Webentwicklung – Teil 4

DHBW SS2018 – Elias Henrich

Überblick Heute

- Wdh.: Externe CSS-Datei + Box-Model
- z-index
- Transitions & Animations
- CSS-Funktionen und -Filter
- Einführung JavaScript

Externe CSS-Dateien

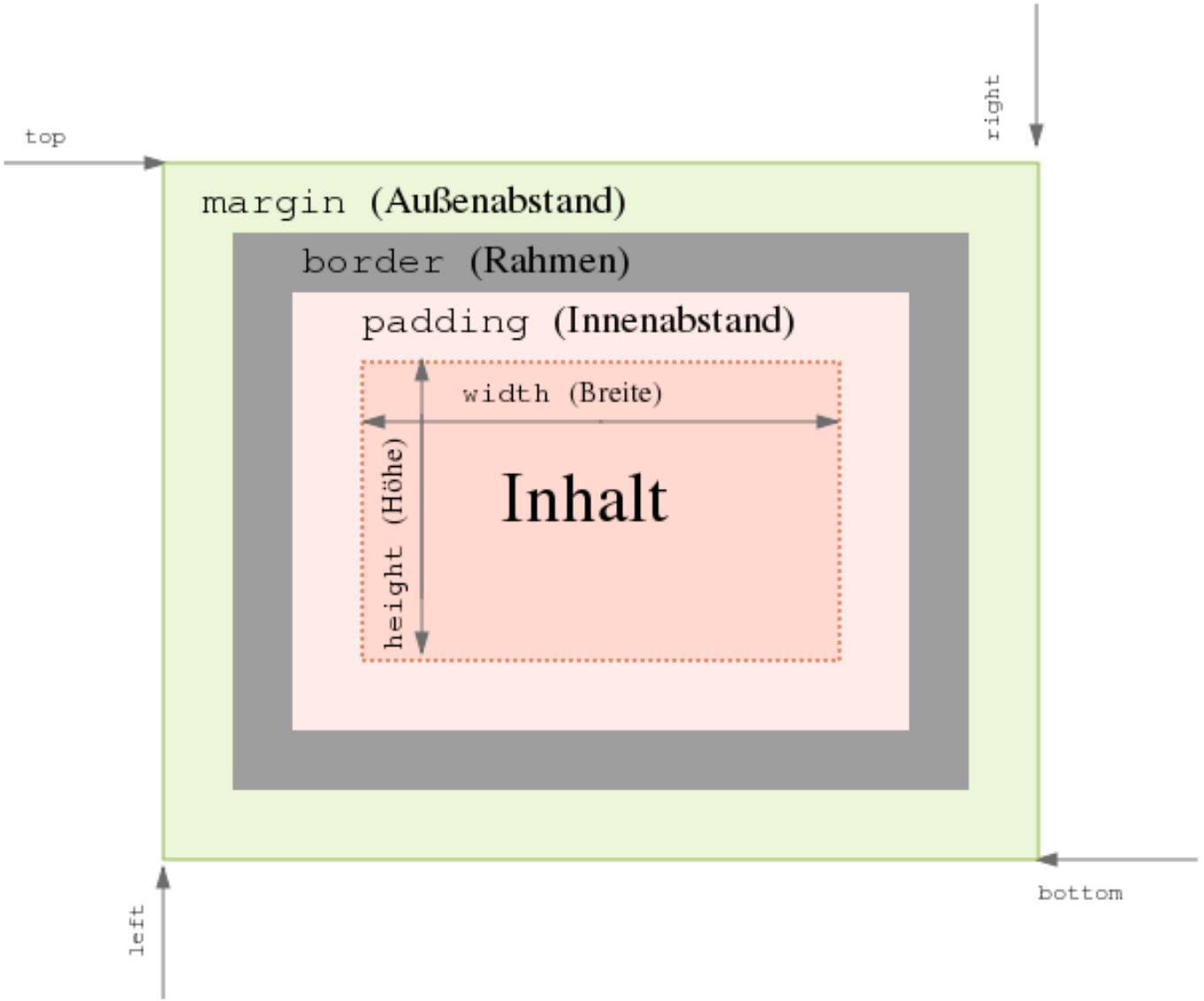
Einbinden im Head-Bereich:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Seitentitel</title>
        <link rel="stylesheet" href="pfad/zum/stylesheet.css">
