CAPTURING EVENTS WITH JAVASCRIPT

HTML events are **"things"** that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can **"react"** on these events. An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

Example)

```
<button>PRESS ME!</button>
  <script type="text/javascript">
    var btn = document.querySelector('button');
     btn.onclick = function(){ alert("Hi There!");} // good for only one run
</script>
```

```
//anonymous function: use for multiple runs
 function touch(){
    alert("Hi There!");
  }
  btn.onclick = touch;
```

Event listeners

addEventListener()

The `addEventListener()` method attaches an event handler to the specified element. `addEventListener()` can capture many different events.

Example)

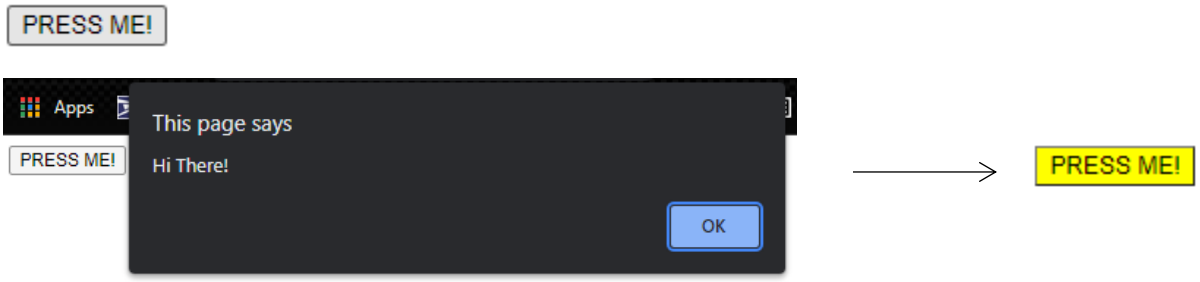
```
button>PRESS ME!</button>
  <script type="text/javascript">
    var btn = document.querySelector('button');
      btn.addEventListener('click', function(){
        alert("Hi There!");}
   );
```

The event object

There is an event object that can be passed into event handlers, and sometimes you need to access it specifically.

Example)

```
<button>PRESS ME!</button>
  <script type="text/javascript">
    var btn = document.querySelector('button');
    btn.addEventListener('click', function(event){
      event.target.style.backgroundColor = "yellow";
      alert('Hi There!');
    });
</script>
```



preventDefault()

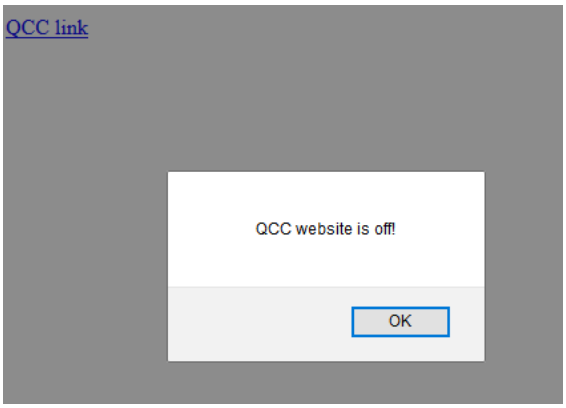
The `preventDefault()` method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

For example, this can be useful when:

- Clicking on a "Submit" button, prevent it from submitting a form
- Clicking on a link, prevent the link from following the URL

Example)

```
<a href="www.qcc.cuny.edu">QCC link</a>
  <script type="text/javascript">
    var link = document.querySelector('a');
    link.addEventListener('click',function(e){
      e.preventDefault();
      alert('QCC website is off!');
    });
</script>
```



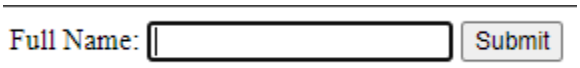
If we remove `e.preventDefault();` line from the code, when we click on the QCC link, the alert message will appear and the load www.qcc.cuny.edu website.

