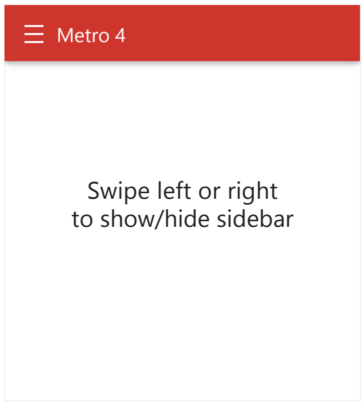
Touch and swipe

This component designed and to be used on touch devices such us iPad, iPhone, Android Phone, etc.

About

Based on **Matt Bryson** TouchSwipe plugin

The `touch` component designed and to be used on touch devices such us iPad, iPhone, Android Phone, etc. Component can detects single and multiple finger swipes, pinches and falls back to mouse 'drags' on the desktop. You can set time and distance thresholds to distinguish between swipe gesture and slow drag. The component allows exclusion of child elements (interactive elements) as well allowing page scrolling or page zooming depending on configuration.



To activate component, you must add attribute `data-role="touch"` to your touch and swipe area, or use function init method:

```
<body data-role="touch">
...
</body>
```

or

```
$("body").touch({...});
```

Main features

The main features of the `touch` component:

- Detects swipes in 4 directions, "up", "down", "left" and "right"
- Detects pinches "in" and "out"
- Supports single finger or double finger touch events
- Supports click events both on the touchSwipe object and its child objects
- Definable threshold / maxTimeThreshold to determining when a gesture is actually a swipe
- Events triggered for swipe "start","move","end" and "cancel"
- End event can be triggered either on touch release, or as soon as threshold is met
- Allows swiping and page scrolling
- Disables user input elements (Button, form, text etc) from triggering swipes

Options

You can set any options to change reaction component to your touches.

Option	Data-*	Default	Desc
`fingers`	`data-fingers`	1	Constants representing the number of fingers used in a swipe. Use integer value or 'ALL'. Any swipes that do not meet this requirement will NOT trigger swipe handlers.
`threshold`	`data-threshold`	75	The number of pixels that the user must move their finger by before it is considered a swipe.
`cancelThreshold`	`data-cancel-threshold`	null	The number of pixels that the user must move their finger back from the original swipe direction to cancel the gesture.
`pinchThreshold`	`data-pinch-threshold`	20	The number of pixels that the user must pinch their finger by before it is considered a pinch.
`pinchThreshold`	`data-pinch-threshold`	20	The number of pixels that the user must pinch their finger by before it is considered a pinch.
`maxTimeThreshold`	`data-max-time-threshold`	null	Time, in milliseconds, between touchStart and touchEnd must NOT exceed in order to be considered a swipe.
`fingerReleaseThreshold`	`data-finger-release-threshold`	250	Time in milliseconds between releasing multiple fingers. If 2 fingers are down, and are released one after the other, if they are within this threshold, it counts as a simultaneous release.
`longTapThreshold`	`data-long-tap-threshold`	500	Time in milliseconds between tap and release for a long tap.
`doubleTapThreshold`	`data-double-tap-threshold`	200	Time in milliseconds between 2 taps to count as a double tap.
			If true, the swipe events are triggered when the touch end event is

Option	Data-*	Default	Desc
`triggerOnTouchEnd`	`data-trigger-on-touch-end`	true	received (user releases finger). If false, it will be triggered on reaching the threshold, and then cancel the touch event automatically.
`triggerOnTouchLeave`	`data-trigger-on-touch-leave`	false	If true, then when the user leaves the swipe object, the swipe will end and trigger appropriate handlers.
`allowPageScroll`	`data-allow-page-scroll`	auto	How the browser handles page scrolls when the user is swiping on a swipe object. `auto` : all undefined swipes will cause the page to scroll in that direction. `none` : the page will not scroll when user swipes. `horizontal` : will force page to scroll on horizontal swipes. `vertical` : will force page to scroll on vertical swipes.
`fallbackToMouseEvents`	`data-fallback-to-mouse-events`	true	If true mouse events are used when run on a non touch device, false will stop swipes being triggered by mouse events on non touch devices.
`excludedElements`	`data-excluded-elements`	.no-swipe	A selector that specifies child elements that do NOT trigger swipes. By default this excludes elements with the class `.no-swipe` .
`preventDefaultEvents`	`data-prevent-default-events`	true	By default default events are cancelled, so the page doesn't move. You can disable this so both native events fire as well as your handlers.

Events

You can handle various events to respond to user actions: `swipe` , `swipe left` , `swipe right` , `swipe up` , `swipe down` , `swipe status` , `pinch in` , `pinch out` , `pinch status` , `tap` , `double tap` , `long tap` and `hold` .