        <link rel="stylesheet" href="pfad/zum/zweitenStylesheet.css">
    </head>
```

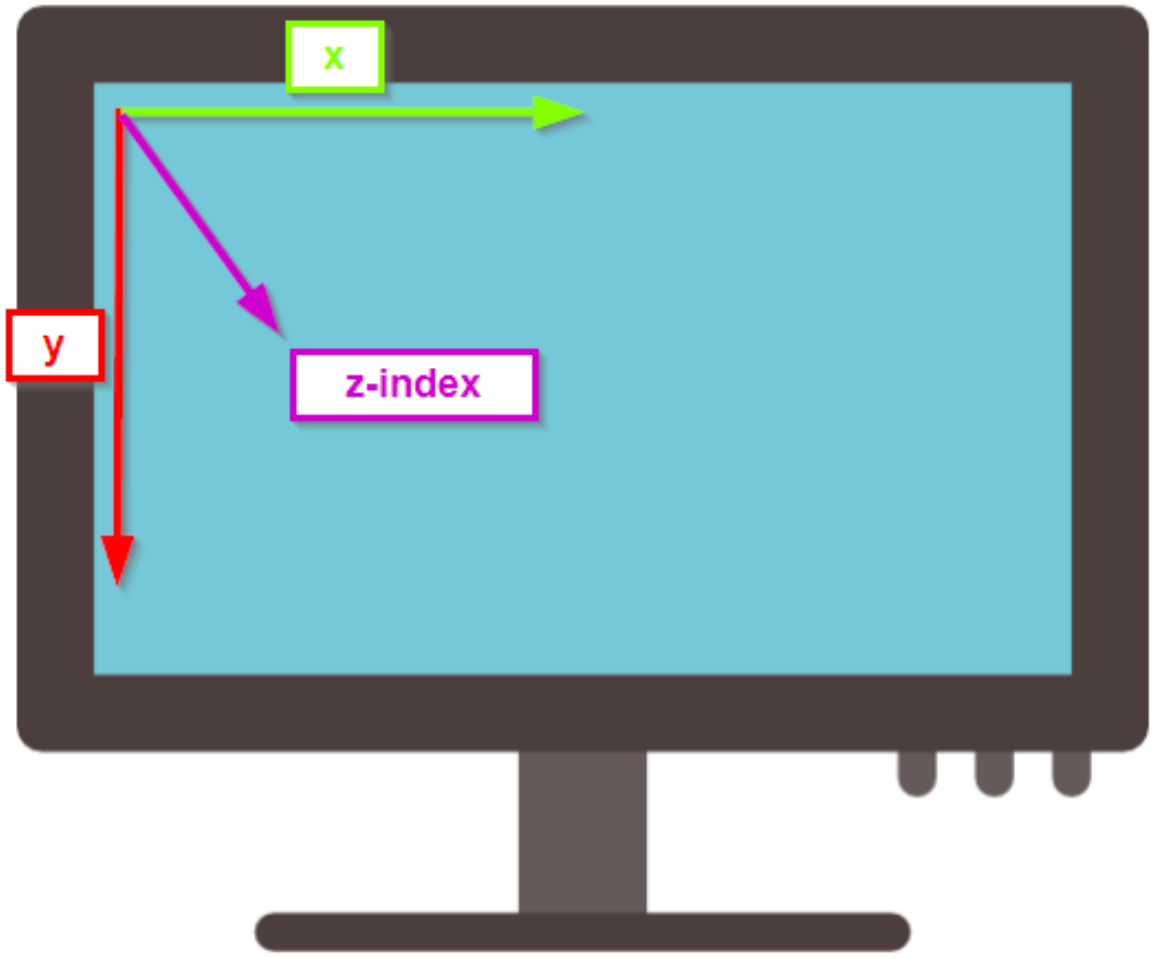
`<link>`	Verbindung zu anderer Datei
`rel`	Beziehungstyp („*relationship*")
`href`	Pfad zur referenzierten Datei

<link> ist ein empty Tag!

Box-Model

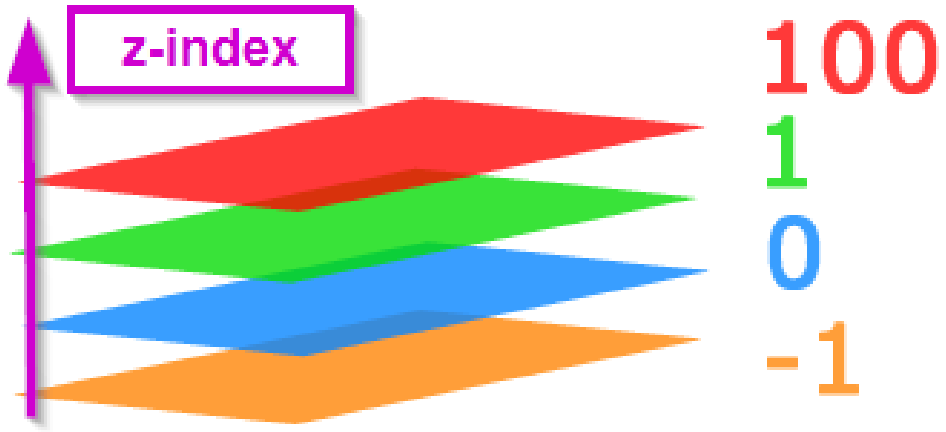

top
right
margin (Außenabstand)
border (Rahmen)
padding (Innenabstand)
width (Breite)
height (Höhe)
Inhalt
