JavaScript

Was ist JavaScript?

Eine moderne Programmiersprache, die auf der C-Syntax basiert

Seit 1995 im Einsatz

JavaScript, HTML und CSS bilden zusammen die drei Kernsprachen für die Webentwicklung

Einstieg

Zum Schreiben von JavaScript reicht eine Datei mit der Endung .js

Keine speziellen Code-Markierungen nötig

Fokus: JavaScript, das client-seitig läuft, also im Browser auf Ihrem eigenen Rechner

JavaScript einbinden

Wie bei CSS mit style-Tags
können Sie JavaScript direkt
zwischen `<script>` Tags schreiben

Wie bei CSS mit `<link>` Tags können
Sie JavaScript auch in separaten
Dateien speichern und über das `src`
Attribut des `<script>` Tags einbinden

Variablen in JavaScript

Ähnlich zu Ihren Erfahrungen aus C und Python

Keine Typendeklaration erforderlich

Lokale Variablen werden mit `let` oder `const` deklariert

Beispiel:

```
let x = 44;
const pi = 3.14;
```

Kontrollstrukturen

Alle bekannten Konstrukte aus C verfügbar:

- if
- else if
- else
- switch
- ? … : … (*ternärer Operator*)

```
if (bedingung) {
    // Code
} else if (andereBedingung) {
    // Alternative
} else {
    // Sonst
}
```

Schleifen

Auch hier alle C-Konstrukte verfügbar:

- while
- do-while
- for

```
// Klassische for-Schleife
for (let i = 0; i < array.length; i++) {
    // Code
}
```

Und zusätzlich anders als in C:

- for…in (für Objekteigenschaften)
- for…of (für Array-Elemente)

```
// Moderne for...of Schleife
for (let element of array) {
    // Direkter Zugriff auf element
}
```

Funktionen

Werden mit dem Schlüsselwort `function` eingeführt

JavaScript-Funktionen können anonym sein

Eine Besonderheit: Funktionen als Event-Handler für HTML-Elemente

```
function beispielFunktion(parameter) {
    return parameter * 2;
}

// Anonyme Funktion
let verdoppeln = function(x) {
    return x * 2;
}
```

Arrays

Deklaration ist unkompliziert

Arrays können verschiedene Datentypen mischen

Verfügen über nützliche eingebaute Methoden wie `map()`, `filter()`, `push()`

```
let nums = [1, 2, 3, 4, 5];
let gemischt = [1, true, 3.333, 'text'];


// Array-Methode map() - ähnlich
// wie in Python
nums = nums.map(function(num) {
    return num * 2;
});
```

Arrays

```
let zahlen = [1, 2, 3, 4, 5];
let verdoppelt = zahlen.map(function(zahl) {
    return zahl * 2;
}); // Ergebnis: [2, 4, 6, 8, 10]
```

Arrays

```
// Unterschiede zu Python-Listen:
let arr = [1, 2, 3];
arr[5] = 6; // Kein IndexError! arr[3] und arr[4] bleiben leer (undefined)
console.log(arr.length); // 6, nicht 3!


// Vorsicht bei Vergleichen:
let a1 = [1, 2, 3];
let a2 = [1, 2, 3];
console.log(a1 == a2); // false! Arrays sind Referenztypen
```

Objekte

Ähnlich wie C-Structs

Bestehen aus Eigenschaften (properties) und Methoden

Flexible Key-Value-Paare

Grundbaustein für JSON

```
let auto = {
    jahr: 1963,
    modell: "Beetle"
};


// Zugriff wie bei C-Structs
auto.jahr = 1964;
```

Objekte

```
let auto2 = {
    jahr: 1963,
    marke: 'VW',
    modell: 'Beetle',
    beschreibung: function() {
        return this.marke + " " + this.modell;
    }
};


console.log(auto2.beschreibung()); // "VW Beetle"
```

Objekte

```
// Fallstrick 1: Undefined Properties
let benutzer = {name: "Max"};
console.log(benutzer.alter); // undefined
console.log(benutzer.alter.toString()); // TypeError!


// Lösung: Property-Existenz prüfen
if (benutzer.alter !== undefined) {
    console.log(benutzer.alter.toString());
}
```

Objekte

```
// Fallstrick 2: Properties vs Methoden
let produkt = {
    name: "Buch",
    preis: 9.99,
    // Methode
    getPreis: function() {
        return this.preis;
    }
};


let preisMethode = produkt.getPreis; // Vorsicht!
console.log(preisMethode()); // undefined
```