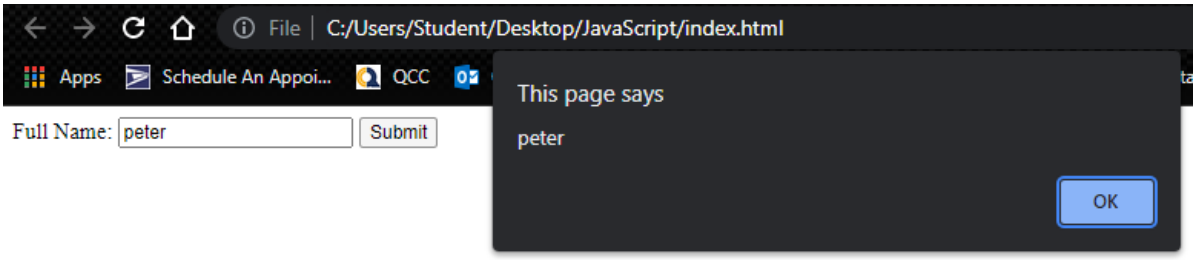
`addEventListener()`

The `addEventListener()` method is used to attach an event handler to a particular element. It does not override the existing event handlers. We can add multiple event handlers to a particular element without overwriting the existing event handlers.

Example)

```
<form method="get">
  <label>Full Name: <input type="text" name="name"> </label>
  <input type="submit">
</form>

  <script type="text/javascript">
    var myForm = document.querySelector('form');

    myForm.addEventListener('submit', function(e){
      e.preventDefault();

      var formData = document.querySelector('input').value;

      alert(formData);
    });
```





If we take the line `e.preventDefault();` out the code, the input field will clear after clicking the OK button. We use `e.preventDefault();` because most of the time we want the information of the form to pass to somewhere.

Mouseover, mouseout events

The `mouseover` event takes place when the pointer of the mouse comes over an element. On the contrary, the `mouseout` event occurs when it leaves.

Example)

```
h1>Mouseover</h1>
<div class="square"></div> <!-- Square -->

  <script type="text/javascript">
  var heading = document.querySelector('h1');
  var box = document.querySelector('div');
  box.addEventListener('mouseover',function(){
    heading.innerHTML='The mouse is OVER the box';
  });
  box.addEventListener('mouseout', function(){
      heading.innerHTML='The mouse is OUT the box';
  });


box.addEventListener('mouseover',function(e){
    heading.innerHTML='The mouse is OVER the box';
    e.target.style.backgroundColor = "yellow";
    e.target.style.width = "100px";
    e.target.style.height = "100px";
  });

  box.addEventListener('mouseout', function(e){
      heading.innerHTML='The mouse is OUT the box';
      //e.target.style.backgroundColor = "lightgreen";
      e.target.style = ".square";
  });
```

Scroll events

When you scroll a document or an element, the scroll events fire. `scroll` events on the window object to handle the scroll of the whole webpage.

To register a scroll event handler, you call the `addEventListener()` method on the target element, like this:

```
targetElement.addEventListener('scroll', (event) => {
    // handle the scroll event
});
```

or assign an event handler to the `onscroll` property of the target element:

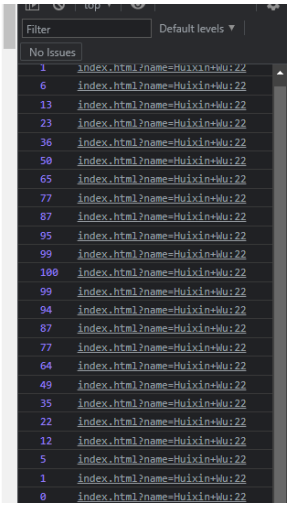
```
targetElement.onscroll = (event) => {
    // handle the scroll event
};
```

Example)

```
<div class="square"></div> <!-- Square -->
  <script type="text/javascript">
  var pageTop;
window.addEventListener('scroll', function(){
    pageTop = window.pageYOffset;
    console.log(pageTop);
  });
```

When we scroll, it will show the pixes height of the box.