left
bottom

Positionierung: z-index



Positionierung: z-index

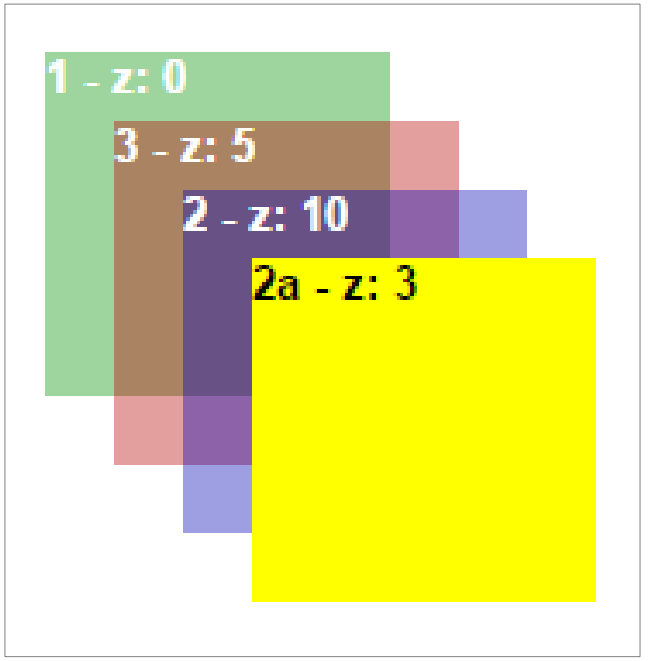
- nur für positionierte Elemente
  d.h. nicht bei static
- z-index: auto (default)
  Elemente überlappen in der Reihenfolge, wie sie im Quellcode auftauchen
- mit z-index:



Positionierung: z-index

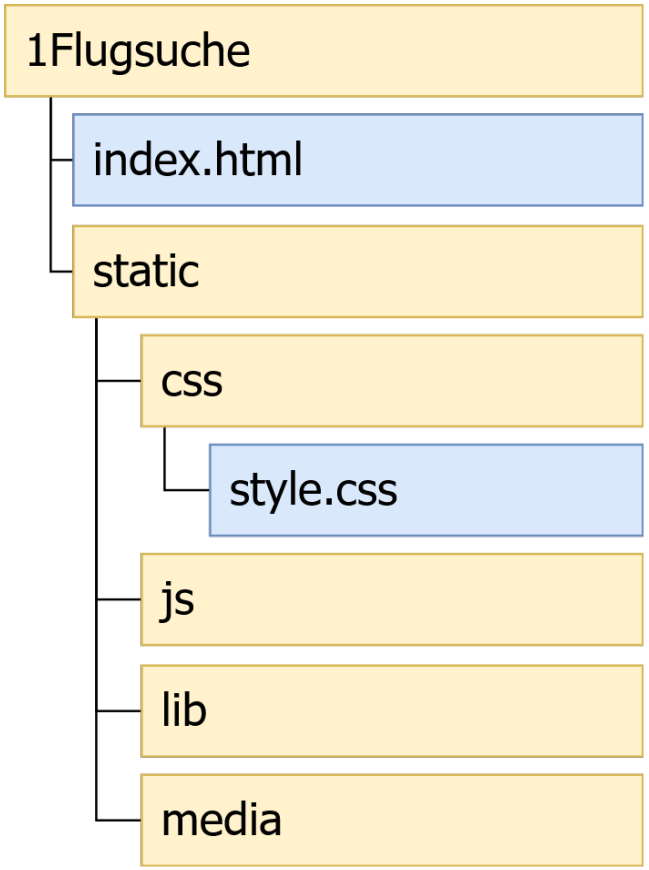
Elemente mit z-index erzeugen für Kinder einen **„eigenen Stapelkontext“**

