JavaScript Eventhandler

Die einfachste Art des Eventhandling

- JavaScript kann als Eventhandler für ein HTML-Element direkt im HTML annotiert werden über die Attribute
  - onclick
  - onload
  - onmouseover
  - ...

```
<body>
<button onclick="onClickHandler()">Click</button>
<script>
function onClickHandler() {
    console.log('click');
}
</script>
</body>
```

Empfehlung für unaufdringliches JavaScript

- Christian Heilmann entwickelte den Begriff Unobtrusive JavaScript, also unaufdringliches oder auch barrierefreies JavaScript.
- Die Interaktivität soll statt dessen dem Dokument automatisch hinzugefügt werden.
- Im HTML-Code sollte sich also kein JavaScript in Form von Eventhandler-Attributen befinden, sondern beim Laden des Dokuments sollten die Scripte automatisch starten und die Ereignisüberwachung an den betreffenden Elementen übernehmen.

Empfehlung für unaufdringliches JavaScript

- Im Idealfall sollte im `head`-Element das ein oder andere `script`-Element stehen, um eine externe JavaScript-Datei einzubinden, die dann ihrerseits aktiv wird und die Eventhandler an die jeweiligen Elemente anbringt.
- Dazu können Elemente, denen ein bestimmtes Verhalten hinzugefügt werden soll, z.B. mit einer Klasse markiert oder bei Bedarf mit einer ID ausgezeichnet werden.
- Mischen Sie also kein JavaScript mit HTML.
  - Hängen Sie statt dessen Eventhandler dynamisch an HTML-Elemente an.
- Trennen Sie auch CSS-Anweisungen von JavaScript-Code.
  - Definieren Sie die Formatierungsregeln in einem zentralen Stylesheet, nicht im JavaScript.
  - Sorgen Sie im JavaScript dafür, dass diese Formatierungsregeln angewendet werden, indem Sie einem Element dynamisch eine Klasse hinzufügen.

https://wiki.selfhtml.org/wiki/HTML/Tutorials/Trennung_von_Inhalt,_Pr%C3%A4sentation_und_Verhalten#unaufdringliches_JavaScript

Hinzufügen und Entfernen von Eventhandlern

- Man kann einen Event Listener mittels **`element.addEventListener`** hinzufügen.
- Falls der Event Handler nicht mehr ausgeführt werden soll, muss er mittels **`element.removeEventListener`** wieder entfernt werden.
- Auf diese Weise können beliebig viele Handler für ein Event auf einem Element erstellt werden.
- Mann kann ihn auch direkt im HTML oder durch **`element.on<event-name>`** hinzufügen.
  - Dabei kann jedoch nur ein Handler benutzt werden.

Ein dynamischer Handler

```
main {
    padding: 0.5em 1em;
    max-width: 40em;
    background-color: hsl(210, 50%, 80%);
    outline: 1px solid black;
}
```

https://wiki.selfhtml.org/wiki/JavaScript/DOM/EventTarget/addEventListener

Ein dynamischer Handler

```
<body>
<h1>Beispiel: Registrierung eines Event-Handlers</h1>
<main>

<p>Mit dem blau eingefärbten Main-Element ist bislang
keine Funktion verknüpft, das heißt, bei einem Klick
auf die blaue Fläche passiert nichts. Wird jedoch auf
den Button unten geklickt, dann wird für das Main-Element
und das Ereignis <em>click</em> ein Event-Handler registriert.
Klicken Sie also auf den Button und danach nocheinmal auf die
blaue Fläche!</p>

<button type="button" lang="en">addEventListener</button>
<p id="message" hidden>Event-Handler für das Element Main hinzugefügt!</p>

</main>
</body>
```

https://wiki.selfhtml.org/wiki/JavaScript/DOM/EventTarget/addEventListener

Ein dynamischer Handler

```
window.addEventListener('DOMContentLoaded', function () {
    var button = document.querySelector('button');
    button.addEventListener('click', handlerForButton);

    function handlerForButton() {
        var main = document.querySelector('main');
        main.addEventListener('click', handlerForMain, true);
    }

    function handlerForMain() {
        var paragraph = document.getElementById('message');
        paragraph.hidden = false;
    }
});
```

