https://developer.mozilla.org/en-US/docs/Web/API/TouchEvent

Constructor

==========

TouchEvent()
Creates a TouchEvent object.

The **TouchEvent()** constructor creates a new TouchEvent.

Syntax

```
event = new TouchEvent(typeArg, touchEventInit);
```

Values

typeArg
Is a DOMString representing the name of the event.
touchEventInit Optional
Is a TouchEventInit dictionary, having the following fields:

- "touches", optional and defaulting to `[]`, of type `Touch[]`, that is a list of objects for every point of contact currently touching the surface.
- "targetTouches", optional and defaulting to `[]`, of type `Touch[]`, that is a list of objects for every point of contact that is touching the surface *and* started on the element that is the target of the current event.
- "changedTouches", optional and defaulting to `[]`, of type `Touch[]`, that is a list of objects for every point of contact which contributed to the event.
- "ctrlKey", optional and defaulting to false, of type Boolean, that indicates if the ctrl key was simultaneously pressed.
- "shiftKey", optional and defaulting to false, of type Boolean, that indicates if the shift key was simultaneously pressed.
- "altKey", optional and defaulting to false, of type Boolean, that indicates if the alt key was simultaneously pressed.
- "metaKey", optional and defaulting to false, of type Boolean, that indicates if the meta key was simultaneously pressed.

The TouchEventInit dictionary also accepts fields from UIEventInit *and from* EventInit *dictionaries*.

Specifications

Specification	Status	Comment
Touch Events – Level 2 The definition of 'TouchEvent' in that specification.	Editor's Draft	Added TouchEvent constructor.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	48.0 [1]	?	12.0	15.0	?

[1] Chrome only supports the following touchEventInit properties: touches, targetTouches, changeTouches.

See also

- TouchEvent, the interface of the objects it constructs.

Properties

=========

This interface inherits properties from its parent, UIEvent *and* Event.

TouchEvent.altKey

TouchEvent.altKey Read only
A Boolean value indicating whether or not the alt key was down when the touch event was fired.

Summary

A Boolean value indicating whether or not the alt (Alternate) key is enabled when the touch event is created. If the alt key is enabled, the attribute's value is true. Otherwise, it is false.
This property is Read only .

Syntax

```
var altEnabled = touchEvent.altKey;
```

Return value

altEnabled
true if the alt key is enabled for this event; and false if the alt is not enabled.

Example

This example illustrates how to access the TouchEvent key modifier properties: TouchEvent.altKey, TouchEvent.ctrlKey, TouchEvent.metaKey and TouchEvent.shiftKey.
In following code snippet, the touchstart event handler logs the state of the event's modifier keys.

```
someElement.addEventListener('touchstart', function(e) {
   // Log the state of this event's modifier keys
   console.log("altKey = " + e.altKey);
   console.log("ctrlKey = " + e.ctrlKey);
   console.log("metaKey = " + e.metaKey);
   console.log("shiftKey = " + e.shiftKey);
}, false);
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	No support	No support	No support

TouchEvent.changedTouches

TouchEvent.changedTouches Read only
A TouchList of all the Touch objects representing individual points of contact whose states changed between the previous touch event and this one.

Summary

A TouchList whose touch points (Touch objects) varies depending on the event type, as follows:

- For the touchstart event, it is a list of the touch points that became active with the current event.
- For the touchmove event, it is a list of the touch points that have changed since the last event.
- For the touchend event, it is a list of the touch points that have been removed from the surface (that is, the set of touch points corresponding to fingers no longer touching the surface).

This property is Read only .

Syntax

```
var changes = touchEvent.changedTouches;
```

Return value

changes
A TouchList whose Touch objects include all the touch points that contributed to this touch event.

Example

This example illustrates the TouchEvent object's TouchEvent.changedTouches property. The TouchEvent.changedTouches property is a TouchList object that contains one Touch object for each touch point which contributed to the event. In following code snippet, the touchmove event handler iterrates through the changedTouches list and prints the identifier of each touch point that changed since the last event.

