EVENT-LISTENER

Die addEventListener() Methode setzt einen „Zuhörer“ (engl. listener) auf ein Zielelement, bei welchem es aufgerufen wird. Das Zielelement ist in den meisten Fällen ein Element im Document, das Document selbst oder ein Window-Objekt.

Syntax:

zielelement.addEventListener(*event, function, option*)

event ist ein String und kann verschiedene Werte haben (click, change, mouseover, mouseenter, mouseout…)

function ist die Funktion (anonym oder benamt/benannt), welche ausgeführt wird.
WICHTIG: Soll der EventListener später entfernt werden, MUSS es eine benannte Funktion sein. Hier wird nur der Funktionsname ohne ausführende Klammern () geschrieben.

Option sind optionale Werte (siehe weiter unten bei Entfernen von EventListeners und Bubbling)

Unterschied zwischen .onclick() und .addEventListener()

Werden mehrere .addEventListener() mit demelben Event auf ein und dasselbe Objekt gelegt, so werden diese nacheinander ausgeführt.

```
insideDivObj.addEventListener(`click`, () => {
   insideDivObj.innerHTML = "Wurde geclickt";
});

insideDivObj.addEventListener(`click`, () => {
   insideDivObj.style.background = "yellow";
});
```

Hier werden zwei Click-Events auf das insideDivObj gelegt und beide werden ausgeführt.

Bei der .onclick() Methode funktioniert das nicht!

```
insideDivObj.onclick=()=>{alert('Hallo')};
insideDivObj.onclick=()=>{alert('Good-Bye')}
```

Hier wird mit der zweiten Zuweisung der .onclick() Methode die erste überschrieben, d.h. es wird nur „Good-Bye“ alertet!

.removeEventListener() Methode

Sollen eventListener später wieder entfernt werden, so muss bei dem Setzen der addEventListener() Methode die auszuführende Funktion eine benannte Funktion sein und zusätzlich ein Parameter true oder false übergeben werden.

Dann kann die .removeEventListener() Methode angewandt werden.

```
const redBackground = () => {
   outsideDivObj.style.background= "red";
}
insideDivObj.addEventListener(`mouseenter`, redBackground, true);

insideDivObj.removeEventListener(`mouseenter`, redBackground, true);
```

Übergabe von Parametern

Möchte man Parameter beim Event an die auszuführende Funktion übergeben, so nutzt man in der .addEventListener() Methode eine anonyme Funktion, welche die gewünschte Funktion mit Parametern aufruft.

Syntax:

```
zielelement.addEventListener(`change`, () =>
                                {functionToDo(parameter1, parameter2)})
```

Event Bubbling vs Event Capturing

Ist ein Click-Event auf ein HTML-Element gesetzt und in diesem HTML-Element ist ein weiteres HTML-Element mit Click-Event, welches wird zuerst ausgeführt?

Dies kann mit der *option* useCapture entschieden werden, dies kann true oder false sein.

Syntax:

zielelement.addEventListener(`click`, function, *true* oder *false*).

Dies bedeutet:

BUBBLING = Das innere Event wird vor dem äußeren ausgeführt, *useCapture* ist in diesem Fall *false.*

By default (also Grundeinstellung) ist *useCapture = false*. Somit ist Bubbling ohne Angabe von *useCapture* das normale Verhalten.

CAPTURING = Das äußere Event wird vor dem inneren ausgeführt, *useCapture* ist in diesem Fall *true*.

Hier der komplette Code zum Ausprobieren von EventListener:

HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>tests/eventlister</title>
   <style>
      #outside {
         width: 500px;
         height: 500px;
         background: darkslategray;
         padding-top: 195px;
         box-sizing: border-box;
      }
      #inside {
         width: 100px;
         height: 100px;
         border: 10px solid black;
         background: white;
         margin: 0 auto;
         cursor: pointer;
         text-align: center;
         line-height: 100px;
      }
      #outsideLeft {
         width: 300px;
         height: 200px;
         line-height: 200px;
         background: green;
         box-sizing: border-box;
         padding-top: 50px;
      }
      #insideLeft {
         width: 100px;
         height: 100px;
         background: yellow;
         margin: 0 auto;
      }
      #outsideRight {
         width: 300px;
```

```
            height: 200px;
            line-height: 200px;
            background: red;
            box-sizing: border-box;
            padding-top: 50px;
        }
        #insideRight {
            width: 100px;
            height: 100px;
            background: orange;
            margin: 0 auto;
        }
        .bubbling_capturing {
            display: flex;
            justify-content: space-between;
            margin-top: 10px;
        }
        .text {
            font-size: 25px;
            line-height: 50px;
            text-align: center;
        }
    </style>

</head>

<body>
    <div id="outside">
        <div id="inside">Hier klicken</div>
    </div>
    <div class="bubbling_capturing">
        <div id="outsideLeft">
            <div id="insideLeft"></div>
            <div class="text">Capturing</div>
        </div>
        <div id="outsideRight">
            <div id="insideRight"></div>
            <div class="text">Bubbling</div>
        </div>
    </div>
    <script src="main.js"></script>
</body>

</html>
```

JAVASCRIPT

```
const insideDivObj = document.getElementById('inside');
const outsideDivObj = document.getElementById('outside');
const insideLeftDivObj = document.getElementById(`insideLeft`);
const insideRightDivObj = document.getElementById(`insideRight`);
const outsideLeftDivObj = document.getElementById(`outsideLeft`);
const outsideRightDivObj = document.getElementById(`outsideRight`);

const redBackground = () => {
    outsideDivObj.style.background= "red";
}

insideDivObj.addEventListener(`click`, () => {
    insideDivObj.innerHTML = "Wurde geclickt";
});

insideDivObj.addEventListener(`click`, () => {
    insideDivObj.style.background = "yellow";
});

insideDivObj.addEventListener(`mouseenter`, redBackground, true);

insideDivObj.addEventListener(`mouseout`, () => {
    insideDivObj.style.background = "green";
    outsideDivObj.style.background = `lightblue`;
    insideDivObj.removeEventListener(`mouseenter`, redBackground, true);
});


insideDivObj.onclick=()=>{alert('Hallo')};
insideDivObj.onclick=()=>{alert('Good-Bye')}; // diese Definition überschreit
// ddas onclick Event in der Zeile darüber

insideLeftDivObj.addEventListener(`click`, ()=> {
    alert(`CAPTURING: Hallo vom gelben Div`)}, true
);

outsideLeftDivObj.addEventListener(`click`, ()=> {
    alert(`CAPTURING: Hallo vom grünen Div`)}, true
);

insideRightDivObj.addEventListener(`click`, ()=> {
    alert(`BUBBLING: Hallo vom orangen Div`)}, false
);

outsideRightDivObj.addEventListener(`click`, ()=> {
    alert(`BUBBLING: Hallo vom roten Div`)}, false
);
```

Questions? You‘re welcome!

holger.zerbe@digitalcareerinstitute.org