

1 Ein Sprung ins kalte JavaScript-Wasser

Zeit für nasse Füße



Na komm schon rein, das Wasser ist großartig! Wir werfen gleich einen ersten Blick auf JavaScript, schreiben etwas Code, lassen ihn laufen und sehen, wie er mit Ihrem Browser interagiert! In kürzester Zeit schreiben Sie Ihren eigenen Code.

JavaScript verleiht Superkräfte. Es ist die Programmiersprache des Webs.

Mit JavaScript bekommen Ihre Webseiten nämlich **Verhalten**. Das heißt: keine trockenen, langweiligen statischen Seiten mehr, die einfach nur dasitzen und Sie anstarren. Mit JavaScript erreichen Sie Ihre Benutzer direkt. Sie können auf interessante Events reagieren, Daten aus dem Web einbinden, Diagramme direkt in Ihren Seiten erstellen und vieles mehr. Haben Sie JavaScript einmal verstanden, können Sie vollkommen neue Verhaltensweisen für Ihre Benutzer erstellen.

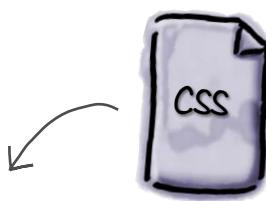
Dabei sind Sie in guter Gesellschaft. JavaScript ist nämlich nicht nur eine der **beliebtesten** Programmiersprachen, sondern wird auch von allen modernen (und den meisten älteren) Browsern **unterstützt**. Zusätzlich kommt JavaScript in vielen Umgebungen außerhalb des Browsers zum Einsatz. Dazu später mehr. Jetzt wollen wir erst mal loslegen!

Wie JavaScript funktioniert

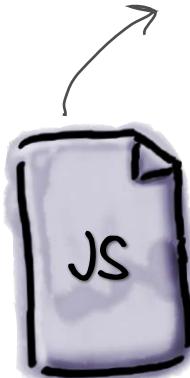
Wenn Sie bereits wissen, wie man Webseiten mit Struktur, Layout und Stilen versieht, ist der nächste logische Schritt eigentlich, etwas Verhalten hinzuzufügen, oder? Heutzutage müssen Webseiten nicht mehr statisch auf dem Bildschirm *herumhängen*. Gute Seiten sollten dynamisch und interaktiv sein, und für Ihre Benutzer sollte die Arbeit mit ihnen zu einer ganz neuen Erfahrung werden. Genau da kommt JavaScript ins Spiel. Wir wollen mal sehen, wie sich JavaScript in das *Ökosystem einer Webseite* einfügt.



Wie Sie bereits wissen, verwenden wir HTML (Hypertext Markup Language), um **Seiteninhalte** wie Absätze, Überschriften und Abschnitte zu **strukturieren**.