```
someElement.addEventListener('touchmove', function(e) {
   // Iterate through the list of touch points that changed
   // since the last event and print each touch point's identifier.
   for (var i=0; i < e.changedTouches.length; i++) {
     console.log("changedTouches[" + i + "].identifier = " + e.changedTou-
ches[i].identifier);
   }
}, false);
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	N

TouchEvent.ctrlKey

TouchEvent.ctrlKey Read only
A Boolean value indicating whether or not the control key was down when the touch event was fired.

Summary

A Boolean value indicating whether the control (Control) key is enabled when the touch event is created. If this key is enabled, the attribute's value is true. Otherwise, it is false.
This property is Read only .

Syntax

```
var ctrlEnabled = touchEvent.ctrlKey;
```

Return value

ctrlEnabled
true if the control key is enabled for this event; and false if the control is not enabled.

Example

The TouchEvent.altKey example includes an example of this property's usage.

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	No support	No support	No support

TouchEvent.metaKey

TouchEvent.metaKey Read only
A Boolean value indicating whether or not the meta key was down when the touch event was fired.

Summary

A Boolean value indicating whether or not the Meta key is enabled when the touch event is created. If this key is enabled, the attribute's value is true. Otherwise, it is false.
This property is Read only .
Note: On Macintosh keyboards, this is the Command key. On Windows keyboards, this is the Windows key ().

Syntax

```
var metaEnabled = touchEvent.metaKey;
```

Return value

metaEnabled
true if the Meta key is enabled for this event; and false if the Meta is not enabled.

Example

The TouchEvent.altKey example includes an example of this property's usage.

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	No support	No support	No support

TouchEvent.shiftKey

TouchEvent.shiftKey Read only
A Boolean value indicating whether or not the shift key was down when the touch event was fired.

Summary

A Boolean value indicating whether or not the shift key is enabled when the touch event is created. If this key is enabled, the attribute's value is true. Otherwise, it is false.
This property is Read only .

Syntax

```
var shiftEnabled = touchEvent.shiftKey;
```

Return value

shiftEnabled
true if the shift key is enabled for this event; and false if the shift key is not enabled.

Example

The TouchEvent.altKey example includes an example of this property's usage.

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	No support	No support	No support

TouchEvent.targetTouches

TouchEvent.targetTouches Read only
A TouchList of all the Touch objects that are both currently in contact with the touch surface **and** were also started on the same element that is the target of the event.

Summary

A TouchList listing all the Touch objects for touch points that are still in contact with the touch surface **and** whose touchstart event occurred inside the same target element as the current target element.
This property is Read only .

Syntax

```
var touches = touchEvent.targetTouches;
```

Return value

touches
A TouchList listing all the Touch objects for touch points that are still in contact with the touch surface **and** whose touchstart event occurred inside the same target element as the current target element.

Example

This example illustrates the TouchEvent object's TouchEvent.targetTouches property. The TouchEvent.targetTouches property is a TouchList object that includes those TPs that are currently touching the surface *and* started on the element that is the target of the current event. As such, the targetTouches list is a strict subset of the touches list.
In following code snippet, the function compares the length of the touches list to the the length of the targetTouches list and returns true if the lengths are the same and returns false otherwise.

```
function touches_in_target(ev) {
  // Return true if all of the touches are within the target element;
  // otherwise return false.
  return (ev.touches.length == ev.targetTouches.length ? true : false);
}
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

TouchEvent.touches

TouchEvent.touches Read only
A TouchList of all the Touch objects representing all current points of contact with the surface, regardless of target or changed status.

Summary

A TouchList listing all the Touch objects for touch points that are currently in contact with the touch surface, regardless of whether or not they've changed or what their target element was at touchstart time.
This property is Read only .

Syntax

```
var touches = touchEvent.touches;
```

Return value

touches
A TouchList listing all the Touch objects for touch points that are still in contact with the touch surface, regardless of whether or not they've changed or what their target element was at touchstart time.

Example

This example illustrates the TouchEvent object's TouchEvent.touches property. The TouchEvent.touches property is a TouchList object and containing a list of Touch objects for every point of contact currently touching the surface.
In following code snippet, the touchstart event handler checks the length of the TouchEvent.touches list to determine the number of touch points that were activated and then invokes different handlers depending on the number of touch points.