```
<div class="wrapper">
    <div id="box1">
        1 - z: 0
    </div>
    <div id="box2">
        2 - z: 10
        <div id="box2a">
            2a - z: 3
        </div>
    </div>
    <div id="box3">
        3 - z: 5
    </div>
</div>
```



Projektstruktur

Beispielhafte Implementierung bis einschl. VL 04:
https://github.com/eliashenrich/dhbwWebDev



index.html im Projektroot ist meistens der Einstiegspunkt in eine Web-Anwendung

3-5 eigene CSS-Dateien typisch

Häufig ist eine analoge Verzeichnisstruktur zu finden

Transitions

„Weicher“ Übergang zwischen zwei Zuständen

```
.orangeBox {
    width: 100px;
    transition: width 2s;
}

.orangeBox:hover {
    width: 150px;
}
```

```
transition: p t[, p_n t_n];
```

p Property-Name
t Zeit bis Zielwert erreicht

Opt. weitere Wertänderungen, durch Komma getrennt.

Transition-Timing-Function

„Beschleunigungskurve“ der Wertänderung

```
transition-timing-function: ease;
```

(default)

Bildquelle: https://stackoverflow.com/questions/39735012/

Übung: Animieren!

Welche vier Zustände besitzen die Textfelder?

Gestalten Sie den Übergang der Hintergrundfarbe aller Inputfelder „weicher“:

- Dauer: 0.4 Sekunden, Timing-Function: ease-in-out

Auch der Wechsel der BG-Color der Liste soll animiert werden, so dass beim Überfahren mehrerer Elemente eine „Spur“ nachgezogen wird.

Zwei Transition-Anweisungen notwendig:

1) 0.7s, Ease-in
2) 0.1s, Ease-out

Animationen

Übergange zwischen einer unbegrenzten Anzahl an Zuständen

```
@keyframes ruleName {
     from {
          left: 50px;
     }
     to {
          left: 200px;
     }
}
```

```
.orangeBox {
     width: 100px;
     animation: ruleName;
     animation-duration: 3s;
}
```

Animationen

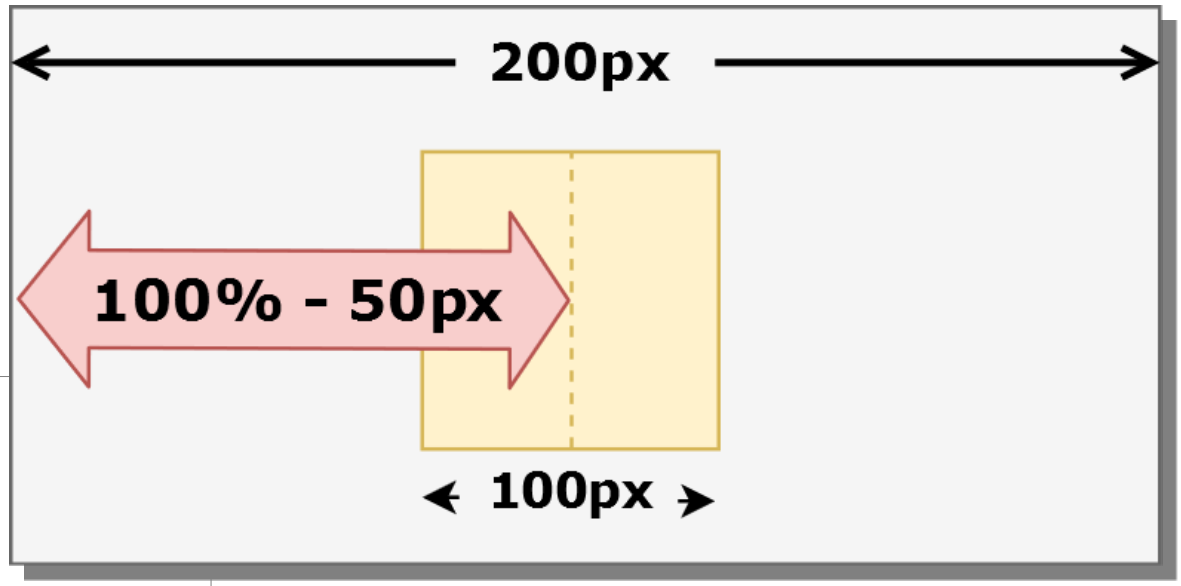
Übergange zwischen einer unbegrenzten Anzahl an Zuständen

```
@keyframes ruleName {
    0% {
        width: 100px;
        height: 50px;
    }
    50% {
        width: 200px;
        height: 50px;
    }
    100% {
        width: 200px;
        height: 100px;
    }
}
```

```
.orangeBox {
    width: 100px;
    animation: ruleName;
    animation-duration: 3s;
}
```

CSS-Funktionen: calc()

→ Eigenschaften mit Grundrechenarten dynamisch berechnen




