COURSE TECHNOLOGY
CENGAGE Learning

JavaScript, Sixth Edition

Chapter 10
Programming for Touchscreens and Mobile Devices

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 moust
  - 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

EVENT	DESCRIPTION
`mousedown`	A user presses the mouse button
`mouseup`	A user releases the mouse button
`click`	A user clicks an element; equivalent to `mousedown` followed by `mouseup`
`mousemove`	A user moves the mouse pointer
`mouseover`	A user moves the mouse pointer within an element
`mouseout`	A user moves the mouse pointer off of an element

Table 10-1 Mouse events

Understanding Mouse Events on a Touchscreen Device

- On a touchscreen device
  - Some web page elements respond to mouse events
  - `div` element not considered clickable
    - doesn't fire mouse events on touchscreen
  - links and form elements are clickable
    - browsers respond to touch
- Touch cascade
  - Browser checks a touched element for an event handler for multiple events
    - Including some mouse events

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

```
touchstart
touchend
mouseover
mousemove
mousedown
mouseup
click
```

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

EVENT	DESCRIPTION
`touchstart`	A user places a finger on the screen
`touchmove`	A user moves a finger on the screen
`touchend`	A user removes a finger from the screen
`touchcancel`	A user moves a finger out of the browser window, or the interface or app cancels the touch

Table 10-2 Touch events

Implementing Touch Events (cont'd.)

- `touchstart` event
  - analogous to `mousedown` mouse event
- `touchmove` event
  - corresponds to `mousemove`
- `touchend` event
  - similar to `mouseup`
- `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.)

ARRAY	CONTENTS
`touches`	Coordinates of all touches currently on the screen
`targetTouches`	Coordinates of all touches on the current element
`changedTouches`	Coordinates of all touches involved in the event

Table 10-3 Array properties of touch events

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.)

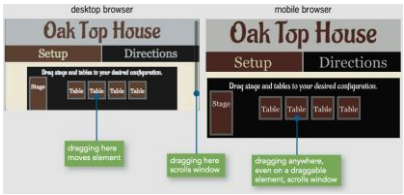

Figure 10-7 Multiple uses of a single touch event

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

Implementing Pointer Events (cont'd.)

EVENT	FINGER/STYLUS DESCRIPTION	MOUSE DESCRIPTION
`pointerdown`	A touch is initiated over the element	The mouse button is pressed while the pointer is over the element
`pointerup`	A touch is ended over the element	The mouse button is released while the pointer is over the element
`pointercancel`	An active touch or click is cancelled by the app or the interface	
`pointermove`	An active touch or active mouse pointer moves	
`pointerover`	An active touch or click moves within the boundaries of an element	
`pointerout`	An active touch or active mouse pointer leaves the boundaries of an element	
`pointerenter`	An active touch or active mouse pointer moves within the boundaries of an element or one of its descendant elements	
`pointerleave`	An active touch or active mouse pointer leaves the boundaries of an element and all of its descendant elements	

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

API	PURPOSE
Geolocation	Provides user's latitude and longitude coordinates (user opt-in required)
Battery Status	Reports charge level of device battery
Device Orientation	Provides access to device orientation and changes in orientation
WebRTC	Provides access to device camera, microphone, and/or screen (user opt-in required for all)

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

METHOD	DESCRIPTION
`getCurrentPosition(success [, fail, options])`	Provides current position of device, subject to user authorization, where `success` is code to run if the request is successful, `fail` is code to run if the request fails, and `options` represents one or more optional parameters
`watchPosition(success [, fail, options])`	Provides current position of device, and continues to monitor position, providing updated position when position changes
`clearWatch(number)`	Stops monitoring position, where `number` is the number returned by the original `watchPosition()` statement

Table 10-5 Selected hardware APIs for mobile devices

Using the Geolocation API (cont'd.)

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

Using the Geolocation API (cont'd.)

PROPERTY	DESCRIPTION
`latitude`	Geographic latitude, in degrees
`longitude`	Geographic longitude, in degrees
`altitude`	Elevation, in meters
`accuracy`	Accuracy of latitude and longitude values, in meters
`altitudeAccuracy`	Accuracy of altitude value, in meters
`heading`	Direction of travel, in degrees
`speed`	Current speed, in meters per second

Table 10-7 Properties passed on a successful geolocation request

Using the Geolocation API (cont'd.)

- Basic example:

```
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
  - Handle failed request, set 10-second timeout:

```
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

```
if (navigator.geolocation {
  navigator.geolocation.getCurrentPosition(createDirections,↵
    fail, {timeout: 10000});
} else {
  fail();
}
```

Using the Geolocation API (cont'd.)

- Need to clear geolocation history for testing

BROWSER	STEPS
Internet Explorer	Click the **Tools** button, click **Internet options**, click the **Privacy** tab, then in the Location section click **Clear Sites**, and then click **OK**
Chrome	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
Firefox	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 the **Use Default** box, and then close the dialog box.

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.)

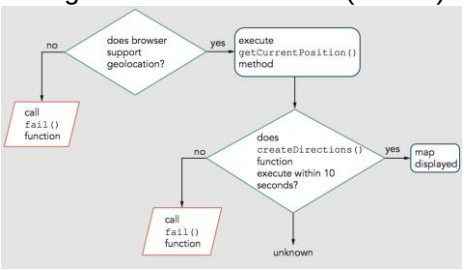

Figure 10-10 Flowchart illustrating flow of current geolocation code

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.)

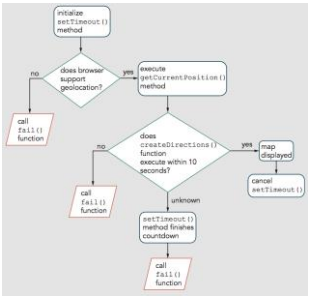

Figure 10-11 Flowchart illustrating geolocation code that incorporates `setTimeout()`

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:

```
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"),↵
  mapOptions);
```


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

PROPERTY	DESCRIPTION
`charging`	Boolean with a value of `true` if the device is charging
`chargingTime`	Time until battery is fully charged (in seconds)
`dischargingTime`	Time until system will be shut down due to low battery (in seconds)
`level`	Current level of charge, on a scale from 0 to 1

Table 10-10 Properties of the `battery` object

EVENT	DESCRIPTION
`chargingchange`	The value of the `charging` property changes
`chargingtimechange`	The value of the `chargingTime` property changes
`dischargingtimechange`	The value of the `dischargingTime` property changes
`levelchange`	The value of the `level` property changes

Table 10-11 Events 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:

COMPANY	OPERATING SYSTEM	SIMULATION SOFTWARE
Apple	iOS	Xcode
Google	Android	Android SDK
Microsoft	Windows Phone	Windows Phone SDK

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

Enhancing Mobile Web Apps (cont'd.)

- Minimizing Download Size (cont'd.)
  - Loading Scripts Responsively

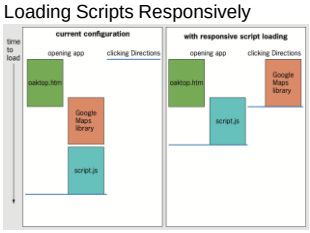

Figure 10-14 Implementing responsive script loading for oaktop.htm

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

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