.clientY;
```

6. Below the code you entered in the previous step, enclose the statements assigning values to `coords[0]` and `coords[1]` in an `else` statement. Your `if/else` statement should match the following:

```
if (event.targetTouches && event.targetTouches.length) {
var thisTouch = event.targetTouches[0];
coords[0] = thisTouch.clientX;
coords[1] = thisTouch.clientY;
} else {
coords[0] = event.clientX;
coords[1] = event.clientY;
}
```

7. Below the `removeDragListener()` function, add the following `removeTouchListener()` function:

```
// remove touch event listeners when dragging ends
function removeTouchListener() {
this.removeEventListener("touchmove", moveDrag, false);
this.removeEventListener("touchend", removeTouchListener,
```

```
5 false);
6 }
```

8. Save your changes to **script.js**, open **index.htm** in a desktop browser, and then drag one of the tiles at the bottom to the grid to verify that mouse events still work. Figure 10-18 shows the app in a desktop browser.

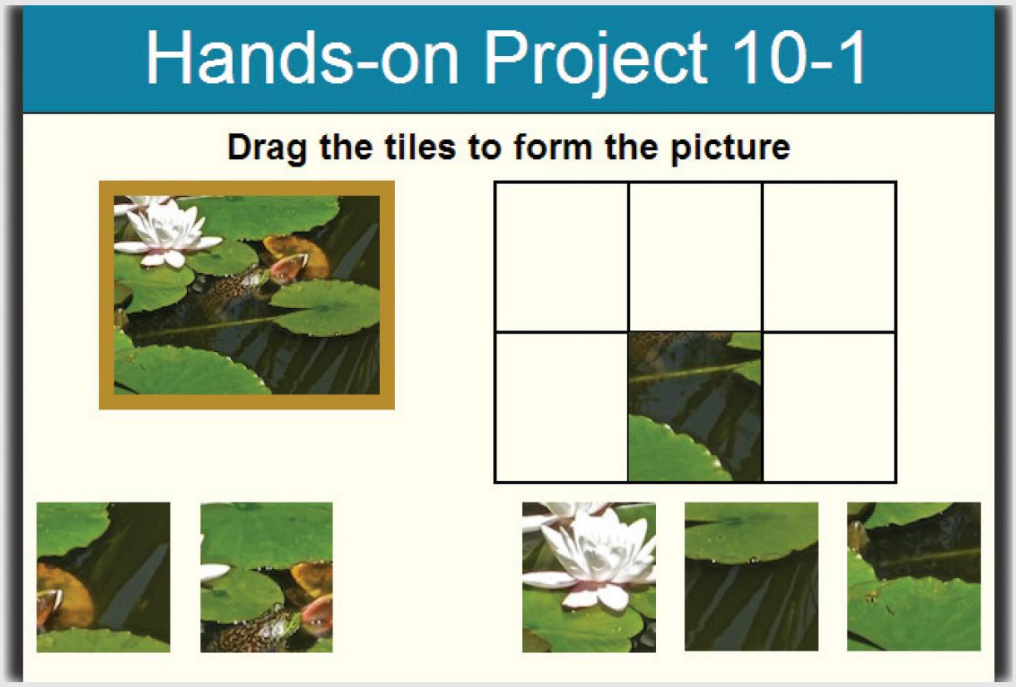

Figure 10-18: index.htm in a desktop browser