```
.boxWrapper {
    width: 200px;
    height: 50px;
}

.image {
    width: 100px;
    left: calc(100% - 50px);
}
```

CSS-Funktionen: calc()

→ Eigenschaften mit Grundrechenarten dynamisch berechnen

+ / -	**Addition / Subtraktion von Größenangaben**	`calc(100% - 5em)`
*	**Multiplikation einer Größenangabe mit Faktor**	`calc(100% - 2 * 5em)`
/	**Division einer Größenangabe durch einen Divisor**	`calc((100% - 5em) / 2)`

Weitere CSS-Funktionen

Elemente	attr()	a:after {
		content: " (" attr(href) ")";
		}

Gradienten	linear- gradient()	linear-gradient(red, yellow, blue);
Farbräume	hsl()	hsl(120,100%,50%; /* grün */

→ Insgesamt 12 Funktionen seit CSS3:
https://www.w3schools.com/cssref/css_functions.asp

filter

„...achieve varying visual effects (sort of like Photoshop filters for the browser)“

```
.box img {
    filter: blur(5px) grayscale(100%);
}
```

filter

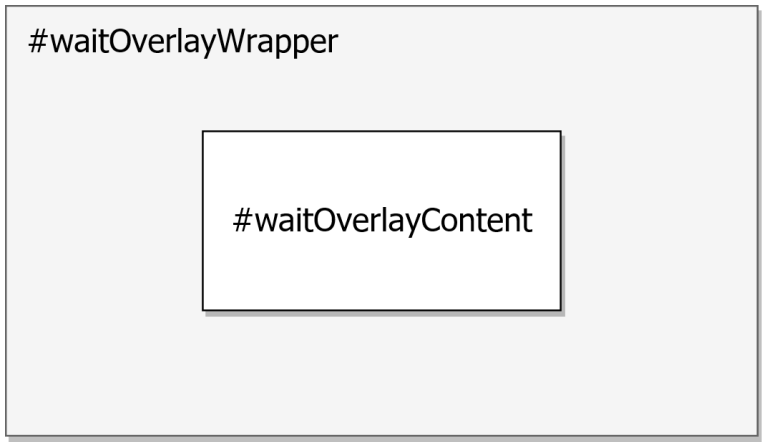
- blur(5px)
- brightness(200%)
- contrast(200%)
- drop-shadow(8px 8px 10px red)
- grayscale(100%)
- hue-rotate(90deg)
- invert(100%)
- opacity(30%)
- saturate(8)
- sepia(100%)
- **contrast(200%) brightness(150%)**
- none

Genauere Infos z.B.
bei W3Schools

Filter lassen sich
verketten und dadurch
viele Effekte erzielen

Übung: Wait-Overlay 1/2

Wenn eine asynchrone Request (Flugsuche) gestellt wird, soll während der Wartezeit ein „Bitte warten"-Fenster angezeigt werden.



div#waitOverlayWrapper:

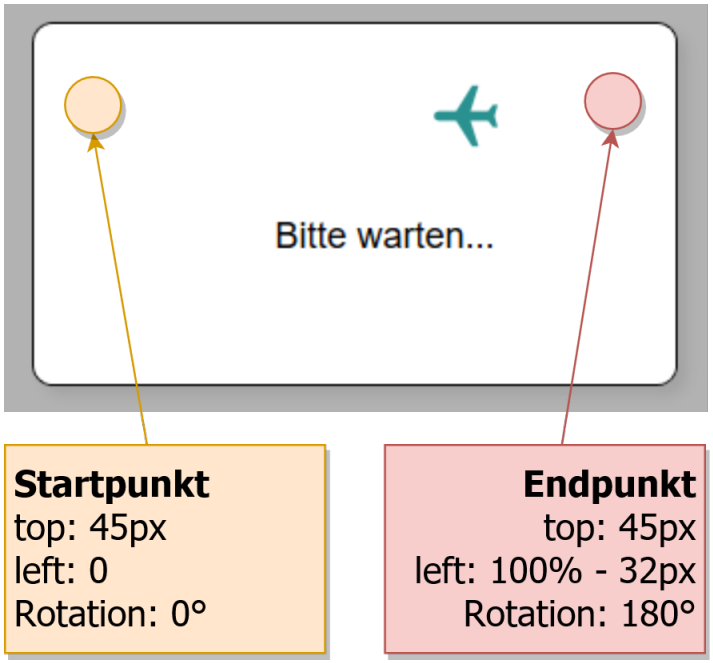
- width:100%, height: 100%
- überlagert restlichen Content
- Hintergrund: #464646, Transparenz 40%

div#waitOverlayContent:

- width: 20%, height: 150px
- vertikal & horizontal zentriert
- Schlagschatten: 5px 5px 5px #9F9F9F
- Border-Radius: 10px
- Hintergrund: weiß

Übung: Wait-Overlay 2/2

Wenn eine asynchrone Request (Flugsuche) gestellt wird, soll während der Wartezeit ein „Bitte warten"-Fenster angezeigt werden.



- Fügen Sie eine Flugzeuggrafik* (#008B8B, 32px * 32px) ein
- Das Flugzeug bewegt sich kontinuierlich von links nach rechts und zurück → Keyframes:
  1) Startpunkt (Rotation 0°)
  2) Endpunkt vor Rotation
  3) Endpunkt nach Rotation
  4) Startpunkt vor Rotation
- Textmeldung „Bitte warten"

→ Fügen Sie abschließend `.overlayWrapper` die Eigenschaft `display: none;` hinzu

* Flugzeug und andere Icons: https://www.flaticon.com/free-icon/black-plane_61212

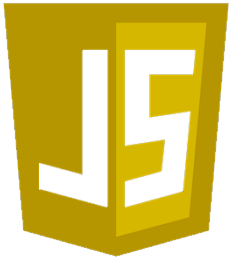
Abschluss CSS

Fragen & Wünsche CSS

Tools des Webentwicklers

HTML
beschreibt die Inhalte

CSS
legt das Layout fest



JavaScript
steuert das Verhalten

Was ist JavaScript?

JavaScript is the programming language of
HTML and the Web.

JavaScript is easy to learn.

Quelle: https://www.w3schools.com/js/default.asp

JavaScript (JS) ist eine leichtgewichtige, interpretierte oder
JIT-übersetzte Sprache mit First-Class-Funktion.

Quelle: https://developer.mozilla.org/de/docs/Web/JavaScript

JS einbinden

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charset="UTF-8">

        <script>
            /* Platz für JavaScript */
        </script>

    </head>
    <body>
        <p>
            Ich bin der Body
        </p>

        <script>
            /* Platz für JavaScript */
        </script>

    </body>
</html>
```

