1/Event: preventDefault() method

The **preventDefault()** method of the Event interface tells the user agent that if the event does not get explicitly handled, its default action should not be taken as it normally would be.

The event continues to propagate as usual, unless one of its event listeners calls stopPropagation() or stopImmediatePropagation(), either of which terminates propagation at once.

As noted below, calling **preventDefault()** for a non-cancelable event, such as one dispatched via EventTarget.dispatchEvent(), without specifying cancelable: true has no effect.

Syntax

JSCopy to Clipboard

```
event.preventDefault()
```

Examples

Blocking default click handling

Toggling a checkbox is the default action of clicking on a checkbox. This example demonstrates how to prevent that from happening:

JavaScript

```
const checkbox = document.querySelector("#id-checkbox");



checkbox.addEventListener("click", checkboxClick, false);



function checkboxClick(event) {
  let warn = "preventDefault() won't let you check this!<br>";
  document.getElementById("output-box").innerHTML += warn;
  event.preventDefault();
}
```

HTML

HTMLPlayCopy to Clipboard

```
<p>Please click on the checkbox control.</p>

<form>
  <label for="id-checkbox">Checkbox:</label>
  <input type="checkbox" id="id-checkbox" />
</form>

<div id="output-box"></div>
```

2/Event: stopPropagation() method

The **stopPropagation()** method of the Event interface prevents further propagation of the current event in the capturing and bubbling phases. It does not, however, prevent any default behaviors from occurring; for instance, clicks on links are still processed. If you want to stop those behaviors, see the preventDefault() method. It also does not prevent propagation to other event-handlers of the current element. If you want to stop those, see stopImmediatePropagation().

Syntax

JSCopy to Clipboard

```
event.stopPropagation()
```

Parameters

None.

Return value

None.

Examples of web and XML development using the DOM

This chapter provides some longer examples of web and XML development using the DOM. Wherever possible, the examples use common APIs, tricks, and patterns in JavaScript for manipulating the document object.

Example 1: height and width

The following example shows the use of the height and width properties alongside images of varying dimensions:

HTMLCopy to Clipboard

```
<!doctype html>
<html lang="en">
  <head>
    <title>width/height example</title>
    <script>
      function init() {
        const arrImages = new Array(3);

        arrImages[0] = document.getElementById("image1");
        arrImages[1] = document.getElementById("image2");
        arrImages[2] = document.getElementById("image3");

        const objOutput = document.getElementById("output");
        let strHtml = "<ul>";

        for (let i = 0; i < arrImages.length; i++) {
          strHtml +=
            "<li>image" +
            (i + 1) +
            ": height=" +
            arrImages[i].height +
            ", width=" +
            arrImages[i].width +
            ", style.height=" +
            arrImages[i].style.height +
            ", style.width=" +
            arrImages[i].style.width +
            "<\/li>";
        }

        strHtml += "<\/ul>";

        objOutput.innerHTML = strHtml;
      }
    </script>
  </head>
  <body onload="init();">
    <p>
      Image 1: no height, width, or style
      <img id="image1" src="http://www.mozilla.org/images/mozilla-banner.gif" />
    </p>

    <p>
      Image 2: height="50", width="500", but no style
      <img
        id="image2"
        src="http://www.mozilla.org/images/mozilla-banner.gif"
        height="50"
```

```
      width="500" />
  </p>

  <p>
   Image 3: no height, width, but style="height: 50px; width: 500px;"
   <img
    id="image3"
    src="http://www.mozilla.org/images/mozilla-banner.gif"
    style="height: 50px; width: 500px;" />
  </p>

  <div id="output"></div>
 </body>
</html>
```

3/Event: stopImmediatePropagation() method

The **stopImmediatePropagation()** method of the Event interface prevents other listeners of the same event from being called.

If several listeners are attached to the same element for the same event type, they are called in the order in which they were added. If stopImmediatePropagation() is invoked during one such call, no remaining listeners will be called, either on that element or any other element.

Syntax

JSCopy to Clipboard

```
event.stopImmediatePropagation()
```

Examples

Comparing event-stopping functions

The example below has three buttons inside of three nested divs. Each button has three event listeners registered for click events, and each div has an event listener, also registered for click events.

- The top button allows normal event propagation.
- The middle button calls stopPropagation() in its first event handler.

- The bottom button calls stopImmediatePropagation() in its first event handler.

HTML

HTMLPlayCopy to Clipboard

```
<h2>Click on the buttons</h2>
<div>
  outer div<br />
  <div>
    middle div<br />
    <div>
      inner div<br />
      <button>allow propagation</button><br />
      <button id="stopPropagation">stop propagation</button><br />
      <button id="stopImmediatePropagation">immediate stop propagation</button>
    </div>
  </div>
</div>
<pre></pre>
```

CSS

CSSPlayCopy to Clipboard

```
div {
  display: inline-block;
  padding: 10px;
  background-color: #fff;
  border: 2px solid #000;
  margin: 10px;
}

button {
  width: 100px;
  color: #008;
  padding: 5px;
  background-color: #fff;
  border: 2px solid #000;
  border-radius: 30px;
  margin: 5px;
}
```

JavaScript

JSPlayCopy to Clipboard

```
const outElem = document.querySelector("pre");

/* Clear the output */
document.addEventListener(
  "click",
  () => {
    outElem.textContent = "";
  },
  true,
);
```

```
/* Set event listeners for the buttons */
document.querySelectorAll("button").forEach((elem) => {
  for (let i = 1; i <= 3; i++) {
    elem.addEventListener("click", (evt) => {
      /* Do any propagation stopping in first event handler */
      if (i === 1 && elem.id) {
        evt[elem.id]();
        outElem.textContent += `Event handler for event 1 calling ${elem.id}()\n`;
      }

      outElem.textContent += `Click event ${i} processed on "${elem.textContent}" button\n`;
    });
  }
});

/* Set event listeners for the divs */
document
  .querySelectorAll("div")
  .forEach((elem) =>
    elem.addEventListener(
      "click",
      (evt) =>
        (outElem.textContent += `Click event processed on "${elem.firstChild.data.trim()}"\n`),
    ),
  );
```

Result

Each click-event handler displays a status message when it is called. If you press the middle button, you will see that stopPropagation() allows all of the event handlers registered for clicks on that button to execute but prevents execution of the click-event handlers for the divs, which would normally follow. However, if you press the bottom button, stopImmediatePropagation() stops all propagation after the event that called it.

Click on the buttons