```
someElement.addEventListener('touchstart', function(e) {
   // Invoke the appropriate handler depending on the
   // number of touch points.
   switch (e.touches.length) {
     case 1: handle_one_touch(e); break;
     case 2: handle_two_touches(e); break;
     case 3: handle_three_touches(e); break;
     default: console.log("Not supported"); break;
   }
 }, false);
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.
Touch Events	Recommendation	Initial definition.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

Types
=====

There are several types of event that can be fired to indicate that touch-related changes have occurred. You can determine which of these has happened by looking at the event's TouchEvent.type property.
Note: It's important to note that in many cases, both touch and mouse events get sent (in order to let non-touch-specific code still interact with the user). If you use touch events, you should call event.preventDefault() to keep the mouse event from being sent as well.

touchstart

Sent when the user places a touch point on the touch surface. The event's target will be the element in which the touch occurred.

The touchstart event is fired when a touch point is placed on the touch surface.

General info

Specification	Touch Events
Interface	TouchEvent
Bubbles	Yes
Cancelable	Yes
Target	Document, Element
Default Action	undefined

Properties

Property	Type	Description
target Read only	EventTarget	The event target (the topmost target in the DOM tree).
type Read only	DOMString	The type of event.
bubbles Read only	boolean	Does the event normally bubble?
cancelable Read only	boolean	Is it possible to cancel the event?
view Read only	WindowProxy	document.defaultView (window of the document)
detail Read only	long (float)	0.
touches Read only	TouchList	A list of Touches for every point of contact currently touching the surface.
targetTouches Read only	TouchList	A list of Touches for every point of contact that is touching the surface and started on the element that is the target of the current event.
changedTouches Read only	TouchList	A list of Touches for every point of contact which contributed to the event. For the touchstart event this must be a list of the touch points that just became active with the current event. For the touchmove event this must be a list of the touch points that have moved since the last event. For the touchend and touchcancel events this must be a list of the touch points that have just been removed from the surface.
ctrlKey Read only	boolean	true if the control key was down when the event was fired. false otherwise.
shiftKey Read only	boolean	true if the shift key was down when the event was fired. false otherwise.
altKey Read only	boolean	true if the alt key was down when the event was fired. false otherwise.
metaKey Read only	boolean	true if the meta key was down when the event was fired. false otherwise.

Examples

Code samples for those events are available on the dedicated page: Touch events.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

Related Events

- ontouchstart

touchend

Sent when the user removes a touch point from the surface (that is, when they lift a finger or stylus from the surface). This is also sent if the touch point moves off the edge of the surface; for example, if the user's finger slides off the edge of the screen.
The event's target is the same element that received the touchstart event corresponding to the touch point, even if the touch point has moved outside that element.
The touch point (or points) that were removed from the surface can be found in the TouchList specified by the changedTouches attribute.

The touchend event is fired when a touch point is removed from the touch surface.

General info

Specification	Touch Events
Interface	TouchEvent
Bubbles	Yes
Cancelable	Yes
Target	Document, Element
Default Action	undefined

Properties

Property	Type	Description
target Read only	EventTarget	The event target (the topmost target in the DOM tree).
type Read only	DOMString	The type of event.
bubbles Read only	boolean	Does the event normally bubble?
cancelable Read only	boolean	Is it possible to cancel the event?
view Read only	WindowProxy	document.defaultView (window of the document)
detail Read only	long (float)	0.
touches Read only	TouchList	A list of Touches for every point of contact currently touching the surface.
targetTouches Read only	TouchList	A list of Touches for every point of contact that is touching the surface and started on the element that is the target of the current event.
changedTouches Read only	TouchList	A list of Touches for every point of contact which contributed to the event. For the touchstart event this must be a list of the touch points that just became active with the current event. For the touchmove event this must be a list of the touch points that have moved since the last event. For the touchend and touchcancel events this must be a list of the touch points that have just been removed from the surface.
ctrlKey Read only	boolean	true if the control key was down when the event was fired. false otherwise.
shiftKey Read only	boolean	true if the shift key was down when the event was fired. false otherwise.
altKey Read only	boolean	true if the alt key was down when the event was fired. false otherwise.
metaKey Read only	boolean	true if the meta key was down when the event was fired. false otherwise.