9. Open **index.htm** in a mobile browser that uses touch events, such as Safari Mobile or Chrome Mobile, and then drag one of the tiles at the bottom to the grid to verify that touch events now work. Figure 10-19 shows the app in a handheld browser. Note that if you're using a touch device running Windows 8, the page will scroll at the same time that the tiles move. If you complete Hands-on Project 10-2 , you'll enhance the app to make it work better on these devices.

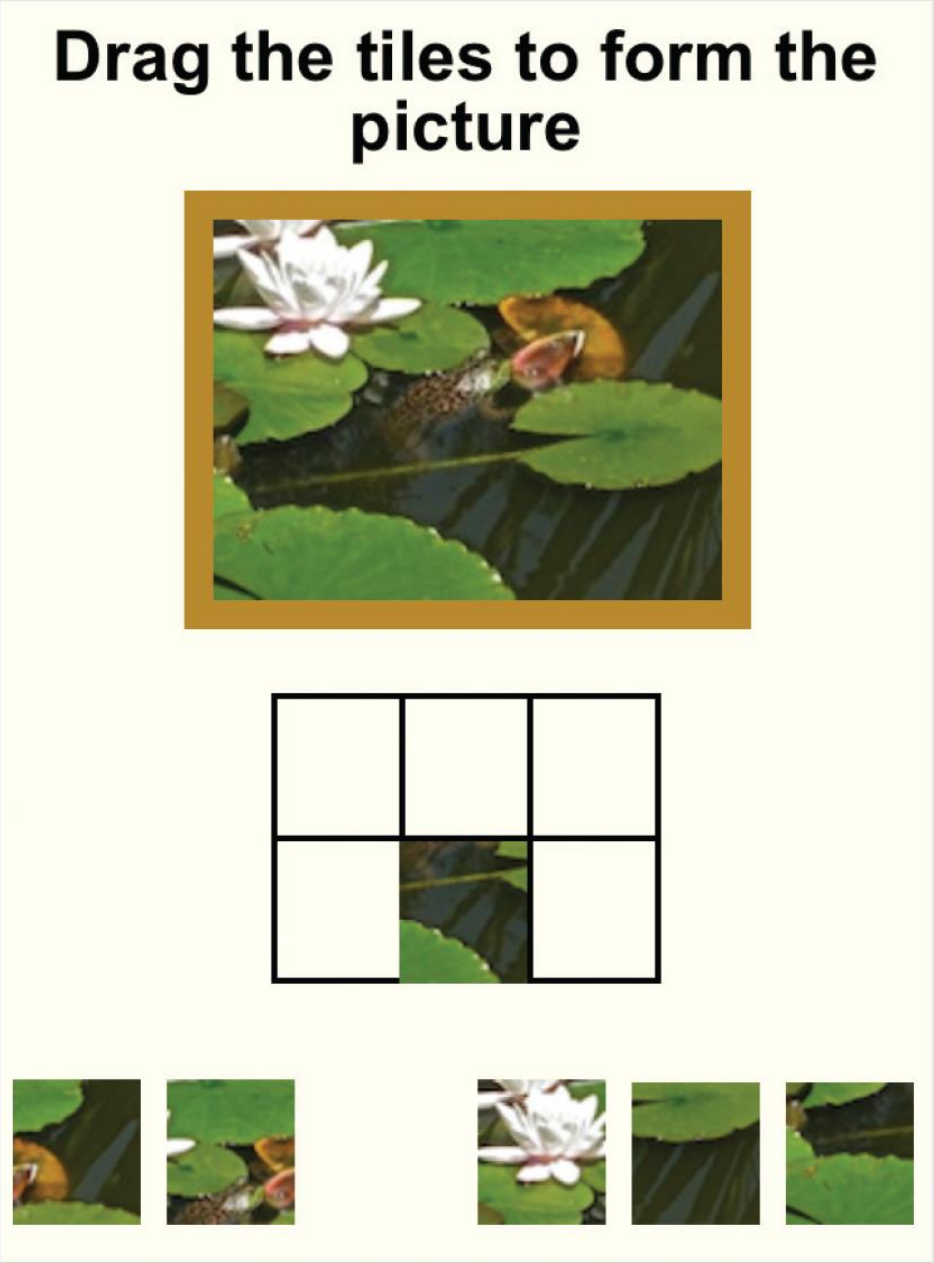

Figure 10-19: index.htm in a handheld browser

Hands-On Project 10-2

In this project, you'll enhance the app from Hands-on Project 10-2 to support pointer events.

1. In the file manager for your operating system, copy the completed contents of the HandsOnProject 10-1 folder to the HandsOnProject 10-2 folder.

2. In your text editor, open the **script.js** file from the HandsOnProject 10-2 folder, change "Hands-on Project 10-1" to **Hands-on Project 10-2** in the comment section, and then save your changes.

3. In the **script.js** file, in the `setUpPage()` function, within the `for` statement, just before the `if` statement, add the following statements to disable interface gestures for IE versions 10 and higher:

