

# JavaScript position and sizes

property Description	use with			considers			
	<i>eventListener</i>	<i>window</i>	<i>element</i>	<i>padding</i>	<i>border</i>	<i>scrollBar</i>	<i>margin</i>
<b>.pageX / .pageY</b> Position of mouse relative to whole page	✓						
<b>.clientX / .clientY</b> Position of mouse relative to viewable window	✓						
<b>.style</b> Only works if previously set with JS			✓				
<b>.getBoundingClientRect()</b> Returns the exact height, width...		✓	✓				
<b>.innerHeight / .innerWidth</b> Returns height and width of window only		✓				✓	
<b>.clientHeight / .clientWidth</b> Returns height and width of window and elements		✓	✓	✓			
<b>.offsetHeight / .offsetWidth</b> Returns height and width of window and elements		✓	✓	✓	✓	✓	
<b>.offsetTop / .offsetLeft</b> Returns position of elements			✓				

## Contents

Mouse movement event listener.....	3
.pageX / .pageY .....	4
.clientX / .clientY .....	4
Comparison of .clientY and .pageY .....	4
Height and width of elements .....	5
.style.height / .style.width (!warning).....	5
window.innerHeight / window.innerWidth.....	5
.clientHeight / .clientWidth.....	5
.offsetHeight / .offsetWidth.....	5
.offsetLeft / .offsetTop.....	5

## Mouse movement event listener

```
document.addEventListener("mousemove", mouseMove);

function mouseMove(e){
  console.log(e)
}
```

By default, an eventListener sends an argument (in this case we name it **e**) to the function you wish to call. Console logging this argument (e) we can see that we get an object with many properties from the "mousemove" eventListener:

Some of them are very useful for us, such as:

```
MouseEvent {isTrusted: true, screenX: 703, screenY: 552, clientX:
  330, clientY: 453, ...} ⓘ
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelBubble: false
  cancelable: true
  clientX: 330
  clientY: 453
  composed: true
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 0
  eventPhase: 0
  fromElement: null
  isTrusted: true
  layerX: 330
  layerY: 453
  metaKey: false
  movementX: 64
  movementY: -56
  offsetX: 330
  offsetY: 453
  pageX: 330
  pageY: 453
  path: (3) [html.gr__127_0_0_1, document, Window]
  relatedTarget: null
  returnValue: true
  screenX: 703
  screenY: 552
  shiftKey: false
  sourceCapabilities: InputDeviceCapabilities {firesTouchEvents: fa
  srcElement: html.gr__127_0_0_1
  target: html.gr__127_0_0_1
  timeStamp: 323063.8999999501
  toElement: html.gr__127_0_0_1
  type: "mousemove"
  view: Window {postMessage: f, blur: f, focus: f, close: f, frames
    which: 0
    x: 330
    y: 453
```