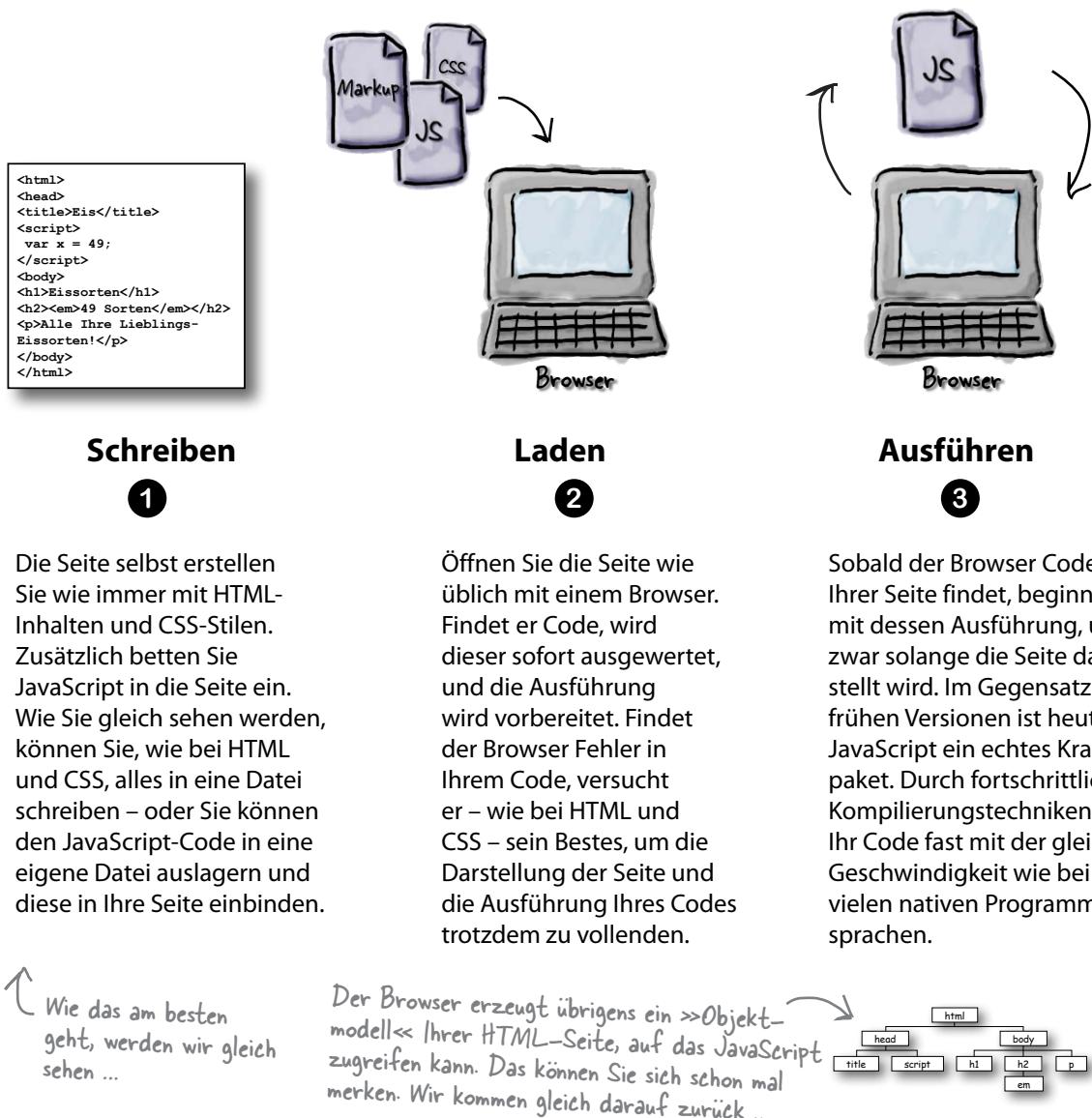
Sie wissen außerdem, dass wir CSS (Cascading Style Sheets) einsetzen, um die Darstellung der HTML-Elemente zu steuern, also welche Farben, Schriften, Rahmen, Abstände und welches Layout benutzt wird. CSS versieht Webseiten mit **Stil**, wobei Gestaltung und Struktur der Seite(n) separat voneinander definiert werden.



Und damit wollen wir Ihnen JavaScript vorstellen, den Cousin von HTML und CSS (was die Programmierung angeht), der für das **Verhalten** Ihrer Webseiten sorgt. Brauchen Sie eine bestimmte Reaktion, wenn der Besucher auf den »Sonderaktion«-Button klickt? Sollen Benutzereingaben vor dem Abschicken überprüft werden? Wollen Sie Twitter-Meldungen einbinden? Dann sollten Sie sich JavaScript ansehen. Mit JavaScript können Sie Ihre Seiten mit Programmierung versehen, wodurch die Seite Dinge berechnen, reagieren, zeichnen, kommunizieren, Meldungen ausgeben, verändern, aktualisieren und vieles mehr kann ... – eigentlich alles, was irgendwie dynamisch ist – das ist JavaScript in Aktion.