Examples

Code samples for those events are available on the dedicated page: Touch events.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

Related Events

- ontouchend

touchmove

Sent when the user moves a touch point along the surface. The event's target is the same element that received the touchstart event corresponding to the touch point, even if the touch point has moved outside that element.
This event is also sent if the values of the radius, rotation angle, or force attributes of a touch point change.
Note: The rate at which touchmove events is sent is browser-specific, and may also vary depending on the capability of the user's hardware. You must not rely on a specific granularity of these events.

The touchmove event is fired when a touch point is moved along the touch surface.

General info

Specification	Touch Events
Interface	TouchEvent
Bubbles	Yes
Cancelable	Yes
Target	Document, Element
Default Action	undefined

Properties

Property	Type	Description
target Read only	EventTarget	The event target (the topmost target in the DOM tree).
type Read only	DOMString	The type of event.
bubbles Read only	boolean	Does the event normally bubble?
cancelable Read only	boolean	Is it possible to cancel the event?
view Read only	WindowProxy	document.defaultView (window of the document)
detail Read only	long (float)	0.
touches Read only	TouchList	A list of Touches for every point of contact currently touching the surface.
targetTouches Read only	TouchList	A list of Touches for every point of contact that is touching the surface and started on the element that is the target of the current event.
changedTouches Read only	TouchList	A list of Touches for every point of contact which contributed to the event. For the touchstart event this must be a list of the touch points that just became active with the current event. For the touchmove event this must be a list of the touch points that have moved since the last event. For the touchend and touchcancel events this must be a list of the touch points that have just been removed from the surface.
ctrlKey Read only	boolean	true if the control key was down when the event was fired. false otherwise.
shiftKey Read only	boolean	true if the shift key was down when the event was fired. false otherwise.
altKey Read only	boolean	true if the alt key was down when the event was fired. false otherwise.
metaKey Read only	boolean	true if the meta key was down when the event was fired. false otherwise.

Examples

Code samples for those events are available on the dedicated page: Touch events.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

touchcancel

Sent when a touch point has been disrupted in some way. There are several possible reasons why this might happen (and the exact reasons will vary from device to device, as well as browser to browser):

- An event of some kind occurred that canceled the touch; this might happen if a modal alert pops up during the interaction.
- The touch point has left the document window and moved into the browser's UI area, a plug-in, or other external content.
- The user has placed more touch points on the screen than can be supported, in which case the earliest Touch in the TouchList gets canceled.

The touchcancel event is fired when a touch point has been disrupted in an implementation-specific manner (for example, too many touch points are created).

General info

Specification	Touch Events
Interface	TouchEvent
Bubbles	Yes
Cancelable	No
Target	Document, Element
Default Action	None

Properties

Property	Type	Description
target Read only	EventTarget	The event target (the topmost target in the DOM tree).
type Read only	DOMString	The type of event.
bubbles Read only	boolean	Does the event normally bubble?
cancelable Read only	boolean	Is it possible to cancel the event?
view Read only	WindowProxy	document.defaultView (window of the document)
detail Read only	long (float)	0.
touches Read only	TouchList	A list of Touches for every point of contact currently touching the surface.
targetTouches Read only	TouchList	A list of Touches for every point of contact that is touching the surface and started on the element that is the target of the current event.
changedTouches Read only	TouchList	A list of Touches for every point of contact which contributed to the event. For the touchstart event this must be a list of the touch points that just became active with the current event. For the touchmove event this must be a list of the touch points that have moved since the last event. For the touchend and touchcancel events this must be a list of the touch points that have just been removed from the surface.
ctrlKey Read only	boolean	true if the control key was down when the event was fired. false otherwise.
shiftKey Read only	boolean	true if the shift key was down when the event was fired. false otherwise.
altKey Read only	boolean	true if the alt key was down when the event was fired. false otherwise.
metaKey Read only	boolean	true if the meta key was down when the event was fired. false otherwise.

Examples

Code samples for those events are available on the dedicated page: Touch events.

