

Programmierung & JavaScript – Gesamt-Lernheft

EVA, Algorithmen, Compiler/Interpreter, Variablen, Operatoren

🧭 So benutzt du das Lernheft

- **Links** ist das Inhaltsverzeichnis. Der Punkt davor zeigt die IHK-Relevanz.
- Jeder Abschnitt hat ein **IHK-Etikett** und einen farbigen Kasten mit der Begründung. Rot = sehr wichtig, Orange/Gelb = mittel, Grau = gering.
- **Gelb unterstrichen** = Prüfungs-Schlagwort. Das solltest du erklären können.
- Knopf **„🎯 Nur IHK-wichtig“** blendet alles mit „gering“ aus. Gut zum Wiederholen kurz vor der Prüfung.
- Bei jedem Abschnitt kannst du **„gelernt“** abhaken. Der Fortschritt wird im Browser gespeichert.
- Suche oben (Taste `/`), mit Enter springst du zum nächsten Treffer. Lösungen sind eingeklappt – erst selbst überlegen!

sehr hoch **hoch** **mittel bis hoch** **mittel** **gering bis mittel** **gering**

Die IHK-Einstufungen sind eigene Einschätzungen aus den Einzel-Zusammenfassungen, keine amtlichen Angaben der IHK.

Hinweis: Die Einzel-Zusammenfassung zu JavaScript hatte keine IHK-Einschätzungen. Die Einstufungen hier sind für dieses Lernheft ergänzt.

🔥 Top-Prüfungsthemen

hoch 1 · **mittel** 1 · **gering bis mittel** 1

- **hoch** 1. Programmierung: Grundlagen · JavaScript und Programmierung: Grundlagen

IHK	Abschnitt	Warum?
hoch	1. Programmierung: Grundlagen	EVA-Prinzip, Algorithmus, PAP/Struktogramm sowie Compiler vs. Interpreter sind klassische AP1-Themen (Einschätzung, für dieses Lernheft ergänzt)
gering bis mittel	2. JavaScript: Einstieg	Skriptsprache, clientseitig, Sandbox – als Begriffe möglich; JavaScript selbst wird kaum geprüft (Einschätzung, ergänzt)

IHK	Abschnitt	Warum?
mittel	3. Variablen, Operatoren, Ein- und Ausgabe	Datentypen und Operatoren braucht man für Pseudocode-Aufgaben in AP1 (Einschätzung, ergänzt)

JavaScript und Programmierung: Grundlagen

Stand: 10.11.2026

Zusammenfassung der Unterrichtsfolien Programmierung, JavaScript 1 (Einstieg) und JavaScript 2 (Variablen, Operatoren, Ein- und Ausgabe) mit Übungsaufgaben.

1. Programmierung: Grundlagen IHK hoch

> 🎯 **IHK: hoch** EVA-Prinzip, Algorithmus, PAP/Struktogramm sowie Compiler vs. Interpreter sind klassische AP1-Themen (Einschätzung, für dieses Lernheft ergänzt)

Programmieren heißt, Anweisungen zu schreiben, die der Computer der Reihe nach abarbeitet; was nicht dasteht, passiert nicht, und das Programm muss sicher enden.

EVA-Prinzip: Jedes Programm arbeitet nach Eingabe, Verarbeitung und Ausgabe.

Schritt	Bedeutung	Beispiele
E: Eingabe	Programm nimmt Daten entgegen	Tastatur, Maus, Mikro, Scanner, Kamera
V: Verarbeitung	Daten werden im Programm verarbeitet	Rechnen, Vergleichen
A: Ausgabe	Ergebnis wird weitergegeben	Monitor, Drucker, Lautsprecher, anderes Programm

Der Weg zum Programm:

1. Problem analysieren und Anforderungen erfassen
2. Algorithmus formulieren (Lösungsweg)
3. Quellcode in einer Programmiersprache schreiben
4. Quellcode in Maschinencode übersetzen

Algorithmus: ein endlicher, vollständiger Lösungsweg („Kochrezept“). Jede Anweisung ist eindeutig, gleiche Eingaben liefern das gleiche Ergebnis, das Verfahren gilt für alle zulässigen Daten und hat ein eindeutiges Ende.

