The `Event` interface's **`preventDefault()`** method 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

```
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

```
document.querySelector("#id-checkbox").addEventListener("click", functi
        document.getElementById("output-box").innerHTML += "Sorry! <co
        event.preventDefault();
}, false);
```

HTML

```
<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>
```

Result

Stopping keystrokes from reaching an edit field

The following example demonstrates how invalid text input can be stopped from reaching the input field with `preventDefault()`. Nowadays, you should usually use native HTML form validationinstead.

HTML

Here's the form:

```
<div class="container">
  <p>Please enter your name using lowercase letters only.</p>

  <form>
    <input type="text" id="my-textbox">
  </form>
</div>
```

CSS

We use a little bit of CSS for the warning box we'll draw when the user presses an invalid key:

```
.warning {
  border: 2px solid #f39389;
  border-radius: 2px;
  padding: 10px;
```

```
  position: absolute;
  background-color: #fbd8d4;
  color: #3b3c40;
}
```

JavaScript

And here's the JavaScript code that does the job. First, listen for `keypress` events:

```
var myTextbox = document.getElementById('my-textbox');
myTextbox.addEventListener('keypress', checkName, false);
```

The `checkName()` function, which looks at the pressed key and decides whether to allow it:

```
function checkName(evt) {
  var charCode = evt.charCode;
  if (charCode != 0) {
    if (charCode < 97 || charCode > 122) {
      evt.preventDefault();
      displayWarning(
        "Please use lowercase letters only."
        + "\n" + "charCode: " + charCode + "\n"
      );
    }
  }
}
```

The `displayWarning()` function presents a notification of a problem. It's not an elegant function but does the job for the purposes of this example:

```
var warningTimeout;
var warningBox = document.createElement("div");
warningBox.className = "warning";

function displayWarning(msg) {
  warningBox.innerHTML = msg;

  if (document.body.contains(warningBox)) {
```

```
        window.clearTimeout(warningTimeout);
      } else {
        // insert warningBox after myTextbox
        myTextbox.parentNode.insertBefore(warningBox, myTextbox.nextSibling
      }

      warningTimeout = window.setTimeout(function() {
          warningBox.parentNode.removeChild(warningBox);
          warningTimeout = -1;
        }, 2000);
    }
```

Result