Browser compatibility

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

GlobalEventHandlers

==========

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

GlobalEventHandlers.ontouchstart

GlobalEventHandlers.ontouchstart
A global event handler for the touchstart event.

A global event handler for the touchstart event.

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

Note: This attribute has *not* been formally standardized. It is specified in the Touch Events – Level 2 Editor's Draft specification and not in Touch Events Recommendation. This attribute is not widely implemented.

Syntax

```
var startHandler = someElement.ontouchstart;
```

Return value

startHandler
The *touchstart* event handler for element someElement.

Example

This example shows two ways to use *ontouchstart* to set an element's *touchstart* event handler.

```
<html>
<script>
function startTouch(ev) {
 // Process the event
}
function init() {
 var el=document.getElementById("target1");
 el.ontouchstart = startTouch;
}
</script>
<body onload="init();">
<div id="target1"> Touch me ... </div>
<div id="target2" ontouchstart="startTouch(event)"> Touch me ... </div>
</body>
</html>
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.

Browser compatibility

Desktop
Mobile

Basic support

GlobalEventHandlers.ontouchend

GlobalEventHandlers.ontouchend
A global event handler for the touchend event.

A global event handler for the touchend event.

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

Note: This attribute has *not* been formally standardized. It is specified in the Touch Events – Level 2 Editor's Draft specification and not in Touch Events Recommendation. This attribute is not widely implemented.

Syntax

```
var endHandler = targetElement.ontouchend;
```

Return value

endHandler
The *touchend* event handler for element targetElement.

Example

This example shows two ways to use *ontouchend* to set an element's *touchend* event handler.

```
<html>
<script>

function endTouch(ev) {
 // Process the event
}

function init() {
 var el=document.getElementById("target1");
 el.ontouchend = endTouch;
}
</script>
<body onload="init();">
<div id="target1"> Touch me ... </div>
<div id="target2" ontouchend="endTouch(event)"> Touch me ... </div>
</body>
</html>
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.

Browser compatibility

Desktop
Mobile

Basic support

GlobalEventHandlers.ontouchmove

GlobalEventHandlers.ontouchmove
A global event handler for the touchmove event.

A global event handler for the touchmove event.

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

Note: This attribute has *not* been formally standardized. It is specified in the Touch Events – Level 2 Editor's Draft specification and not in Touch Events Recommendation. This attribute is not widely implemented.

Syntax

```
var moveHandler = someElement.ontouchmove;
```

Return value

moveHandler
The *touchmove* event handler for element someElement.

Example

This example shows two ways to use *ontouchmove* to set an element's *touchmove* event handler.

```
<html>
<script>

function moveTouch(ev) {
 // Process the event
}

function init() {
 var el=document.getElementById("target1");
 el.ontouchmove = moveTouch;
}

<body onload="init();">
<div id="target1"> Touch me ... </div>
<div id="target2" ontouchmove="moveTouch(event)"> Touch me ... </div>
</body>
</html>
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.

Browser compatibility

Desktop
Mobile

Basic support

GlobalEventHandlers.ontouchcancel

GlobalEventHandlers.ontouchcancel
A global event handler for the touchcancel event.

A global event handler for the touchcancel event.

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

Note: This attribute has *not* been formally standardized. It is specified in the Touch Events – Level 2 Editor's Draft specification and not in Touch Events Recommendation. This attribute is not widely implemented.

Syntax

```
var cancelHandler = someElement.ontouchcancel;
```

Return value

cancelHandler
The *touchcancel* event handler for element someElement.

Example

This example shows two ways to use *ontouchcancel* to set an element's *touchcancel* event handler.