Wie Sie JavaScript schreiben werden

In der Welt der Programmierung ist JavaScript ziemlich einmalig. Mit einer typischen Programmiersprache schreiben Sie den Code und kompilieren ihn, um das Programm anschließend auszuführen. JavaScript ist wesentlich flüssiger und flexibler. Sie können JavaScript-Code direkt in eine Seite einbetten, sie in den Browser laden – fertig. Danach kann der Browser Ihren Code direkt ausführen. Wie das funktioniert, wollen wir uns jetzt etwas genauer ansehen:



JavaScript in die Seite einbinden

Die wichtigen Dinge zuerst. JavaScript bringt Ihnen nur etwas, wenn Sie es in Ihre Seiten einbinden können. Aber wie? Natürlich mit dem `<script>`-Element!

Wir beginnen mit einer einfachen Webseite, die wir über das `<script>`-Element mit Verhalten ausstatten. Denken Sie noch nicht zu viel darüber nach, was in das `<script>`-Element hineingehört – im Moment geht es einfach nur darum, etwas JavaScript ans Laufen zu bringen.

```
<!doctype html>
<html lang="de">
<head>
<meta charset="utf-8">
<title>Eine einfache Seite</title>
<script>
setTimeout(wakeUpUser, 5000);
function wakeUpUser() {
    alert("Wie lange wollen Sie diese langweilige Seite noch anstarren?");
}
</script>
</head>
<body>
<h1>Eine einfache Überschrift</h1>
<p>Hier gibt es nicht viel zu lesen. Ich bin der obligatorische Absatz, der als Beispiel in einem JavaScript-Buch lebt.
Ich bin auf der Suche nach etwas, das mein Leben etwas interessanter macht.</p>
</body>
</html>
```

Der Standard-doctype für HTML5 sowie die Elemente `<html>`- und `<head>`.

Und hier ein schlichter `<body>` für die Seite.

Aha! Wir haben das `script`-Element in den `<head>`-Teil der Seite eingebaut.

Und darin steht etwas JavaScript-Code.

Denken Sie auch hier nicht zu sehr über die Funktion des Codes nach. Wir sind uns aber sicher, dass Sie genauer wissen wollen, was die einzelnen Teile tun.

Probefahrt

Tippen Sie diesen Code ab und erstellen Sie daraus eine Datei namens »behavior.html«. Ziehen Sie die Datei auf Ihren Browser (oder benutzen Sie den Befehl Datei > Datei öffnen...), um sie zu laden. Was passiert? Tipp: Warten Sie fünf Sekunden ...





Immer mit der Ruhe! Wir erwarten hier nicht, dass Sie JavaScript bereits wie Ihre Muttersprache lesen können. Wir wollen nur, dass Sie ein Gefühl dafür bekommen.

Ganz sind Sie allerdings nicht aus dem Schneider, weil wir Ihr Hirn schon mal ein bisschen wärmlaufen lassen wollen. Erinnern Sie sich an die vorige Seite? Werfen Sie mal einen Blick auf den Code, um ein Gefühl für dessen Funktionsweise zu bekommen:

Ein Weg, wieder-
verwendbaren Code
mit dem Namen
>>wakUpUser<< zu
schreiben?

```
setTimeOut(wakeUpUser, 5000);  
function wakeUpUser() {  
    alert("Wie lange wollen Sie diese langweilige Seite noch anstarren?");  
}
```

Vielleicht eine Möglichkeit, 5 Sekunden zu zählen.
Tipp: 1000 Millisekunden = 1 Sekunde.

Ziemlich sicher die Möglichkeit, eine
Meldung an den Benutzer auszugeben.

Es gibt keine Dummenden Fragen

F: JavaScript soll angeblich nicht besonders schnell sein. Stimmt das?

A: Anfangs war JavaScript tatsächlich kein Kraftprotz. Seine Bedeutung im Web hat inzwischen aber enorm zugenommen. Als Resultat wurden viele Ressourcen (und der Gehirnschmalz einiger der besten Köpfe der Szene) investiert, um die Performance von JavaScript quasi mit einem Turbolader zu versehen. Aber wissen Sie, was? Selbst davor war JavaScript schon superschnell. Es war immer schon eine tolle Sprache, und gleich werden Sie sehen, zu welchen Großtaten JavaScript so fähig ist.