Swipe events

Swipe events receive next arguments:

Argument	Type	Desc
`event`	`eventObject`	The original event object
`direction`	`string`	The direction the user action in: left, right, up, down, in, out
`distance`	`int`	The distance the user action.
`duration`	`int`	The duration of the action in milliseconds.
`fingerCount`	`int`	The number of fingers used.
`fingerData`	`object`	The coordinates of fingers in event.
`currentDirection`	`string`	The current direction the user is swiping.

Swipe

To handle `swipe` event, you must use attribute `data-on-swipe` or, if your use functionality init method, define method `onSwipe` in options.

```
<body data-role="touch" data-on-swipe="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipe: function(...){...}
});
```

Swipe left

To handle `swipe left` event, you must use attribute `data-on-swipe-left` or, if your use functionality init method, define method `onSwipeLeft` in options.

```
<body data-role="touch" data-on-swipe-left="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipeLeft: function(...){...}
});
```

Swipe right

To handle `swipe right` event, you must use attribute `data-on-swipe-right` or, if your use functionality init method, define method `onSwipeRight` in options.

```
<body data-role="touch" data-on-swipe-right="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipeRight: function(...){...}
});
```

Swipe up

To handle `swipe up` event, you must use attribute `data-on-swipe-up` or, if your use functionality init method, define method `onSwipeUp` in options.

```
<body data-role="touch" data-on-swipe-up="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipeUp: function(...){...}
});
```

Swipe down

To handle `swipe down` event, you must use attribute `data-on-swipe-down` or, if your use functionality init method, define method `onSwipeDown` in options.

```
<body data-role="touch" data-on-swipe-down="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipeDown: function(...){...}
});
```

Swipe status

You can observe `swipe` status with attribute `data-on-swipe-status` or method `onSwipeStatus`. If this function return false, swipe canceled.

```
<body data-role="touch" data-on-swipe-status="mySwipeFunction">
...
</body>

<script>
    function mySwipeFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onSwipeStatus: function(...){...}
});
```

Pinch events

Pinch events receive next arguments:

Argument	Type	Desc
`event`	`eventObject`	The original event object
`direction`	`string`	The direction the user action in: left, right, up, down, in, out
`distance`	`int`	The distance the user action.
`duration`	`int`	The duration of the action in milliseconds.
`fingerCount`	`int`	The number of fingers used.

Argument	Type	Desc
`fingerData`	`object`	The coordinates of fingers in event.
`zoom`	`int`	The zoom/scale level the user pinched too, 0-1.

Pinch in

To handle `pinch in` event, you must use attribute `data-on-pinch-in` or, if your use functionality init method, define method `onPinchIn` in options.

```
<body data-role="touch" data-on-pinch-in="myPinchFunction">
...
</body>

<script>
    function myPinchFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onPinchIn: function(...){...}
});
```

Pinch out

To handle `pinch out` event, you must use attribute `data-on-pinch-out` or, if your use functionality init method, define method `onPinchOut` in options.

```
<body data-role="touch" data-on-pinch-out="myPinchFunction">
...
</body>

<script>
    function myPinchFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onPinchOut: function(...){...}
});
```

Pinch status

You can observe `pinch` status with attribute `data-on-pinch-status` or method `onPinchStatus`.

```
<body data-role="touch" data-pinch-status="myPinchFunction">
...
</body>

<script>
    function myPinchFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onPinchStatus: function(...){...}
});
```

Tap events

Tap events receive next arguments:

Argument	Type	Desc
`event`	`eventObject`	The original event object
`target`	`DOMObject`	The element clicked on

Tap

To handle `tap` event, you must use attribute `data-on-tap` or, if your use functionality init method, define method `onTap` in options.

```
<body data-role="touch" data-tap="myTapFunction">
...
</body>

<script>
    function myTapFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onTap: function(...){...}
});
```

Double tap

To handle `double tap` event, you must use attribute `data-on-double-tap` or, if your use functionality init method, define method `onDoubleTap` in options.

```
<body data-role="touch" data-on-double-tap="myTapFunction">
...
</body>

<script>
    function myTapFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onDoubleTap: function(...){...}
});
```

Long tap

To handle `long tap` event, you must use attribute `data-on-long-tap` or, if your use functionality init method, define method `onLongTap` in options.

```
<body data-role="touch" data-long-tap="myTapFunction">
...
</body>

<script>
    function myTapFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onLongTap: function(...){...}
});
```

Hold

To handle `hold` event, you must use attribute `data-on-hold` or, if your use functionality init method, define method `onHold` in options.

```
<body data-role="touch" data-hold="myTapFunction">
...
</body>

<script>
    function myTapFunction(...){
        ...
    }
</script>
```

or

```
$("body").touch({
    onHold: function(...){...}
});
```