https://wiki.selfhtml.org/wiki/JavaScript/DOM/EventTarget/addEventListener

Capture & Bubbling

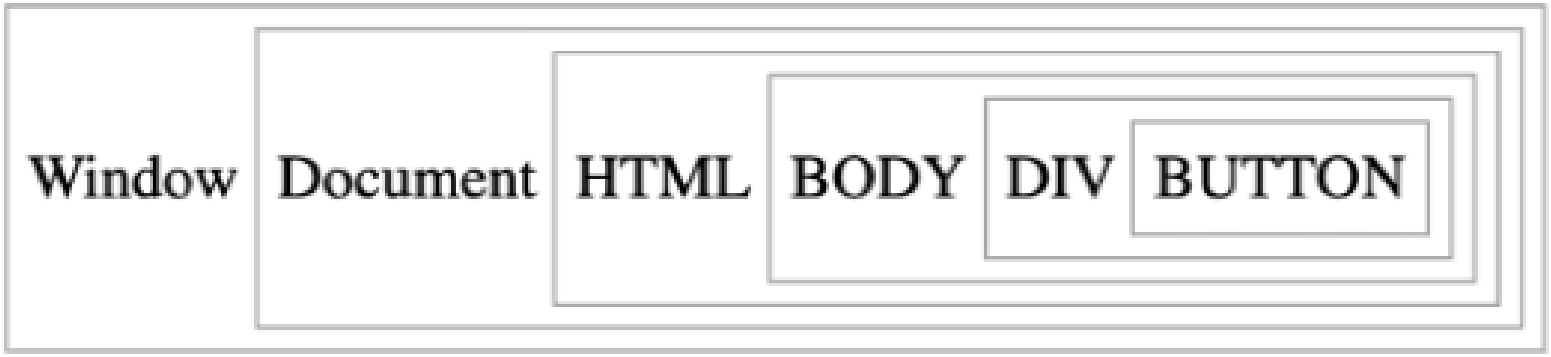
Events und deren Wanderung

- Ein DOM-Event kann ein Mausklick, das Betätigen einer Taste oder ähnliches sein.
- Man kann für jedes Element ein oder mehrere Eventhandler definieren.
  - Diese werden ausgeführt, falls das Event auftritt.

- Ein vom Browser ausgelöstes Event, z.B. ein Buttonklick, wandert von **`window`** zu dem Element, dass das Event ausgelöst hat.
  - Anschließend wandert das Event zurück zu **`window`**.
  - Die erste Phase nennt man Capture, die anderen Bubbling.
  - Bei jedem Schritt dieser Wanderung werden die Eventhandler des aktuellen Elements aufgerufen.
  - Die Wanderung kann durch eine API unterbrochen werden.