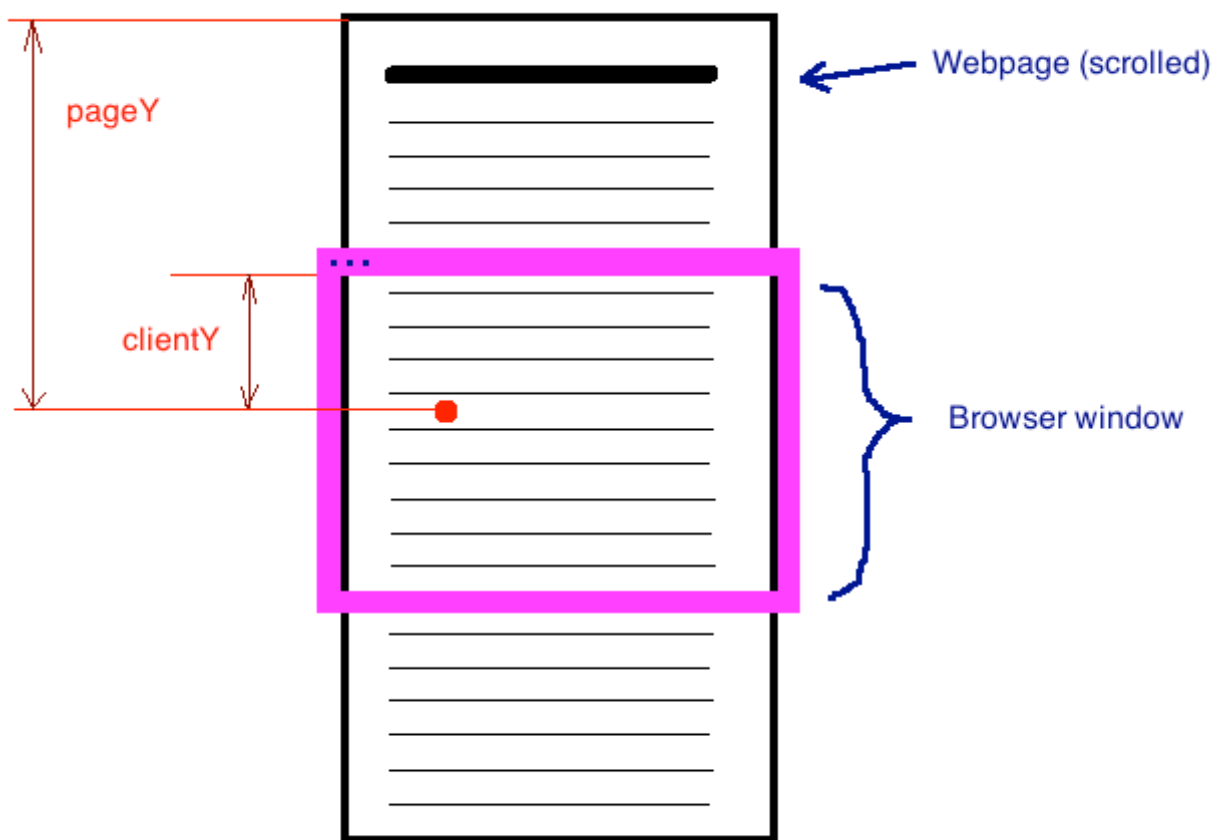
`.pageX / .pageY`

These two methods return the current position of the mouse relatively to the top and left of the webpage.

`.clientX / .clientY`

These two methods return the current position of the mouse relatively to the top and left of the screen, ignoring if the page has been scrolled to the bottom or not.

Comparison of `.clientY` and `.pageY`



## Height and width of elements

Not depending on the (e) of an eventListener. **window** refers to the whole page where as **element** only refers to one object.

### [.style.height / .style.width \(!warning\)](#)

Returns only the height or width of element without padding, border, scrollbar and margin. This only works if the height or width **has been set before in JavaScript**, not in CSS! (So not recommended at all)

In order to access it the height and width, we have to use `box.getBoundingClientRect()` instead of `.style`

E.g.

style.css

```
div{
  height: 10px;
  width: 20px;
}
```

script.js

```
const box = document.querySelector("div");
console.log(box.style.height); // returns nothing
console.log(box.getBoundingClientRect().height); //returns 10px
```

### [window.innerHeight / window.innerWidth](#)

Returns the viewport height or width **of the window only** (not elements) in pixels. The value contains the height or width with the scrollbar.

### [.clientHeight / .clientWidth](#)

Returns the height or width of the visible area of an element, in pixels. The value contains the height or width with the padding, but it does not include the scrollbar, border, and the margin. Both properties only return a value if the height or width has been set by CSS or JavaScript before. If this is not the case the returned value is zero. Use `getBoundingClientRect()` then as the alternative.

### [.offsetHeight / .offsetWidth](#)

Returns the height or width of the visible area of an element, in pixels. The value contains the height or width with the padding, scrollbar, and the border, but does not include the margin.

### [.offsetLeft / .offsetTop](#)

Returns the current position relative to the top, left border.

