**event.preventDefault()**

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

For example, this can be useful when:

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

<a id="myAnchor" href="https://w3schools.com/">Go to W3Schools.com</a>

<script>

document.getElementById("myAnchor").addEventListener("click", function(event){

event.preventDefault()

});

</script>

**Note:** The preventDefault() method will prevent the link above from following the URL.

**event.stopPropagation()**

The **method stops the bubbling of an event to parent elements**, preventing any parent event handlers from being executed.

<script>

$(document).ready(function(){

$("span").click(function(event){

event.stopPropagation();

alert("The span element was clicked.");

});

$("p").click(function(event){

alert("The p element was clicked.");

});

$("div").click(function(){

alert("The div element was clicked.");

});

});

</script>

<div style="height: 100px; width: 500px; padding: 10px; border: 1px solid blue;">

This is a div element.

<p style="background-color: pink">This is a p element, in the div element.

<span style="background-color:orange">Span element in p and div element.

</span>

</p>

</div>

**event.stopImmediatePropagation()**

* This method stops the rest of the event handlers from being executed.
* This method also stops the event from bubbling up the DOM tree.

<script>

$(document).ready(function(){

$("div").click(function(event){

alert("Event handler 1 executed");

event.stopImmediatePropagation();

});

$("div").click(function(event){

alert("Event handler 2 executed");

});

$("div").click(function(event){

alert("Event handler 3 executed");

});

});

</script>

<div style="height: 100px; width: 300px;">Click on this div element.</div>

**Note**: The second and third click event will not be executed due to event.stopImmediatePropagation().

var Paramerter = {

id: $(obj).val()

}

$.ajax({

url: ‘’,

cache: false,

async: false,

data: Paramerter,

success: function (data) {

},

error: function () {

}

});

|  |  |
| --- | --- |
| **Name** | **Value/Description** |
| async | A Boolean value indicating whether the request should be handled asynchronous or not. **Default is true** |
| beforeSend(*xhr*) | A function to run before the request is sent |
| cache | A Boolean value indicating whether the browser should cache the requested pages. **Default is true** |
| complete(*xhr,status*) | A function to run when the request is finished **(execute after success or error functions)** |
| contentType | The content type **used when sending data to the server**. **Default is: "application/x-www-form-urlencoded"** |
| context | Specifies the "this" value for all AJAX related callback functions |
| data | Specifies **data to be sent to the server** |
| dataFilter(*data*,*type*) | A function used to handle the raw response data of the XMLHttpRequest |
| dataType | The data type **expected of the server response**. |
| error(*xhr,status,error*) | A function to **run if the request fails.** |
| global | A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true |
| ifModified | A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false. |
| jsonp | A string overriding the callback function in a jsonp request |
| jsonpCallback | Specifies a name for the callback function in a jsonp request |
| password | Specifies a password to be used in an HTTP access authentication request. |
| processData | A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true |
| scriptCharset | Specifies the charset for the request |
| success(*result,status,xhr*) | A function to be run when the request succeeds |
| timeout | The local timeout (in milliseconds) for the request |
| traditional | A Boolean value specifying whether or not to use the traditional style of param serialization |
| type | Specifies the **type of request. (GET or POST)** |
| url | Specifies the URL to send the request to. **Default is the current page** |
| username | Specifies a username to be used in an HTTP access authentication request |
| xhr | A function used for creating the XMLHttpRequest object |

**jQuery Callback Functions**

* JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
* To prevent this, you can create a callback function.
* A callback function is executed after the current effect is finished.

Typical syntax: $(selector).hide(speed,callback);

Examples

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

<script>

$(document).ready(function(){

$("button").click(function(){  
  $("p").hide("slow", function(){ //after "slow", is the callback function  
   alert("The paragraph is now hidden");  
  });  
});

});

</script>