F: Ist JavaScript mit Java verwandt?

A: Nur namentlich. JavaScript wurde entwickelt, als Java gerade eine richtig heiße Phase hatte. Die Erfinder von JavaScript wollten diese Beliebtheit anzapfen, indem sie sich einen ähnlichen Namen ausdachten. Beide Sprachen borgen sich einen Teil ihrer Syntax von Programmiersprachen wie C. Davon abgesehen, sind sie aber ziemlich unterschiedlich.

F: Ist JavaScript der beste Weg, dynamische Webseiten zu erstellen? Was ist mit Flash?

A: Es gab eine Zeit, zu der Flash die bevorzugte Lösung zur Erstellung interaktiver und dynamischer Seiten war. Mittlerweile bewegt sich die Branche aber deutlich in Richtung HTML5 und JavaScript. Und mit HTML5 ist JavaScript die offizielle Standardskriptsprache für das Web. Viele Ressourcen werden verwendet, um JavaScript schneller zu machen und die Browserfunktionalität über JavaScript-APIs zu erweitern.

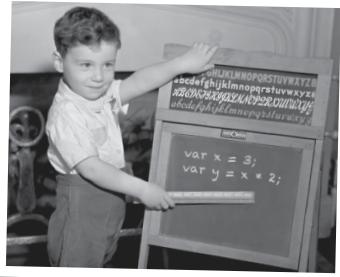
F: Mein Kumpel verwendet JavaScript innerhalb von Photoshop. Zumindest behauptet er das. Geht das wirklich?

A: Ja, JavaScript löst sich vom Browser und wird als Skriptsprache für viele Applikationen eingesetzt. Hierzu gehören Grafik- und Musikprogramme genauso wie die Programmierung auf Serverseite. In Zukunft werden wir JavaScript wohl in vielen Bereichen finden, die weit über die Webprogrammierung hinausgehen.

F: Sie haben gesagt, andere Sprachen werden kompiliert. Was bedeutet das genau, und warum ist das bei JavaScript anders?

A: Bei der Kompilierung wird der Code in eine für Maschinen effiziente Form übersetzt, die meist für die Ausführung optimiert ist. Skriptsprachen werden dagegen üblicherweise interpretiert. Das heißt, der Browser führt den Code aus, sobald er ihn findet. Skriptsprachen legen weniger Gewicht auf die Laufzeitperformance und sind eher für Aufgaben wie Prototyping, interaktive Programmierung und Flexibilität ausgelegt. Bei frühen Versionen von JavaScript war das jedenfalls so, deshalb war die Performance von JavaScript über viele Jahre auch nur mäßig. Es gibt allerdings einen Mittelweg: Eine interpretierte Sprache kann ohne Zwischenschritte kompiliert werden, und genau diese Methode setzen Browserhersteller heutzutage ein. Mit JavaScript haben Sie heute also den Komfort einer Skriptsprache, gepaart mit der Leistungsfähigkeit einer kompilierten Sprache. Wir werden in diesem Buch übrigens die Begriffe *interpretieren*, *auswerten* und *ausführen* verwenden. Je nach Kontext können sie unterschiedliche Bedeutungen haben. Für unsere Zwecke bedeuten sie aber fast immer das Gleiche.

JavaScript, du hast einen langen Weg hinter dir ...



JavaScript 1.0

Netscape ist möglicherweise älter als Sie, aber es war der erste *echte* Browserhersteller. Mitte der 1990er-Jahre war die Konkurrenz hart, speziell mit Microsoft. Neue, spannende Features hatten daher Priorität.

Dafür wollte Netscape eine Skriptsprache entwickeln, mit der jeder seine Seiten mit Skripten versehen konnte. Und so erblickte LiveScript das Licht der Welt, eine Sprache, die dieses Ziel möglichst schnell erreichen sollte. Sollten Sie noch nie von dieser Sprache gehört haben, liegt das vermutlich daran, dass Sun Microsystems ungefähr zur gleichen Zeit Java vorstellte, was zur Folge hatte, dass der Kurs der Sun-Aktie schwindelerregende Höhen erreichte. Warum sollte man sich diesen Erfolg nicht zunutze machen und LiveScript in JavaScript umbenennen? Und wen stört es schon, dass beide Sprachen eigentlich nichts miteinander zu tun haben?