```
// disable IE10+ interface gestures
puzzlePieces[i].style.msTouchAction = "none";
puzzlePieces[i].style.touchAction = "none";
```

4. Within the `if` statement, just before the closing `}`, enter the following statements:

```
puzzlePieces[i].addEventListener("mspointerdown", startDrag,
false);
puzzlePieces[i].addEventListener("pointerdown", startDrag,
false);
```

5. In the `startDrag()` function, below the statement setting an event listener for the `touchmove` event, add the following statements:

```
this.addEventListener("mspointermove", moveDrag, false);
this.addEventListener("pointermove", moveDrag, false);
```

6. In the `startDrag()` function, below the statement setting an event listener for the `touchend` event, add the following statements:

```
this.addEventListener("mspointerup", removeTouchListener,
false);
this.addEventListener("pointerup", removeTouchListener,
false);
```

7. In the `removeTouchListener()` function, below the statement removing an event listener for the `touchmove` event, add the following statements:

```
this.removeEventListener("mspointermove", moveDrag, false);
this.removeEventListener("pointermove", moveDrag, false);
```

8. In the `removeTouchListener()` function, below the statement removing an event listener for the `touchend` event, add the following statements:

```
this.removeEventListener("mspointerup", removeTouchListener,
false);
this.removeEventListener("pointerup", removeTouchListener,
false);
```

9. Open **index.htm** in a browser that uses pointer events, such as Internet Explorer 10 version 10 or later on a touchscreen device, and then drag one of the tiles at the bottom to the grid to verify that pointer events now work.

Hands-On Project 10-3

In this project, you'll add functionality to an app to get the user's current position and display it on a map.

1. In your text editor, open the **index.htm** file from the HandsOnProject 10-3 folder, add your name and today's date where indicated in the comment section, and then save the file. Repeat for **script.js**.

2. In the **script.js** file, enter code to create a function with the name `createMap()`. Add a parameter named `position` to the function declaration. Within the function:

a. Declare variables named `Lat` and `Lng`.
b. Set the value of `Lat` to the `latitude` value of the `coords` property of the `position` argument, and set `Lng` to the corresponding `longitude` value.
c. Declare a variable named `mapOptions`, and set its value to an object containing the following properties:

```
center: new google.maps.LatLng(Lat, Lng),zoom: 10
```

d. Declare a variable named `map`, and set its value to the following:

```
new google.maps.Map(document.getElementById("map"),
mapOptions)
```

3. Below the `createMap()` function, declare a new function named `fail()`. The function should contain a single statement that sets the `innerHTML` value of the element with the `id` value `map` to the string **Unable to access your current location**.

4. Below the `use strict` statement, declare a new function named `geoTest()`. It should contain an `if/else` statement that checks the value of the `navigator .geolocation` property. If the value is truthy, the function should call the `getCurrentPosition()` method, running the `createMap()` function on success or the `fail()` function on failure. If the `navigator.geolocation` value is falsy, the function should call the `fail()` function.

5. Below the use strict statement, define a global variable named `waitForUser`. Within the `geoTest()` function, set the value of `waitForUser` to a timeout that waits 10 seconds and then calls the `fail()` function. Within the `createMap()` function, add a statement that clears the `waitForUser` timeout.

6. Save your changes to the **script.js** file, return to the **index.htm** file in your editor, and then just above the existing script element at the bottom of the body section, add the following script element for the Google Maps API:

```
<script src="https://maps.googleapis.com/maps/api/js?
v=3.exp&sensor=true&callback=geoTest"></script>
```

7. Save your changes to the **index.htm** file, open the **index.htm** file in your browser, allow your app to access your current position, and verify that a map is displayed showing your current location, as shown in Figure 10-20. (*Note*: Your location will not necessarily match the one shown in Figure 10-20.) Repeat the test, denying permission to access your location, and verify that the failure message you specified is displayed instead. Note that the app includes buttons with the names of three cities, which you'll use to enhance the app in a later project.

Figure 10-20: Map displaying current location

Source: Google.

Hands-On Project 10-4

In this project, you'll enhance the app you worked on in Hands-on Project 10-3 with timeouts. You'll also geocode the three cities listed at the top of the screen.

1. In the file manager for your operating system, copy the completed contents of the HandsOnProject 10-3 folder to the HandsOnProject 10-4 folder.

2. In your text editor, open the **index.htm** file from the HandsOnProject 10-4 folder, change "Hands-on Project 10-3" to "Hands-on Project 10-4" in the comment section, in the `title` element, and in the `h1` element, and then save your changes.

3. Open the **script.js** file, and then in the comment section, change "Hands-on Project 10-3" to **Hands-on Project 10-4**.

4. In the **script.js** file, within the `geoTest()` function, add a 10-second timeout to the `getCurrentPosition()` call.

5. Below the `use strict` statement, declare a global variable named `waitForUser`.

6. At the start of the `geoTest()` function, use the `waitForUser` variable to set a 10-second timeout using the `setTimeout()` method. The timeout should call the `fail()` function.

7. In the `createMap()` function, just after the declaration of the `Lat` and `Lng` variables, add a statement that clears the `waitForUser` timeout.

8. Save your changes to script.js, and then in your browser open *maps.google.com* or another geocoding service.

9. Geocode a point in downtown Beijing, China, and then save the coordinates in the **script.js** file in a comment in the `createMap()` function. Repeat for a point in downtown Paris, France, and a point in downtown Rio de Janeiro, Brazil.

10. Save your changes to the **script.js** file.

Hands-On Project 10-5

In this project, you'll further enhance the app you've worked on in Hands-on Projects 10-3 and 10-4. You'll incorporate code that displays a map of each city listed at the top of the screen when you click its button.

1. In the file manager for your operating system, copy the completed contents of the HandsOnProject 10-4 folder to the HandsOnProject 10-5 folder.

2. In your text editor, open the **index.htm** file from the HandsOnProject 10-5 folder, change "Hands-on Project 10-4" to "Hands-on Project 10-5" in the comment section, in the `title` element, and in the `h1` element, and then save your changes.

3. Open the **script.js** file, and then in the comment section, change "Hands-on Project 10-4" to **Hands-on Project 10-5**.

4. In the **script.js** file, in the `createMap()` function, below the `clearTimeout()` statement, enclose the two statements assigning values to the `Lat` and `Lng` variables within an `if` statement that checks the `coords` property of the `position` parameter for a truthy value. After the closing `}` for the `if` statement, add an `else` clause that does the following:

    a. Declares a `city` variable and sets its value to `this.innerHTML`.
    b. Checks if the value of `city` is "Beijing" and if so, sets the values of `Lat` and `Lng` to the geocoding results for Beijing that you stored in a comment.
    c. Repeats Step b for "Paris" and "Rio de Janeiro".
    d. Sets the `innerHTML` value of the element with the `id` value `caption` to the value of the `city` variable.

5. At the top of the document, just below the global variable declaration, add a new function named `setUpPage()`. Within the function, add code to do the following:
    a. Declare a variable named `buttons` that references all the `div` elements within the element with the `id` value `cities`.
    b. For each element referenced by the `buttons` variable, add an event listener that responds to the `click` event and calls the `createMap()` function.

6. At the bottom of the document, add an event listener that calls the `setUpPage()` function when the app finishes loading in a browser.

7. Save your changes to the **script.js** file, open the **index.htm** file in your browser, and then allow your browser to access your location if necessary. Verify that a map showing your current location is still displayed. Then click each of the three buttons at the top of the app, and verify that each one displays a map showing that city and changes the caption text above the map to the name of the city displayed, as shown in Figure 10-21.

Figure 10-21: Caption displaying name of city

Source: Google.