Events

Reaktion auf Benutzerinteraktionen

Event Handler als Callback-Funktionen

HTML-Elemente unterstützen Events als Attribute

Ereignisbasierte Programmierung

```
<button onclick="handleClick(event)">
    Klick mich
</button>
```

```
function handleClick(event) {
    // Reagiert auf Klick-Event
}
```

Events

```
<button onclick="gruesseNutzer(event)">Button 1</button>
<button onclick="gruesseNutzer(event)">Button 2</button>


<script>
    function gruesseNutzer(event) {
        let trigger = event.srcElement;
        alert('Sie haben ' + trigger.innerHTML + ' geklickt');
    }
</script>
```

Beispiele für Event-Handler

`onclick` – Mausklick

`onmouseover` – Maus bewegt sich über Element

`onkeydown` – Tastatureingabe

`onload` – Seite/Ressource geladen

```
// Beispiel für verschiedene Event-Handler
<input
    type="text"
    onkeydown="console.log('Taste gedrückt')"
    onmouseover="console.log('Maus darüber')"
    onfocus="console.log('Feld fokussiert')"
>
```

Beispiele für Event-Handler

```
<div>
    <span id="counter">0</span>
    <button onclick="increment()">+1</button>
</div>


<script>
    let count = 0;
    function increment() {
        count++;
        document.getElementById('counter').innerHTML = count;
        console.log('Neuer Zählerstand:', count);
    }
</script>
```

String-Interpolation mit Backticks

```
// Alte Stringverkettung
let name = "Welt";
console.log("Hallo " + name + "!");


// Moderne Template Literals
console.log(`Hallo ${name}!`);
```

String-Interpolation mit Backticks

```
// Typische Situation: Ein Objekt
const user = {
    name: "Maria",
    alter: 20,
    kurse: ["HTML", "CSS"]
};

// Alte Schreibweise - schwer lesbar
const text1 = user.name + " (" + user.alter + ") belegt " +
user.kurse.length + " Kurse: " + user.kurse.join(", ");

// Mit Template Literals - übersichtlicher
const text2 = `${user.name} (${user.alter}) belegt
${user.kurse.length} Kurse: ${user.kurse.join(", ")}`;

// es gibt noch weitere nützliche String- und Array-Methoden
```

String-Interpolation mit Backticks

```
// Typkonvertierung und parseInt
console.log("5" + 3); // "53"
console.log(parseInt("5") + 3); // 8
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Aufgabe 1

Gegeben ist folgender Code:

```
// Vervollständigen Sie die Funktion, sodass
// sie ein Array mit den verdoppelten Zahlen zurückgibt
let zahlen = [1, 2, 3, 4];


function verdoppleArray(arr) {
    // Ihre Lösung hier
}


// Erwartete Ausgabe: [2, 4, 6, 8]
```

Aufgabe 1

```
function verdoppleArray(arr) {
    return arr.map(function(zahl) {
        return zahl * 2;
    });
}
```

```
function verdoppleArray(arr) {
    let ergebnis = [];
    for (let i = 0; i < arr.length; i++) {
        ergebnis.push(arr[i] * 2);
    }
    return ergebnis;
}
```

Aufgabe 2

```
// Erstellen Sie ein Objekt 'buch' mit:
// - Einem Titel (String)
// - Einem Preis (Number)
// - Einer Methode 'getInfo', die Titel und Preis als String zurückgibt

// Beispielaufruf:
// buch.getInfo() => "Das Buch 'JavaScript' kostet 24.95€"
```

Aufgabe 2

```
let buch = {
    titel: "JavaScript",
    preis: 24.95,
    getInfo: function() {
        return "Das Buch '" + this.titel + "' kostet " + this.preis + "€";
    }
};
```

Ausblick

Grundkonzepte sind vertraut: Variablen, Kontrollstrukturen, Funktionen, Arrays, Objekte

Jede Sprache bringt ihre eigenen Besonderheiten mit

Als Nächstes: das Document Object Model (DOM)

JavaScript ist eine moderne, C-basierte Sprache und läuft client-seitig im Browser

Variablen mit let und const – dynamische Typisierung, keine Typdeklaration nötig

Bekannte Kontrollstrukturen und Schleifen aus C, zusätzlich for…of und for…in

Für Vergleiche immer === statt == verwenden (keine automatische Typkonvertierung)

Funktionen können anonym sein und als Event-Handler auf Interaktionen reagieren

Objekte und Arrays sind flexibel – Objekte sind die Grundlage für JSON