Hatten wir Microsoft schon erwähnt? Kurz nach Netscape entwickelten sie ihre eigene Skriptsprache, die sie – nun ja – JScript nannten und die seltsamerweise sehr große Ähnlichkeit mit JavaScript hatte. Und so begannen die Browserkriege.



JavaScript 1.3

Zwischen 1996 und 2000 wuchs JavaScript heran. Netscape reichte die Sprache zur Standardisierung ein, und so wurde ECMAScript geboren. Kennen Sie nicht? Nicht so schlimm: jetzt schon. ECMAScript dient als Standardsprachdefinition für alle JavaScript-Implementierungen (im Browser und außerhalb).

Während dieser Zeit litten viele Entwickler unter den Folgen der Browserkriege (wegen der vielen Unterschiede in den Browsern), obwohl die Verwendung von JavaScript immer beliebter wurde. Und während subtile Unterschiede zwischen JScript und JavaScript den Entwicklern immer noch Kopfschmerzen machten, wurden die zwei Sprachen einander mit der Zeit immer ähnlicher,

Noch immer war JavaScript seinem Ruf als Amateursprache nicht entkommen – aber das sollte sich bald ändern ...



JavaScript 1.8.5

Endlich wird JavaScript erwachsen und verdient sich den Respekt der professionellen Entwickler. Vielleicht glauben Sie, dass hierfür hauptsächlich ein solider Standard wie ECMAScript 5 verantwortlich ist, der inzwischen von allen modernen Browsern implementiert wird. Tatsächlich war es aber Google, das die Verwendung von JavaScript ins professionelle Rampenlicht geholt hat. Mit der Veröffentlichung von Google Maps im Jahr 2005 zeigten sie der Welt, zu welchen Leistungen JavaScript bei dynamischen Webseiten tatsächlich in der Lage war.

Durch die große Aufmerksamkeit arbeiteten nun einige der fähigsten Köpfe der Programmiersprache daran, die JavaScript-Interpreter zu verbessern und so die Laufzeitperformance deutlich zu erhöhen. JavaScript hat sich mit wenigen Änderungen bis heute gehalten, und trotz seiner überstürzten Geburt hat es sich zu einer mächtigen und ausdrucksstarken Sprache entwickelt.

1995

2000

2012



Spitzen Sie Ihren Bleistift

Sehen Sie, wie einfach es ist, JavaScript zu schreiben

```
var price = 28.99;
var discount = 10;
var total =
    price - (price * (discount / 100));
if (total > 25) {
    freeShipping();
}

var count = 10;
while (count > 0) {
    juggle();
    count = count - 1;
}

var dog = {name: "Rover", weight: 35};
if (dog.weight > 30) {
    alert("WAU WAU");
} else {
    alert("wau wau");
}

var circleRadius = 20;
var circleArea =
    Math.PI * (circleRadius * circleRadius);
```

Auch wenn Sie JavaScript noch nicht kennen, sind wir sicher, dass Sie bereits ein paar aussichtsreiche Vermutungen über seine Funktionsweise anstellen können. Sehen Sie sich die unten stehenden Zeilen an und überlegen Sie, was der Code macht. Schreiben Sie Ihre Antworten in die Kästen daneben. Das erste Feld haben wir bereits für Sie ausgefüllt. Wenn Sie nicht weiterkommen, schauen Sie sich die Antworten auf der folgenden Seite an.

Erstelle eine Variable namens price und weise ihr den Wert 28.99 zu.



