Objectives

When you complete this chapter, you will be able to:

- Integrate mouse, touch, and pointer events into a web app
- Obtain and work with a user's geolocation information
- Optimize a mobile web app to accommodate the common constraints experienced by mobile users

Using Touch Events and Pointer Events

- On touchscreen device without a mouse
  - browsers fire click event when user touches screen
- Other events don't translate neatly for touchscreens

Creating a Drag-and Drop Application with Mouse Events

- Mouse events
  - events based on actions of mouse or touchpad

<table>
<thead>
<tr>
<th>EVENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>mousedown</td>
<td>A user presses the mouse button</td>
</tr>
<tr>
<td>mouseup</td>
<td>A user releases the mouse button</td>
</tr>
<tr>
<td>click</td>
<td>A user clicks an element, equivalent to mousedown followed by mouseup</td>
</tr>
<tr>
<td>mousemove</td>
<td>A user moves the mouse pointer</td>
</tr>
<tr>
<td>mouseover</td>
<td>A user moves the mouse pointer within an element</td>
</tr>
<tr>
<td>mouseout</td>
<td>A user moves the mouse pointer off of an element</td>
</tr>
</tbody>
</table>

Table 10-1 Mouse events

Understanding Mouse Events on a Touchscreen Device (cont'd.)

- Touch cascade
  - Browser checks a touched element for an event handler for multiple events
  - Including some mouse events

Figure 10-5 Touch cascade order
Understanding Mouse Events on a Touchscreen Device (cont'd.)

- Touchscreen devices fire touchscreen-specific events for some elements
  - Touch events created by Apple
    - Used on Apple iOS and Google Android
  - Pointer events created by Microsoft
    - Used on Windows Phone and Windows 8 and higher

Implementing Touch Events

- Respond to user's finger touches on a touchscreen

<table>
<thead>
<tr>
<th>EVENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>touchstart</td>
<td>A user places a finger on the screen</td>
</tr>
<tr>
<td>touchmove</td>
<td>A user moves a finger on the screen</td>
</tr>
<tr>
<td>touchend</td>
<td>A user removes a finger from the screen</td>
</tr>
<tr>
<td>touchcancel</td>
<td>A user moves a finger out of the browser window,</td>
</tr>
<tr>
<td></td>
<td>or the interface or app cancels the touch</td>
</tr>
</tbody>
</table>

Table 10-2 Touch events

Implementing Touch Events (cont'd.)

- touchstart event
  - analogous tomousedown mouse event
- touchmove event
  - corresponds tomousemove
- touchend event
  - similar tomouseup
- touchcancel event
  - unique to touchscreen

Implementing Touch Events (cont'd.)

- Working with Touch Coordinates
  - Mouse events can work with event properties
    - clientX property = x coordinate of event
    - clientY property = y coordinate of event
  - Touch events support multitouch devices
    - Allow for multiple touches on screen at once
    - Don't support clientX or clientY properties as direct children
    - Each touch event has array properties

Implementing Touch Events (cont'd.)

- Distinguishing Between App and Device Interaction
  - Touchscreen devices use touch events for more than one purpose
    - Can interact via touch with an app
    - Use touch to perform gestures
      - Browser and device interactions like scrolling
    - Use preventDefault() method
      - Ensures that OS interface doesn't respond to events when users interact with your app
Implementing Touch Events (cont'd.)

Figure 10-7 Multiple uses of a single touch event

Implementing Pointer Events

- Touchscreens on new types of devices
  - Tablets
  - Notebook computers
- Makes coding for touch and mouse events more complicated
  - Some devices support stylus input
  - Some devices have trackpads

Implementing Pointer Events (cont'd.)

- Microsoft pointer events
  - Aim to handle input from mouse, finger, or stylus with each event
  - Incorporate other event properties
    - Pressure on screen
    - Angle of stylus
- Only IE Mobile and IE 10 and later support pointer events
- Some versions of IE do not recognize touch events
  - Use mouse+touch+pointer for max. compatibility

Table 10-4 Pointer events

Implementing Pointer Events (cont'd.)