Drei Grundstrukturen:

- Sequenz: Anweisungen nacheinander

- Selektion: Anweisungen nur unter einer Bedingung
- Iteration: Anweisungen wiederholen, solange eine Bedingung gilt (Schleife)

Darstellungsformen: Pseudocode (freie Mischung aus Alltagssprache und Mathe), Programmablaufplan (PAP, DIN 66001), Struktogramm (Nassi-Shneiderman, DIN 66261, veraltet) und UML-Aktivitätsdiagramm (modern).

Compiler und Interpreter:

	Compiler	Interpreter
Ergebnis	direkt ausführbare Datei	übersetzt bei jedem Aufruf neu
Arbeitsweise	übersetzt den ganzen Code	arbeitet zeilenweise
Fehler	schon beim Kompilieren	erst bei der Ausführung, dann Abbruch
Tempo	schnell	langsamer
Entwicklung	komplex	einfach
Läuft	nur auf dem Zielgerät	braucht eine Laufzeitumgebung
Beispiele	C, C++, C#, Pascal	JavaScript, PHP, Perl, BASIC

Assembler und Hochsprache: Assembler ist maschinennah, gilt nur für eine bestimmte CPU, ist schwer lesbar, aber sehr schnell. Hochsprachen sind hardwareunabhängig und gut lesbar.

Syntax und Semantik: Die Syntax ist die Grammatik der Sprache; ein Verstoß (z. B. `,` statt `;`) ist ein Syntaxfehler und führt meist zum Abbruch. Er wird oft schon im Editor markiert. Die Semantik ist die Bedeutung; ein Semantikfehler liegt vor, wenn der Code korrekt geschrieben ist, aber nicht das Gewünschte tut. Er fällt meist erst zur Laufzeit auf.

2. JavaScript: Einstieg IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** Skriptsprache, clientseitig, Sandbox – als Begriffe möglich; JavaScript selbst wird kaum geprüft (Einschätzung, ergänzt)

JavaScript ist eine clientseitige, dynamisch typisierte Skriptsprache, die der Browser ausführt und die mit Java nichts zu tun hat.

Herkunft: 1995 bei Netscape als „LiveScript“ entstanden, 1996 folgte Microsofts JScript. Seit 1997 ist die Sprache als ECMAScript (ECMA-262) standardisiert (1. Ausgabe Juni 1997; die Folie nennt 1998); ECMAScript und JavaScript werden in der Praxis gleichbedeutend benutzt. Der große Sprung war ES6 (2015), seitdem gibt es jedes Jahr Erweiterungen.

Eigenschaften:

- Skriptsprache und damit Interpretersprache: Der Code wird erst beim Aufruf übersetzt und braucht eine Laufzeitumgebung. Im Browser ist das die JavaScript-Engine, jeder Browser hat

seine eigene.
- Clientseitig: Der Code läuft im Browser des Nutzers und ist dort sichtbar. Der Nutzer kann ihn auch blockieren.
- Typische Ziele: DOM ändern, Interaktion, Daten prüfen (Validierung).
- Läuft in einer Sandbox: kaum Zugriff auf Dateisystem, Hardware und Datenbanken.

Ausnahmen (JavaScript auf dem Server): V8 (Googles Engine), Node.js (läuft ohne Browser) und Electron (Framework für Desktop-Apps, z. B. Teams, Visual Studio Code, draw.io).

JavaScript in HTML einbinden:

Weg	Schreibweise	Hinweis
Extern	`<script src="datei.js"></script>`	Tag immer schließen; heute besser ans Ende vor `</body>`
Intern	`<script> ... </script>`	an beliebigen Stellen in `<head>` und `<body>`; alle Blöcke bilden ein Programm
Inline am Element	`<p onclick="alert('Hallo')">`	veraltet, nur zum schnellen Testen

Die Anweisungen laufen in der Reihenfolge, in der sie im Quellcode stehen.