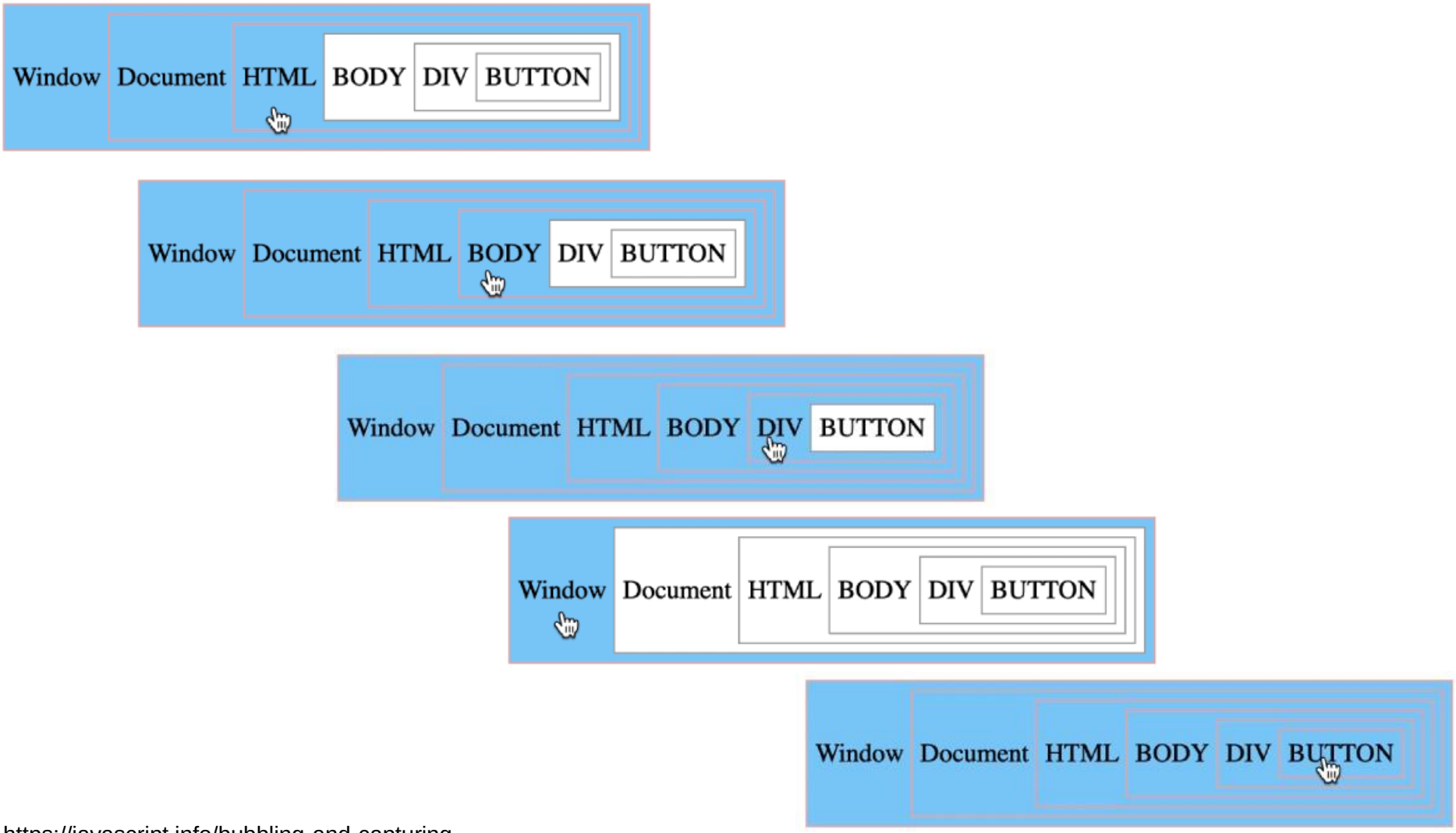
Events und deren Wanderung: Beispiel

- Das nun folgende Beispiel demonstriert, wie Events bei Auslösung wandern.
- Sehen Sie sich die folgenden Screenshots an und programmieren Sie das Beispiel mit dem hier zur Verfügung gestellten Code nach.
- Versuchen Sie den Code nachzuvollziehen.



https://javascript.info/bubbling-and-capturing

Events und deren Wanderung: Beispiel



https://javascript.info/bubbling-and-capturing

Events und deren Wanderung: Beispiel

capture: Window
capture: Document
capture: HTML
capture: BODY
capture: DIV
capture: BUTTON
bubble: BUTTON
bubble: DIV
bubble: BODY
bubble: HTML
bubble: Document
bubble: Window

https://javascript.info/bubbling-and-capturing

Events und deren Wanderung: Beispiel - HTML-Teil

```
<div>Window
    <div>Document
        <div>HTML
            <div>BODY
                <div>DIV
                    <div>BUTTON</div>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- Auf Basis von: https://stackoverflow.com/a/4616720 -->
```

Events und deren Wanderung: Beispiel - CSS-Teil

```css
p {
    line-height: 0;
}

div {
    display:inline-block;
    padding: 5px;
    background: #fff;
    border: 1px solid #aaa;
    cursor: pointer;
}

div:hover {
    border: 1px solid #faa;
    background: #87CEFA;
}
```

Events und deren Wanderung: Beispiel - JavaScript-Teil

```
var storage = "";
var flag = true;

function log(msg) {
    storage += msg + "\n";
    if(flag) {
        flag = false;
        setTimeout(function() {alert(storage);flag = true;storage = "";}, 500);
    }
}

function capture() {
    log('capture: ' + this.firstChild.nodeValue.trim());
}

function bubble() {
    log('bubble: ' + this.firstChild.nodeValue.trim());
}

var divs = document.getElementsByTagName('div');
for (var i = 0; i < divs.length; i++) {
    divs[i].addEventListener('click', capture, true);
    divs[i].addEventListener('click', bubble, false);
}
```

Events und deren Wanderung: Beispiel

capture: Window
capture: Document
capture: HTML
capture: BODY
capture: DIV
capture: BUTTON
bubble: BUTTON
bubble: DIV
bubble: BODY
bubble: HTML
bubble: Document
bubble: Window

https://javascript.info/bubbling-and-capturing