```
<html>
<script>
function cancelTouch(ev) {
 // Process the event
}
function init() {
 var el=document.getElementById("target1");
 el.ontouchcancel = cancelTouch;
}
</script>
<body onload="init();">
<div id="target1"> Touch me ... </div>
<div id="target2" ontouchcancel="cancelTouch(event)"> Touch me ... </div>
</body>
</html>
```

Specifications

Specification	Status	Comment
Touch Events – Level 2	Editor's Draft	Non-stable version.

Browser compatibility

Desktop
Mobile

Basic support

Browser compatibility

==========

- Desktop
- Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	22.0	18.0 (18.0)	Not supported	Not supported	Not supported

Example

=======

This example tracks multiple touch points at a time, allowing the user to draw in a <canvas> with more than one finger at a time. It will only work on a browser that supports touch events.
Note: The text below uses the term "finger" when describing the contact with the surface, but it could, of course, also be a stylus or other contact method.

Create a canvas

```
<canvas id="canvas" width="600" height="600" style="border:solid black 1px;">
  Your browser does not support canvas element.
</canvas>
<br>
<button onclick="startup()">Initialize</button>
<br>
Log: <pre id="log" style="border: 1px solid #ccc;"></pre>
```

Setting up the event handlers

When the page loads, the startup() function shown below should be called by our <body> element's onload attribute (but in the example we use a button to trigger it, due to limitations of the MDN live example system).

```
function startup() {
  var el = document.getElementsByTagName("canvas")[0];
  el.addEventListener("touchstart", handleStart, false);
  el.addEventListener("touchend", handleEnd, false);
  el.addEventListener("touchcancel", handleCancel, false);
  el.addEventListener("touchmove", handleMove, false);
  log("initialized.");
}
```

This simply sets up all the event listeners for our <canvas> element so we can handle the touch events as they occur.

Tracking new touches

We'll keep track of the touches in-progress.

```
var ongoingTouches = [];
```

When a touchstart event occurs, indicating that a new touch on the surface has occurred, the handleStart() function below is called.

```
function handleStart(evt) {
  evt.preventDefault();
  log("touchstart.");
  var el = document.getElementsByTagName("canvas")[0];
  var ctx = el.getContext("2d");
  var touches = evt.changedTouches;

  for (var i = 0; i < touches.length; i++) {
    log("touchstart:" + i + "...");
    ongoingTouches.push(copyTouch(touches[i]));
    var color = colorForTouch(touches[i]);
    ctx.beginPath();
    ctx.arc(touches[i].pageX, touches[i].pageY, 4, 0, 2 * Math.PI, false);  // a
circle at the start
    ctx.fillStyle = color;
    ctx.fill();
    log("touchstart:" + i + ".");
  }
}
```

This calls event.preventDefault() to keep the browser from continuing to process the touch event (this also prevents a mouse event from also being delivered). Then we get the context and pull the list of changed touch points out of the event's TouchEvent.changedTouches property.

After that, we iterate over all the Touch objects in the list, pushing them onto an array of active touch points and drawing the start point for the draw as a small circle; we're using a 4-pixel wide line, so a 4 pixel radius circle will show up neatly.

Drawing as the touches move

Each time one or more fingers moves, a touchmove event is delivered, resulting in our handleMove() function being called. Its responsibility in this example is to update the cached touch information and to draw a line from the previous position to the current position of each touch.

```
function handleMove(evt) {
  evt.preventDefault();
  var el = document.getElementsByTagName("canvas")[0];
  var ctx = el.getContext("2d");
  var touches = evt.changedTouches;

  for (var i = 0; i < touches.length; i++) {
    var color = colorForTouch(touches[i]);
    var idx = ongoingTouchIndexById(touches[i].identifier);

    if (idx >= 0) {
      log("continuing touch "+idx);
      ctx.beginPath();
      log("ctx.moveTo(" + ongoingTouches[idx].pageX + ", " + ongoingTou-
ches[idx].pageY + ");");
      ctx.moveTo(ongoingTouches[idx].pageX, ongoingTouches[idx].pageY);
      log("ctx.lineTo(" + touches[i].pageX + ", " + touches[i].pageY + ");");
      ctx.lineTo(touches[i].pageX, touches[i].pageY);
      ctx.lineWidth = 4;
      ctx.strokeStyle = color;
      ctx.stroke();

      ongoingTouches.splice(idx, 1, copyTouch(touches[i]));  // swap in the new
touch record
      log(".");
    } else {
      log("can't figure out which touch to continue");
    }
  }
}
```

This iterates over the changed touches as well, but it looks in our cached touch information array for the previous information about each touch in order to determine the starting point for each touch's new line segment to be drawn. This is done by looking at each touch's Touch.identifier property. This property is a unique integer for each touch, and remains consistent for each event during the duration of each finger's contact with the surface.

This lets us get the coordinates of the previous position of each touch and use the appropriate context methods to draw a line segment joining the two positions together.

After drawing the line, we call Array.splice() to replace the previous information about the touch point with the current information in the ongoingTouches array.

Handling the end of a touch

When the user lifts a finger off the surface, a touchend event is sent. We handle both of these the same way: by calling the handleEnd() function below. Its job is to draw the last line segment for each touch that ended and remove the touch point from the ongoing touch list.

