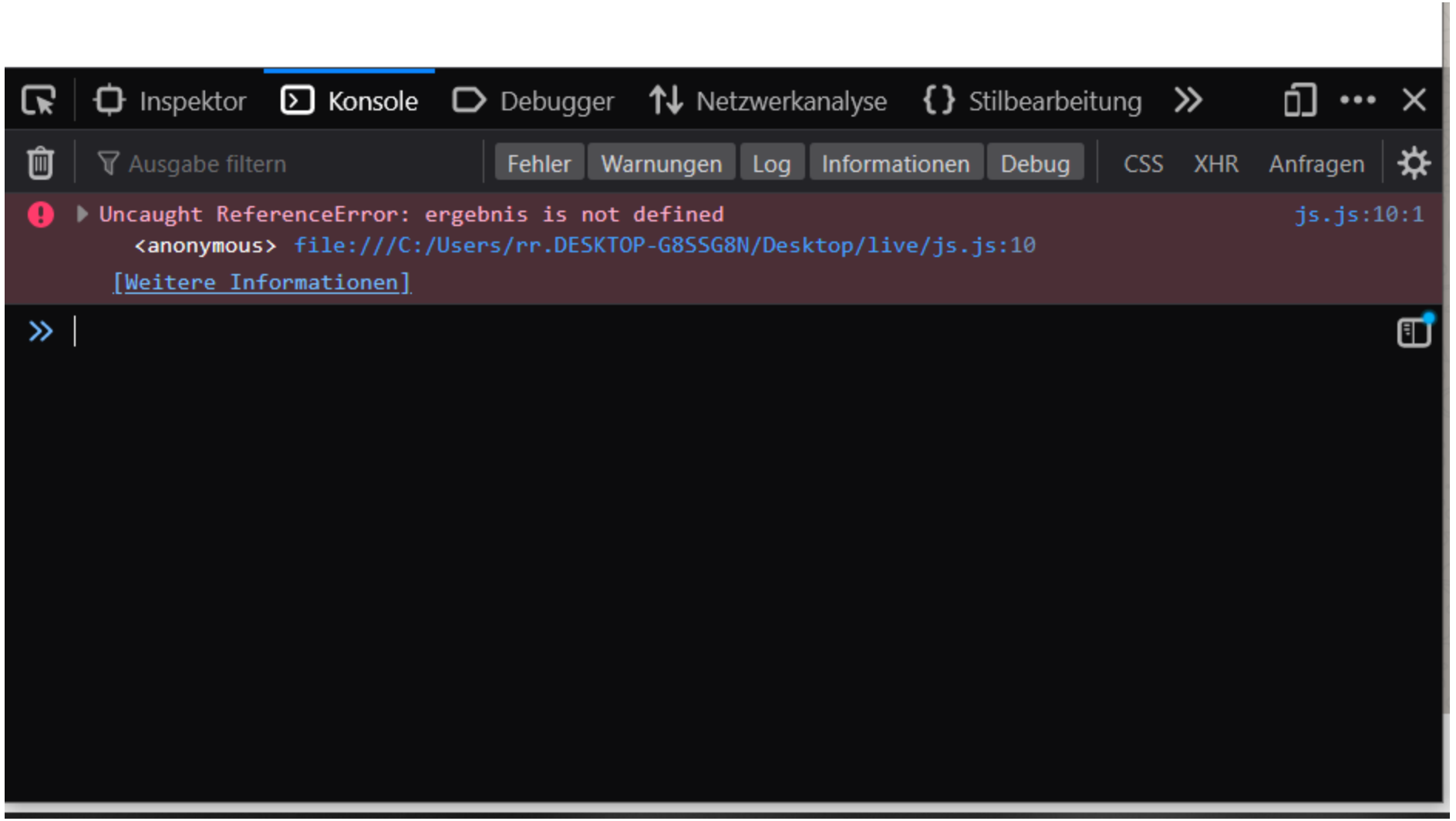
JavaScript 02

Fehlerbehandlung in JavaScript

SYNTAXFEHLER



Ausnahmen für Laufzeitfehler erstellen

- Mit `try` versucht das Programm die gewünschte Aktion auszuführen
- Ein `try`-Block darf niemals allein stehen. Es muss sich ein `catch`-Block anschließen

```
"use strict";

let a = 1.12345678912345678912345678 9;
let x = prompt("Wie viele Stellen sollen angezeigt werden?");
try {
    // reduziert die Anzahl der Nachkommastellen auf die Zahl
    // die der User eingibt
    let b = a.toPrecision(x);
    document.write("Wert mit der gewünschten Präzision: " + b + "<br>");
}
catch {
    alert("Gib einen Wert zwischen 1 und 100 ein!");
}
document.write("Weitere Inhalte");
```