- Identifying pointer screen coordinates
  - clientX and clientY properties like mouse events
- Stopping OS gestures
  - Requires setting msTouchAction CSS property
    - Set value to none

Using Programming Interfaces for Mobile Devices

- APIs available to access information provided by mobile device hardware

<table>
<thead>
<tr>
<th>API</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geolocation</td>
<td>Provides user's latitude and longitude coordinates (user opt-in required)</td>
</tr>
<tr>
<td>Battery Status</td>
<td>Reports charge level of device battery</td>
</tr>
<tr>
<td>Device Orientation</td>
<td>Provides access to device orientation and changes in orientation</td>
</tr>
<tr>
<td>WebRTC</td>
<td>Provides access to device camera, microphone, and/or screen (user opt-in required for all)</td>
</tr>
</tbody>
</table>

Table 10-5 Selected hardware APIs for mobile devices
Using the Geolocation API

- Provides access to user's latitude & longitude
- Accessed using `geolocation` property of `Navigator` object

**Table 10-5** Selected hardware APIs for mobile devices

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getCurrentPosition()</code></td>
<td>Provides current position of device, subject to user authentication, where <code>success</code> &amp; <code>error</code> to call if the request is successful. <code>fail</code> is code to run if the request fails, and <code>options</code> represents one or more optional parameters.</td>
</tr>
<tr>
<td><code>watchPosition()</code></td>
<td>Provides current position of device, and continues to monitor position, providing updated position when position changes.</td>
</tr>
<tr>
<td><code>clearWatch()</code></td>
<td>Stops monitoring position, where <code>number</code> is the number returned by the original <code>watchPosition()</code> statement.</td>
</tr>
</tbody>
</table>

Using the Geolocation API (cont'd.)

- Callbacks
  - Arguments that contain reference executable code
- `getCurrentPosition()` method
  - Request user's position a single time

**Table 10-7** Properties passed on a successful geolocation request

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude</td>
<td>Geographic latitude, in degrees</td>
</tr>
<tr>
<td>longitude</td>
<td>Geographic longitude, in degrees</td>
</tr>
<tr>
<td>altitude</td>
<td>Elevation, in meters</td>
</tr>
<tr>
<td>accuracy</td>
<td>Accuracy of latitude and longitude values, in meters</td>
</tr>
<tr>
<td>altitudeAccuracy</td>
<td>Accuracy of altitude value, in meters</td>
</tr>
<tr>
<td>heading</td>
<td>Direction of travel, in degrees</td>
</tr>
<tr>
<td>speed</td>
<td>Current speed, in meters per second</td>
</tr>
</tbody>
</table>

Using the Geolocation API (cont'd.)

- Basic example:
  ```javascript
  navigator.geolocation.getCurrentPosition(showLocation);
  function showLocation(position) {
    console.log("Longitude: " + position.coords.longitude);
    console.log("Latitude: " + position.coords.latitude);
  }
  ```

Using the Geolocation API (cont'd.)

- Enhanced example:
  ```javascript
  navigator.geolocation.getCurrentPosition(showLocation, fail, {timeout: 10000});
  function showLocation(position) {
    console.log("Longitude: " + position.coords.longitude);
    console.log("Latitude: " + position.coords.latitude);
  }
  function fail() {
    var content = document.getElementById("mainParagraph");
    content.innerHTML = "<p>Geolocation information not available or not authorized.</p>";
  }
  ```

Using the Geolocation API (cont'd.)

- Need to fail gracefully in older browsers
  ```javascript
  navigator.geolocation.getCurrentPosition(createDirections, fail, {timeout: 10000});
  ```
  ```javascript
  fail();
  ```
Using the Geolocation API (cont'd.)

• Need to clear geolocation history for testing

<table>
<thead>
<tr>
<th>BROWSER</th>
<th>STEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>Click the Tools button, click Internet options, click the Privacy tab, then at the Location section click Clear Sites, and then click OK.</td>
</tr>
<tr>
<td>Chrome</td>
<td>With the page open in Chrome, click the View site information button to the left of the URL in the Location bar, then in the Permissions section click the list box for Location, click Use global default (Ask), and then click in the browser window outside of the dialog box.</td>
</tr>
<tr>
<td>Firefox</td>
<td>With the page open in Firefox, click the icon to the left of the URL in the Location bar, click More information, click Permissions, then in the Access Your Location section check Use Default Box, and then close the dialog box.</td>
</tr>
</tbody>
</table>

Table 10-9 Steps to clear saved geolocation history

Using the Geolocation API (cont'd.)