Schreibweise:

- Jede Anweisung steht in einer eigenen Zeile und endet mit `;`
- JavaScript ist case sensitive: `Name` ist nicht `name`
- Leerzeichen und Zeilenumbrüche (Whitespace) werden ignoriert
- Kommentare: `// einzeilig` und `/* mehrzeilig */`

Browser-Werkzeuge: Mit F12 öffnest du die Entwicklerwerkzeuge. In der Konsole kannst du Code ausprobieren, ohne eine Webseite zu bauen. Fehler im JavaScript-Code werden nur dort angezeigt.

3. Variablen, Operatoren, Ein- und Ausgabe IHK mittel

> 🎯 **IHK: mittel** Datentypen und Operatoren braucht man für Pseudocode-Aufgaben in AP1 (Einschätzung, ergänzt)

Eine Variable ist ein benanntes Kästchen, in dem ein Wert liegt; der Wert kann sich im Programm ändern, bei einer Konstanten nicht.

Variablen und Konstanten:

```
let antwort;            // deklariert, noch leer (undefined)
let antwort2 = 42;      // deklariert und Wert zugewiesen
```

```
antwort2 = antwort2 + 3; // neuer Wert, ohne let
const player1 = "Peter"; // fester Wert
player1 = "Petra";       // Fehler
```

`let` steht nur bei der Deklaration. Eine Konstante braucht immer einen Wert bei der Deklaration.

Namensregeln: keine Leerzeichen, Satzzeichen oder führende Ziffer, kein reserviertes Wort wie `let`, und mit `let` darf ein Name nicht zweimal deklariert werden. Empfehlung: nur a bis Z, 0 bis 9 und `_`, mit sprechenden Namen.

Code	Richtig?	Grund
`let var1 = "Test";`	ja	
`let 1variable = "Test";`	nein	führende Ziffer
`let let = "Test";`	nein	reserviertes Schlüsselwort
`let var Eins = "Test";`	nein	Leerzeichen
`let var!Eins = "Test";`	nein	Satzzeichen

Scope: Eine mit `let` oder `const` in einem Block `{ }` deklarierte Variable ist nur in diesem Block sichtbar. Der Zugriff von außerhalb führt zu einer Fehlermeldung.

Datentypen in JavaScript: JavaScript ist dynamisch (schwach) typisiert, der Typ ergibt sich automatisch und kann sich im Ablauf ändern. Starke Typisierung (z. B. Java: `int zahl = 15;`) ist strenger, findet Fehler früher und übersetzt kompakter.

Typ	Inhalt
Number	Zahlen, ganz und mit Komma (Punkt: `3.14`)
String	Text in `"..."` oder `'...'`
Boolean	`true` oder `false`
Object	eigene Datentypen
undefined	Variable ohne Wert
null	„positives Nichts“, z. B. Klick auf Abbrechen im Dialog

Operatoren:

Operator	Bedeutung	Beispiel
`+ - * /`	Grundrechenarten	`7 + 3`
`%`	Modulo, Rest einer Division	`7 % 3` ergibt 1

Operator	Bedeutung	Beispiel
`**`	Potenz	`5 ** 4`
`++` `--`	um 1 erhöhen oder senken	`counter++`
`+=` `-=` `*=` `/=` `%=` `**=`	Kurzformen	`a += 5` heißt `a = a + 5`

Reihenfolge: Klammern zuerst, dann Punkt vor Strich, bei gleicher Stufe von links nach rechts. Beispiel: `7 + 5 * 3 - 1` ergibt 21, `(7 + 5) * (3 - 1)` ergibt 24.