- Bei Laufzeitfehler, erzeugt JavaScript automatisch ein Objekt, das Details zum auftretenden Problem enthält
- Mit `err` kann auf das Fehler-Objekt zugegriffen werden
- Attribute `name` ist die Bezeichnung des Fehlers
- Attribut `message` ist die Fehlermeldung

```
// Alternative für catch Block
catch(err) {
    alert(err.name);
    alert(err.message);
}
```

JavaScript und Webbrowser

- Wesentliches Hilfsmittel für Interaktion zwischen Webbrowser und JS-Programm = Events
  - Wenn Anwender bestimmte Aktion durchführt, löst der Browser entsprechendes Event aus
  - JS besitzt zahlreiche vorgefertigte Funktionen
- Browser Object Model (BOM)
  - Dient dazu, die grundlegenden Eigenschaften des Browsers zu steuern
  - Besondere Bedeutung: `window`-Objekt (folgt noch ausführlicher)
    - → Grundlegende Eigenschaften des Fensters wie Größe festlegen
    - → `alert-` und `prompt`-Befehl gehören zu diesem Element
    - → `window`-Objekt muss nicht ausdrücklich im Programm genannt werden (`alert()` ist eigentlich `window.alert()`)
- Document Object Model (DOM)
  - Abgeleitet vom BOM
  - Bsp für `document`-Objekt: `document.write`-Befehl (`document.write()` ist eigentlich `window.document.write()` muss aber nicht geschrieben werden)
- CSS Object Model (CSSOM)
  - Große Bedeutung
  - Layout-Vorgaben die mit CSS angefertigt wurden, werden verändert (Schriftfarbe, Hintergrundfarbe, Größe, …)

Browser Events

- Einige der wichtigsten Events:
  - `click`: Klick auf ein beliebiges Element
  - `contextmenu`: Klick mit der rechten Maustaste auf ein beliebiges Element
  - `mouseover`: Cursor wird auf ein Element bewegt
  - `mouseout`: Cursor wird von einem Element entfernt
  - `mousedown`: Maustaste wird gedrückt
  - `mouseup`: Maustaste wird losgelassen
  - `mousemove`: Maus wird bewegt
  - `dblclick`: Doppelklick auf das Element
  - `submit`: Formular wird abgeschickt
  - `focus`: Anwender setzt den Fokus auf ein Element
  - `keydown`: Drücken einer Taste auf der Tastatur
  - `keyup`: Loslassen einer Taste auf der Tastatur
  - `DOMContentLoaded`: Wird ausgelöst, wenn der HTML-Inhalt der Seite geladen ist
  - `transitioned`: Ende einer CSS-Animation

Auf Events reagieren
Beispiel 1

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
    </head>
    <body>
        <input value="Hier klicken!" onclick="alert('Du hast den Button geklickt!')" type="button">
    </body>
</html>
```

Auf Events reagieren
Beispiel 2

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
    </head>
    <body>
        <input value="Hier klicken!" onclick="verdopplung()" type="button">
        <script>
            function verdopplung() {
                let zahl = prompt("Gib eine Zahl ein:");
                zahl *= 2;
                alert("Doppelter Wert: " + zahl);
            }
        </script>
    </body>
</html>
```

Auf Events reagieren
Beispiel 3

- Auf HTML-Element per `id`-Attribut zugreifen

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
    </head>
    <body>
        <p> Absatz 1</p>
        <p id="absatz">Absatz 2</p>
        <p>Absatz 3</p>
        <script>
            // Name des entsprechenden HTML-Elements nennen
            // nach dem Punkt folgt Art des Events mit vorangestelltem Präfix "on"
            // diesem Ausdruck dann eine Funktion zuweisen
            absatz.onmouseover = function() {
                alert("Hier befindet sich Absatz 2..");
            }
        </script>
    </body>
</html>
```

```
// Variante 2
function nachricht() {
    alert("Hier befindet sich Absatz 2..");
}
absatz.onmouseover = nachricht;
```