Interne Anweisungen:

- in `<head>` oder `<body>`
- innerhalb von `<script>`
- `type`-Attribut nicht notwendig!
- beliebig viele `<script>`-Abschnitte erlaubt

JS einbinden

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charset="UTF-8">

         <script src="script.js"></script>

    </head>
    <body>
        <p>
            Ich bin der Body
        </p>
    </body>
</html>
```

Externe JS-Dateien:

- in `<head>` oder `<body>`
- Tag mit Attribut `src` und Angabe einer belieb. Datei
- Dateiname muss mit `.js` enden
- mehrere Dateien erlaubt

JS einbinden

Externe JS-Datei

- Trennung von UI und Funktionalität
- jede externe JS-Datei benötigt eigene HTTP-Request
- große JS-Anwendungen durch Browser-Cache wesentlich schneller

empfehlenswert bei komplexen Programmen. Häufig sind Scripte kompakt und externe Dateien erschweren die Lesbarkeit des Codes

→ für unseren Anfang: INTERNE SKRIPTE

Was ist JavaScript?

- 1995 als *LiveScript* im Netscape Browser eingeführt
- außer Name kein Bezug zu Java, eher zu C
- offizieller Name ist *ECMAScript*
- zahlreiche Einsatzgebiete:
  - lokale Anwendung (Server-seitig, z.B. node.js, MS IIS*)
  - andere Geräte, z.B. Smartphones, µ-Controller / MCUs)
  - **Webbrowser (Client-seitig)**

* „Microsoft Internet Information Services“ – Server-Produktfamilie

Was ist JavaScript?

- dynamische Typisierung

```
<script>
    var a = 5;              // Typ: Number
    alert(a);

    a = "Hallo";            // Typ: String
    alert(a);

    a = function() {        // Typ: Function
        return 5;
    };
    alert(a);
    alert(a());
</script>
```

Was ist JavaScript?

- First-Class-Funktionen

```
<script>
    function addition(a, b) {
        return (a + b);
    }

    var subtraktion = function(a, b) {
        return (a - b);
    };

    alert(addition(5, 3));
    alert(subtraktion(5, 3));
    alert(addition(subtraktion(5,3), 4));
</script>
```

Was ist JavaScript?

- objektorientiert
  (aber bis vor kurzem ohne explizite Klassen!)

```
<script>
    var Fahrzeug = function(reifen) {
        this.anzReifen = reifen;
    };

    var rad = new Fahrzeug(2);
    var pkw = new Fahrzeug(4);
    var lkw = new Fahrzeug(8);

    alert("Rad: " + rad.anzReifen);
    alert("PKW: " + pkw.anzReifen);
    alert("LKW: " + lkw.anzReifen);
</script>
```

Was ist JavaScript?

- objektorientiert
  (aber bis vor kurzem ohne explizite Klassen!)