```
function handleEnd(evt) {
  evt.preventDefault();
  log("touchend");
  var el = document.getElementsByTagName("canvas")[0];
  var ctx = el.getContext("2d");
  var touches = evt.changedTouches;

  for (var i = 0; i < touches.length; i++) {
    var color = colorForTouch(touches[i]);
    var idx = ongoingTouchIndexById(touches[i].identifier);
```

```
    if (idx >= 0) {
      ctx.lineWidth = 4;
      ctx.fillStyle = color;
      ctx.beginPath();
      ctx.moveTo(ongoingTouches[idx].pageX, ongoingTouches[idx].pageY);
      ctx.lineTo(touches[i].pageX, touches[i].pageY);
      ctx.fillRect(touches[i].pageX - 4, touches[i].pageY - 4, 8, 8);  // and a
square at the end
      ongoingTouches.splice(idx, 1);  // remove it; we're done
    } else {
      log("can't figure out which touch to end");
    }
  }
}
```

This is very similar to the previous function; the only real differences are that we draw a small square to mark the end and that when we call Array.splice(), we simply remove the old entry from the ongoing touch list, without adding in the updated information. The result is that we stop tracking that touch point.

Handling canceled touches

If the user's finger wanders into browser UI, or the touch otherwise needs to be canceled, the touchcancel event is sent, and we call the handleCancel() function below.

```
function handleCancel(evt) {
  evt.preventDefault();
  log("touchcancel.");
  var touches = evt.changedTouches;

  for (var i = 0; i < touches.length; i++) {
    ongoingTouches.splice(i, 1);  // remove it; we're done
  }
}
```

Since the idea is to immediately abort the touch, we simply remove it from the ongoing touch list without drawing a final line segment.

Convenience functions

This example uses two convenience functions that should be looked at briefly to help make the rest of the code more clear.

Selecting a color for each touch

In order to make each touch's drawing look different, the colorForTouch() function is used to pick a color based on the touch's unique identifier. This identifier is an opaque number, but we can at least rely on it differing between the currently-active touches.

```
function colorForTouch(touch) {
  var r = touch.identifier % 16;
  var g = Math.floor(touch.identifier / 3) % 16;
  var b = Math.floor(touch.identifier / 7) % 16;
  r = r.toString(16); // make it a hex digit
  g = g.toString(16); // make it a hex digit
  b = b.toString(16); // make it a hex digit
  var color = "#" + r + g + b;
  log("color for touch with identifier " + touch.identifier + " = " + color);
  return color;
}
```

The result from this function is a string that can be used when calling <canvas> functions to set drawing colors. For example, for a Touch.identifier value of 10, the resulting string is "#aaa".

Copying a touch object

Some browsers (mobile Safari, for one) re-use touch objects between events, so it's best to copy the bits you care about, rather than referencing the entire object.

```
function copyTouch(touch) {
  return { identifier: touch.identifier, pageX: touch.pageX, pageY: touch.pageY
};
```

```
}
```

Finding an ongoing touch

The ongoingTouchIndexById() function below scans through the ongoingTouches array to find the touch matching the given identifier, then returns that touch's index into the array.

```
function ongoingTouchIndexById(idToFind) {
  for (var i = 0; i < ongoingTouches.length; i++) {
    var id = ongoingTouches[i].identifier;

    if (id == idToFind) {
      return i;
    }
  }
  return -1;    // not found
}
```

Showing what's going on

```
function log(msg) {
  var p = document.getElementById('log');
  p.innerHTML = msg + "\n" + p.innerHTML;
}
```