```
// Variante 3
function nachricht() {
    alert("Hier befindet sich Absatz 2..");
}
// Hier wird ein Event-Listener verwendet
// In diesem Beispiel bringt er keinen Vorteil
absatz.addEventListener("mouseover", nachricht);
```

Auf Events reagieren
Beispiel 4

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
    </head>
    <body>
        <button type="button" id="btn">Klick mich</button>
        <script>
            function nachricht1() {
                alert("Nachrit 1");
                btn.addEventListener("click", nachricht2);
                btn.removeEventListener("click", nachricht1);
            }
            function nachricht2() {
                alert("Nachricht 2");
                btn.addEventListener("click", nachricht1);
                btn.removeEventListener("click", nachricht2);
            }
            btn.addEventListener("click", nachricht1);
        </script>
    </body>
</html>
```

Beispiel 5

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            body {
                padding: 25px;
            }
        </style>
    </head>
    <body id="b">
        <div id="d">
            Das ist ein div-Element
            <p id="p">
                Hier steht ein Text, bei dem ein Teil
                <strong>Fett markeirt</strong> ist.
            </p>
        </div>
        <script>
            function bodyTag() {
                alert("Body-Tag");
                // das Event wirkt sich nicht mehr auf
                // die übergeordneten Bereiche aus
                // mit event.stopPropagation()
                this.style.backgroundColor = 'green';
                event.stopPropagation();
            }
            b.onclick = bodyTag;

            function divTag() {
                alert("Div-Tag");
                this.style.backgroundColor = 'yellow';
                event.stopPropagation();
            }
            d.onclick = divTag;

            function pTag() {
                alert("P-Tag");
                this.style.backgroundColor = 'grey';
                event.stopPropagation();
            }
            p.onclick = pTag;

            function strongTag() {
                alert("strong-Tag");
                this.style.backgroundColor = 'lightblue
';

                event.stopPropagation();
            }
            p.onclick = strongTag;
        </script>
    </body>
</html>
```

https://www.mediaevent.de/javascript/event-handler-default-verhindern.html

EVENTS DELEGIEREN

```
<body id="bodypoint">
        <div>
            <p>Absatz 1</p>
            <p>
                Hier steht ein Text, bei dem ein Teil
                <strong>Fett markeirt</strong> ist.
            </p>
            <p>Absatz 3</p>
        </div>
        <script>
            function hintergrund() {
                // target = Zielelement auf das in diesem Beispiel geklickt wird
                // closest = gibt das Element mit der entsprechenden Bezeichnung
                // zurück, das das auslösende Element umfasst
                let p = event.target.closest('p');
                p.style.backgroundColor = 'green';
            }
            bodypoint.onclick = hintergrund;
        </script>
    </body>
```

Mouse- und Keyboard-Events

```
<body>
    <input value="Hier klick" id="btn" type="button">
    <script>
        function click() {
            alert("Click-Event");
        }
        function mdown() {
            alert('Mousedown-Event');
        }
        function mup() {
            alert("Mouseup-Event");
        }
        btn.onclick = click;
        btn.onmousedown = mdown;
        btn.onmouseup = mup;
    </script>
</body>
```

```
<body>
    <input id="eingabe" type="input">
    <script>
        function tastatureingabe(e) {
            alert("Key: " + e.key + "\nCode: " + e.code);
        }
        eingabe.onkeypress = tastatureingabe;
    </script>
</body>
```

- onClick = eigentlicher Mouseklick
- onmousedown = wenn Mousetaste losgelassen wird
- Onmouseup = wenn Mousetaste gedrückt wird

AUFGABE

1. Gestalte eine Seite, mit vier verschiedenen HTML-Elementen. Jedes von ihnen soll auf eine andere Art vom Event reagieren. Gestalte die passenden Event-Handler dafür
2. Schreibe ein Programm mit vier ineinander verschachtelten Elementen. Jedes von ihnen soll einen Event-Handler erhalten. Das Programm soll dem Element, das angeklickt wurde, einen grünen Hintergrund geben. Falls es sich dabei nicht bereits um das äußerste Element handelt, soll das darüber liegende Element einen gelben Hintergrund bekommen. Die übrigen Elemente sollen hingegen nicht verändert werden
3. Erstelle eine Seite mit einer HTML-Tabelle. Wenn der Anwender auf eines der enthaltenen Felder klickt, soll dieses eine rote Hintergrundfarbe erhalten. Verwende hierfür einen einzigen Event-Handler, der für alle Felder gilt – unabhängig davon, wie viele Spalten und Zeilen die Tabelle enthält. Das Programm soll immer das komplette Feld markieren, auch wenn innerhalb des td-Tags noch weitere HTML-Tags enthalten sind

DAS WINDOW-OBJEKT

- Grundlage aller weiteren Objekte in JS, daher auch root-Objekt genannt
- Beispiele: `alert-`, `confirm-`,`prompt`-Befehl
- Der `confirm`-Befehl gibt ein `true` oder `false` zurück