```
<script>
    var Fahrzeug = function(reifen) {
        this.anzReifen = reifen;
        this.speed = 0;

        this.beschleunigen = function() {
            this.speed += 5;
        };
    };

    var pkw = new Fahrzeug(4);
    pkw.beschleunigen();
    alert(pkw.speed);
    pkw.bremsen();
    alert(pkw.speed);
</script>
```

Syntax

- Anweisungen durch Semikolon getrennt
- Variablen durch Schlüsselwort `var` deklariert
- Nur wenige Datentypen:
  - Number Ganz- / Kommazahlen
  - String " oder '
  - Boolean
  - Objekte
    - Funktionen
    - Arrays

```
<script>
    var a = 3;
    var b = "1.5";
    alert(a / b);
</script>
```

Syntax

- Variablenbezeichnung:
  - case-sensitive
  - beginnen mit Buchstabe, Underscore oder $
  - Bindestrich nicht erlaubt!
  - Typisch „CamelCase“: firstName, lastName...
- White-Spaces werden ignoriert

  Wie immer wichtig: Korrekte Formatierung!

Syntax

- Kommentare

```
<script>
    // Das ist einzeiliger Kommentar
    var a = 3;
    alert(a);
</script>
```

```
<script>
    /*
     und dieser mehrzeilig
     */
    var a = 3;
    alert(a);
</script>
```

Operatoren 1

```
var a = 5;
var b = a;
```

Wertzuweisung

```
var a = 10;
var b = 3;
var c = a % b;
// c = 1
```

Arithmetische Operatoren

+ - / * %

```
var a = 10;
a += ++a;
// a = 21
```

Zuweisungsoperatoren

+= -= ++ -- /= *=

Operatoren 2

Vergleichsoperatoren

- equal value
- equal value and type
- larger / smaller than

```
var a = 10;
b = "10";

var c = (a == b);     // true
var d = (a === b);    // false
```

Logische Operatoren

- and
- or
- not

```
if (!(c === true || d === true)) {
    alert("Keiner ist true!");
}
```

Arrays

Index-basiert

```
var a = [];
alert(a.length);     // 0
a[0] = 5;
a[1] = 2;
alert(a.length);     // 2
```

Assoziativ

```
var a = [];

a['vorname']  = 'Elias';
a['nachname'] = 'Henrich';

alert(a['nachname']);        // Henrich
```

Arrays

Initialwerte

```
var a = {
    0: 5,
    1: 2,
    'vorname': 'Elias',
    'nachname': 'Henrich'
};

alert(a[0]);                // 5
alert(a['nachname']);       // Henrich
alert(a.nachname);          // Henrich
alert(a.0);                 // UNGÜLTIG!
```

ARRAY == OBJEKT == ARRAY

Konditionen: If

```
var a = 5;
var b = "5";

if (a === b) {
    alert("Gleich gleich gleich");
} else if (a == b) {
    alert("Gleich gleich");
} else {
    alert("Ungleich");
}
```

Konditionen: Switch

```
var a = 6;

switch (a) {
    case 3:
        alert("Drei");
        break;
    case 4:
        alert("Vier");
        break;
    default:
        alert("Was anderes");
}
```

Break nicht vergessen!

Schleifen: for

```
var a = 0;

for (var i = 0; i < 5; i++) {
    a += i;
}

alert(a);    // 10
```

Schleifen: `while`

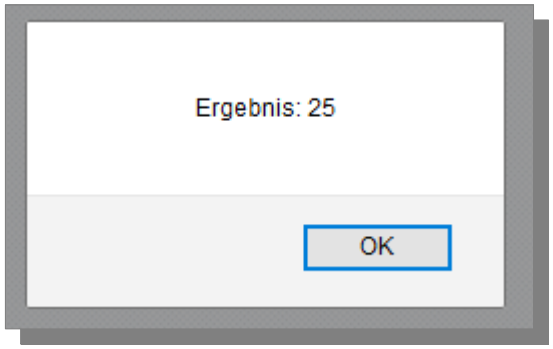
```
var a = 0;

while (a <= 10) {
    a++;
}

alert(a);    // 10? 11?
```

```
var a = 0;

do {
    a++;
} while (a <= 10);

alert(a);    // 10? 11?
```

Ausgabe

- Message-Box

```
var a = 10;
var b = 15;
var c = a + b;

alert("Ergebnis: " + c);
```



Ausgabe

- Message-Box
- HTML-Body anhängen

```
var a = 10;
var b = 15;
var c = a + b;

document.write("Ergebnis: " + c);
```

Hier ist der Body...

Ergebnis: 25

Ausgabe

- Message-Box
- HTML-Body anhängen
- Inhalt eines HTML-Elements ändern (Id)