Capturing scroll event

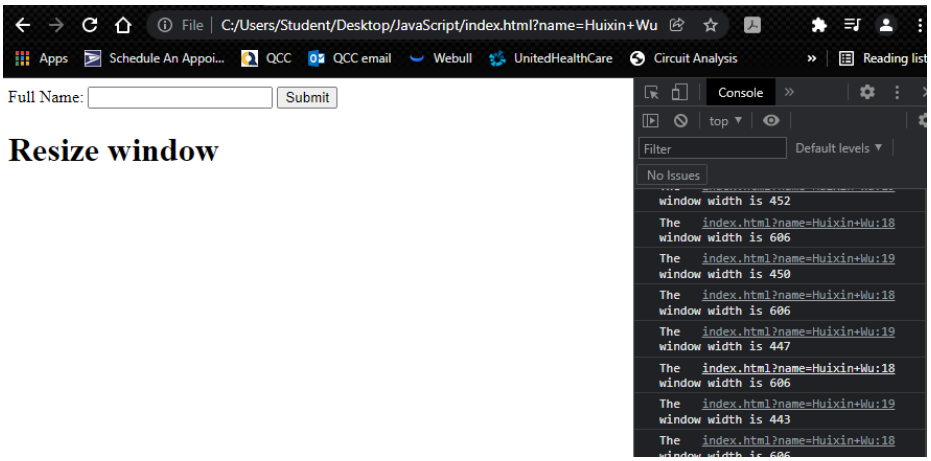


Window resizing

The **`resize`** event fires when the document view (window) has been resized. In some earlier browsers it was possible to register resize event handlers on any HTML element. It is still possible to set **`onresize`** attributes or use `addEventListener()` to set a handler on any element. However, resize events are only fired on the window object (i.e. returned by `document.defaultView`). Only handlers registered on the window object will receive resize events.

Example)

```
<h1>Resize window</h1>
  <script type="text/javascript">
  window.addEventListener('resize',function(){
    console.log(`The window width is ${window.innerWidth}`);
    console.log(`The window width is ${window.innerHeight}`);
  });
```



Key down event

The `keydown` event occurs when the user is pressing a key (on the keyboard).

Example)

```
<h1>Press a key on the keyboard</h1>
  <script type="text/javascript">
  document.addEventListener('keydown', function(){
    alert('A key is pressed!');
  });
```

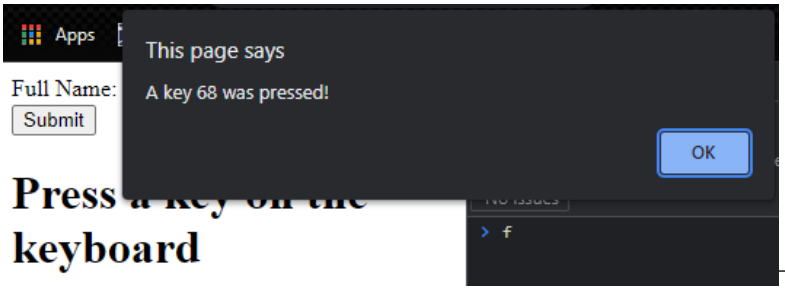
We can also check which key was pressed by given an argument to the anonymous function:

- *which* method shows the ASCII code of the selected key
- *key* method shows the key character that you have pressed

Example)

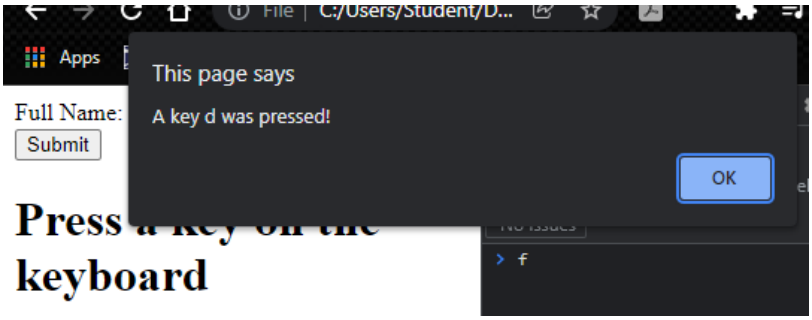
```
<script type="text/javascript">
  document.addEventListener('keydown', function(e){
    alert(`A key ${e.which} was pressed!`);
  });
```

By pressing the letter “d”



Example)

```
<script type="text/javascript">
  document.addEventListener('keydown', function(e){
    alert(`A key ${e.key} was pressed!`);
  });
```



READING

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events

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