```
<script>
    let zahl = prompt("Gib eine Zahl ein");
    alert("eingegebene Zahl: " + zahl);
    let bestaetigung = confirm(decodeURI("Bestätige die Eingabe"));
    if(bestaetigung) {
        document.write("Eingabe bestätigt");
    } else {
        document.write("Eingabe nicht bestätigt");
    }
</script>
```

Fenster schließen und neue Fenster öffnen

- Verhalten nicht vorhersehbar da Befehle bei vielen Browsern starken Einschränkungen unterliegen
- Weitere Befehle
  - windwo.resizeTo()
  - windwo.resizeBy()
  - window.moveTo()
  - window.moveBy()

```
<script>
    window.open("stop.html");
</script>
```

```
<script>
    window.alert("Stop");
    window.close();
</script>
```

- stop.html

```
<script>
    let hoehe = window.innerHeight;
    let breite = window.innerWidth;
    document.write(hoehe + "px Höhe<br>" + breite + "px Breite");
</script>
```

- Höhe und Breite abfragen

Den zeitlichen Ablauf steuern

- Verzögerung der Nachricht durch `setTimeout()`

```
<button onclick="start()">Start</button>
<script>
    function nachricht() {
        alert("Button vor 2 Sekunden gedrückt");
    }
    function start() {
        setTimeout(nachricht, 2000);
    }
</script>
```

```
<script>
    function nachricht() {
        alert("Button vor 2 Sekunden gedrückt");
    }
    function nachricht2() {
        alert("Button vor 4 Sekunden gedrückt");
    }
    function start() {
        setTimeout(nachricht, 2000);
        setTimeout(nachricht2, 4000);
    }
</script>
```

Den zeitlichen Ablauf steuern

```
<button onclick="start()">Start</button>
<button onclick="stop()">Stop</button>
<script>
    let to;
    function f() {
        alert("Button vor 5 Sekunden gedrückt");
    }
    function stop() {
    // Mit clearTimeout() kann Timeout unterbrochen werden
        clearTimeout(to);
    }
    function start() {
        to = setTimeout(f, 5000);
    }
</script>
```

Den zeitlichen Ablauf steuern

```
<script>
    let i = 1;
    function f() {
        document.write(i + "<br>");
        i++;
    }
    // setInterval() ähnlich, Programm ruft
    // entsprechende Funktion immer wieder
    // aufs Neue auf
    setInterval(f, 1000);
</script>
```

AUFGABE

1. Erstelle drei verschiedene HTML Seiten. Jede von ihnen soll einen Button enthalten. Wenn der Anwender auf diesen drückt, soll ihm jeweils eine Rechenaufgabe gestellt werden. Damit diese nicht immer gleich ist, ist es sinnvoll, hierfür Zufallszahlen zu verwenden.
   Wenn der Anwender die Aufgabe der ersten Seite richtig gelöst hat, ruft das Programm automatisch die zweite Seite in einem neuen Tab oder Fenster auf. Wenn der Besucher auch die zweite Aufgabe richtig gelöst hat, öffnet das Programm die dritte Seite und schließt die aktuelle Seite. Wenn er auch die dritte Aufgabe richtig löst, soll auch diese Seite geschlossen werden, nachdem eine entsprechende Meldung ausgegeben wurde.
   Achtung: PopUp Blocker im Browser deaktivieren
2. Das Programm von Seite 19 wird endlos weitergeführt. Sorge dafür, dass es nach einer Minute beendet wird. Nutze hierfür den `clearInterval()`-Befehl. Dieser wird auf die gleiche Weise wie die `clearTimeout()`-Methode verwendet.

ENDE

QUELLE: JAVASCRIPT
PROGRAMMIEREN FÜR EINSTEIGER
ISBN: 978-3-96645-016-4