```
<p>
    Ergebnis: <span id="result">-</span>
</p>

<script>
    var a = 10; var b = 15;
    var c = a + b;

    document.getElementById('result').innerHTML = c;
</script>
```

Ausgabe

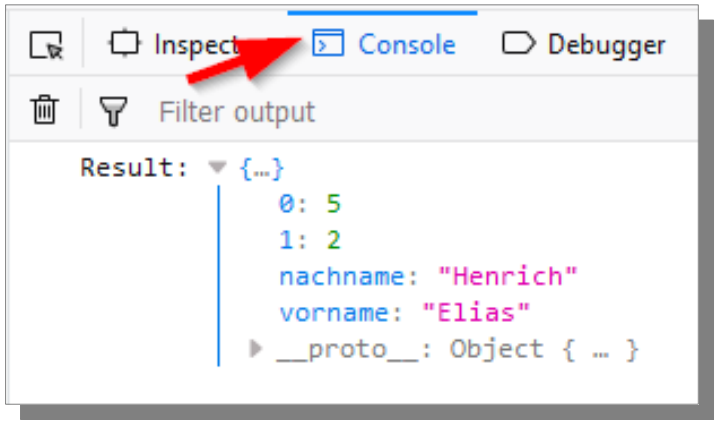
- Message-Box
- HTML-Body anhängen
- Inhalt eines HTML-Elements ändern (Class)

```
<p class="absatz">Erster Absatz</p>
<p class="absatz">Zweiter Absatz</p>

<script>
    var a = 10; var b = 15;
    var c = a + b;

    var elemente = document.getElementsByClassName('absatz');
    for (i = 0; i < elemente.length; i++) {
        elemente[i].innerHTML = "Ergebnis: " + c;
    }
</script>
```

Ausgabe

- Message-Box
- HTML-Body anhängen
- Inhalt eines HTML-Elements ändern
- Browser-Konsole (Developer Tools öffnen)

```
var a = {
    0: 5,
    1: 2,
    'vorname': 'Elias',
    'nachname': 'Henrich'
};

console.log("Result:", a);
```



Übung: Overlay einblenden

1. Fügen Sie dem Formular folgendes Attribut hinzu:

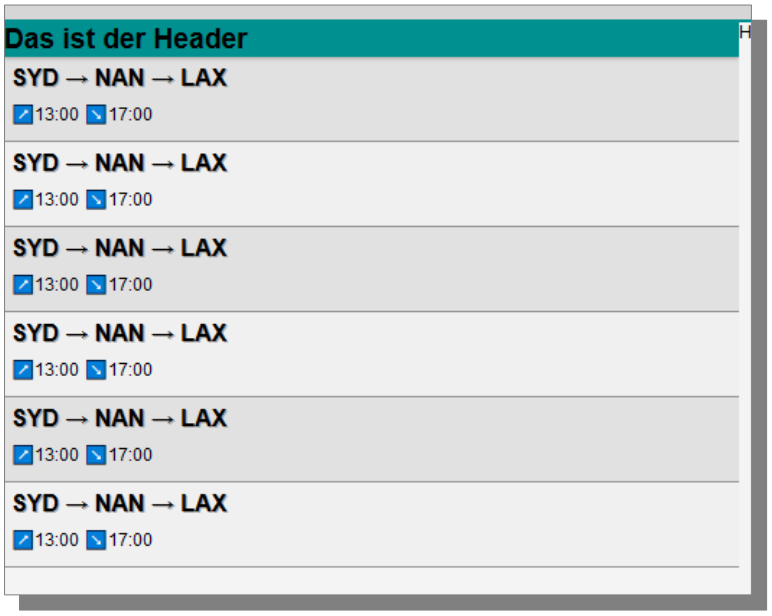
```
onsubmit="return showOverlay()"
```

2. Erstellen Sie eine JS-Funktion `showOverlay()`, die:
   - das Overlay in eine Variable lädt
   - das Overlay einblendet → `element.style.display`
   - den Wert `false` zurückliefert

Nutzen Sie während der Entwicklung `console.log("Test")` um zu prüfen, ob die Funktion tatsächlich ausgeführt wird.

Übung: Overlay einblenden

1. Entfernen Sie, außer das erste, alle Listenelemente
2. Weisen Sie dem Element eine CSS-Klasse zu



Erstellen & testen Sie eine Funktion, die folgendes ausführt:

1. -Liste einer Variablen zuweisen
2. HTML-Code des Listenelements (`element.outerHtml`) einer Variablen zuweisen
3. Der -Liste den HTML-Code des Elements 5-mal hinzufügt

Nutzen Sie min. eine Schleife!

Ziel: Vorbereitung für das Einlesen der Flugdaten per AJAX