Number und String mischen:

Ausdruck	Ergebnis	Grund
`"test" + 7`	`"test7"`	`+` verbindet Texte
`"5" + "7"`	`"57"`	zwei Strings werden verknüpft
`"5" * 3`	`15`	der String wird als Zahl gelesen
`"test" * 3`	`NaN`	nicht als Zahl lesbar (Not a Number)
`5 - 4 + "a"`	`"1a"`	von links nach rechts: erst 1, dann Text

String in Number umwandeln: `Number("123")` oder `"456" * 1`.

Ausgabe:

- `alert(wert)`: Dialogfenster
- `console.log(wert)`: Konsole; mehrere Werte mit Komma trennen
- `document.write(wert)`: in die Seite schreiben (veraltet, aber zum Üben okay)
- Anführungszeichen im String: innen die jeweils andere Sorte oder mit Backslash maskieren: `"Sie sagte: \"Hallo\""`

Template Strings: in Backticks, Werte in `${...}`:

```
let erg = `${a} + ${b} = ${a + b}`;
```

Eingabe: `prompt("Wie heißen Sie?")` liefert immer einen String. Zum Rechnen muss das Ergebnis erst in eine Zahl umgewandelt werden.

4. Aufgaben

Programmierung und JavaScript-Einstieg (ca. 10 Min.)

1. Ordne E, V und A zu: Du tippst eine Zahl, das Programm verdoppelt sie, der Monitor zeigt das Ergebnis.
2. Nenne die drei Grundstrukturen und gib für jede ein Alltagsbeispiel.

3. Ist JavaScript eine Compiler- oder Interpretersprache? Was heißt das für Fehler beim Testen?
4. Syntax- oder Semantikfehler? Du willst `3 + 8` berechnen und schreibst `3 - 8`. Und was ist es bei `3 + + 8;`?
5. Spiel den Pseudocode für den größten gemeinsamen Teiler mit m = 12 und n = 8 durch und schreib bei jedem Durchlauf r, m und n auf (`%` ist der Rest der Division).
6. Wofür steht ECMAScript, und stimmt der Satz „JavaScript ist eine Variante von Java“?
7. Warum kann JavaScript im Browser nicht einfach Dateien von der Festplatte lesen?
8. Nenne die drei Wege, JavaScript in HTML einzubinden. Welcher ist heute empfohlen, und wohin kommt das Script?
9. Finde die Fehler: `<script src="test.js">` und danach `Alert('Hi')`.
10. Praxis: Tipp in der Konsole (F12) `console.log("Hallo, Welt!");` und danach `Console.log("Hallo");`. Was passiert jeweils?

Variablen, Operatoren, Ein- und Ausgabe (ca. 15 Min.)

11. Richtig oder falsch? a) `let 2zahl = 5;` b) `let mein Name = "Max";` c) `const pi;` d) `let x = 1; let x = 2;`
12. Berechne von Hand: a) `7 + 5 * 3 - 1` b) `(7 + 5) * (3 - 1)` c) `17 % 5` d) `2 ** 3`
13. Welchen Wert hat `a` am Ende? `let a = 10; a += 5; a *= 2; a -= 4;`
14. Ergebnis und Datentyp: a) `"5" + "7"` b) `"5" * 3` c) `"hallo" * 3` d) `5 - 4 + "a"`
15. Welchen Datentyp hat `alter` nach `const alter = prompt("Wie alt bist du?");` bei Eingabe von 30, und wie wird daraus eine Zahl?
16. Schreib 3 bis 4 Zeilen Code, die zwei Zahlen speichern und die Summe als `5 + 7 = 12` ausgeben, einmal mit `+` und einmal mit Template String.
17. Praxis: Probiere `let a = 5; let b = 2; console.log(a % b, a ** b, a / b);` in der Konsole, danach `console.log("5" + 2);` und `console.log("5" - 2);`. Was fällt dir auf?
18. Bonus: Warum ergibt `"5" * 3` die Zahl 15, aber `"5" + 3` den Text `"53"`?

Zusammengeführt aus 1 Einzel-Zusammenfassung(en) · Stand 25.11.2026