• Sometimes users don’t notice or ignore geolocation request
  – Request neither fails or succeeds
  – Any dependent code not executed

Using the Geolocation API (cont'd.)

• Can handle lack of yes/no response from user
  – setTimeout() method
  – Start countdown before request
    • If timeout expires, stop waiting and trigger failure code
    • If user responds, cancel timeout

Using the Geolocation API (cont'd.)

• Code complete to acquire geolocation information
  – Then you can integrate with databases
• Using the Google Maps API
  – Can display a map centered on user’s location
  – Can show route/directions between two locations
  – Includes 2 constructors
    • Map() creates a map object
      var name = new google.maps.Map(element, options);
    • LatLng() creates an object containing coordinates
      center: new google.maps.LatLng(latitude, longitude)
Using the Geolocation API (cont'd.)

- Using the Google Maps API (cont'd.)
  - Example—create new map centered on current position with zoom of 11:

```javascript
var currPosLat = position.coords.latitude;
var currPosLng = position.coords.longitude;
var mapOptions = {
  center: new google.maps.LatLng(currPosLat, currPosLng),
  zoom: 11
};
var map = new google.maps.Map(document.getElementById("map"));
```

Using the Geolocation API (cont'd.)

- Using the Google Maps API (cont'd.)
  - Can also create map centered on specified point
  - Geocoding
    - Converting physical address to latitude/longitude coordinates

Using the Battery Status API

- Adds a `battery` property to `Navigator` object

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charging</td>
<td>boolean true if device indicates device is charging</td>
</tr>
<tr>
<td>chargingTime</td>
<td>time until battery will be fully charged (in seconds)</td>
</tr>
<tr>
<td>dischargingTime</td>
<td>time until power will be fully drained (in seconds)</td>
</tr>
<tr>
<td>latency</td>
<td>current level of charge as a number from 0 to 1</td>
</tr>
</tbody>
</table>

Table 10-10 Properties of the `battery` object

Using the Device Orientation API

- Provides access to changes in position and speed
  - Gyroscope
    - Device hardware that detects orientation in space
      - `deviceorientation` event
        - alpha, beta, and gamma coordinate properties
  - Accelerometer
    - Device hardware that detects changes in speed
      - `devicemotion` event
        - reports values for acceleration and rotation

Using the WebRTC API

- Short for web real-time communication
- Enables apps to
  - receive data from device camera and microphone
  - send and receive audio/video/data in real time
- Should eventually allow text/video chat using only HTML and JavaScript

Enhancing Mobile Web Apps

- Testing tools
  - Often impractical to collect many mobile devices
  - Online services available for testing
  - Free testing apps from mobile OS makers:

<table>
<thead>
<tr>
<th>Company</th>
<th>Operating System</th>
<th>Simulation Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>iOS</td>
<td>Kindle</td>
</tr>
<tr>
<td>Google</td>
<td>Android</td>
<td>Android SDK</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Windows Phone</td>
<td>Windows Phone SDK</td>
</tr>
</tbody>
</table>

Table 10-12 Software used to simulate mobile devices
Enhancing Mobile Web Apps (cont'd.)

• Minimizing Download Size
  – Mobile speeds usually slower
  – Mobile users often have data caps

Minimizing Download Size (cont'd.)

– Loading Scripts Responsively

Figure 10-14 Implementing responsive script loading for oaktop.htm

Enhancing Mobile Web Apps (cont'd.)

• Minifying Files
  – Removes comments, indents, and line breaks
  – Tweaks code in other ways to make it smaller
  – Online minifying services available

Minifying Files

Summary

• Touch events focus on responding to a user's finger touches on a touchscreen
• To ensure OS interface doesn't respond to gestures
  – Use the preventDefault() method
• Pointer events are different than touch events
  – Aim to handle input from mouse, finger, or stylus
• Geolocation API provides access to a user's latitude and longitude coordinates
• A number of tools exist for testing mobile web apps virtually

Summary (cont'd.)

• Important to minimize download size of mobile web app
  – Load scripts responsively
  – Minify files