Spitzen Sie Ihren Bleistift

LÖSUNG

Sehen Sie, wie einfach es ist, JavaScript zu schreiben

```

var price = 28.99;
var discount = 10;
var total =
    price - (price * (discount / 100));
if (total > 25) {
    freeShipping();
}

var count = 10;
while (count > 0) {
    juggle();
    count = count - 1;
}

var dog = {name: "Rover", weight: 35};
if (dog.weight > 30) {
    alert("WAU WAU");
} else {
    alert("wau wau");
}

var circleRadius = 20;
var circleArea =
    Math.PI * (circleRadius * circleRadius);

```

Auch wenn Sie JavaScript noch nicht kennen, sind wir sicher, dass Sie bereits ein paar aussichtsreiche Vermutungen über seine Funktionsweise anstellen können. Sehen Sie sich die unten stehenden Zeilen an und überlegen Sie, was der Code macht. Schreiben Sie Ihre Antworten in die Kästen daneben. Das erste Feld haben wir bereits für Sie ausgefüllt. Hier sind die Antworten.

Erstelle eine Variable namens `price` und weise ihr den Wert `28.99` zu.

Erstelle eine Variable namens `discount` und weise ihr den Wert `10` zu.

Berechne durch Anwendung des Rabatts (`discount`) einen neuen Preis und weise ihn der Variablen `total` zu.

Vergleiche den Wert der Variablen `total` mit `25`. Wenn der Wert größer ist ...

... dann mache etwas mit `freeShipping`.

Ende der `if`-Anweisung.

Erstelle eine neue Variable namens `count` und weise ihr den Wert `10` zu.

Solange der Wert von `count` größer ist als `0` ...

... jongliere ein bisschen damit herum und ...

... verringere den Wert von `count` jedes Mal um `1`.

Ende der `while`-Schleife.

Erstelle einen Hund mit einem Namen und einem Gewicht (`weight`).

Ist das Gewicht des Hundes größer als `30` ...

... gib auf der Webseite die Meldung „WAU WAU“ aus.

Ansonsten ...

... gib auf der Webseite die Meldung „wau wau“ aus.

Ende der `if/else`-Anweisung.

Erstelle die Variable `circleRadius` und weise ihr den Wert `20` zu.

Erstelle die Variable `circleArea` ...

... und weise ihr das Ergebnis dieses Ausdrucks zu (1256.6370614359173).



Es ist wahr.

Mit HTML und CSS können Sie ziemlich schicke Seiten erstellen. Aber wenn Sie JavaScript kennen, bekommen Ihre Seiten eine völlig neue Dimension.

So gesehen, könnten Sie Ihre Seiten sogar als Applikationen (oder »Erfahrungen«) betrachten.

Jetzt sagen Sie vielleicht: »Klar, weiß ich. Warum sollte ich wohl sonst dieses Buch lesen?!«. Eigentlich wollten wir diese Gelegenheit nutzen, um ein wenig über das Lernen von JavaScript zu plaudern. Wenn Sie bereits eine Programmier- oder Skriptsprache beherrschen, können Sie sich vielleicht denken, was noch kommt. Haben Sie bisher allerdings hauptsächlich mit HTML und CSS gearbeitet, sollten Sie wissen, dass es beim Lernen einer Programmiersprache ein paar wesentliche Unterschiede gibt.

Bei HTML und CSS gehen Sie meistens deklarativ vor. Sie legen beispielsweise fest, dass bestimmter Text als Absatz betrachtet werden soll oder dass Elemente in der Klasse »sale« rot dargestellt werden sollen. Mit JavaScript versehen Sie die Seite dagegen mit *Verhalten*. Die hierfür nötigen Berechnungen müssen beschrieben werden. Sie müssen Dinge beschreiben können wie: »Berechne den Punktestand des Benutzers durch Addition aller korrekten Antworten!« oder: »Führe diese Aktion zehnmal durch!« oder: »Immer wenn ein Benutzer diesen Button anklickt, soll der Sie-haben-gewonnen-Sound abgespielt werden!« oder sogar: »Hole meine letzten Tweets und binde sie in diese Seite ein.«

Damit das funktioniert, brauchen Sie eine Sprache, die sich deutlich von HTML und CSS unterscheidet. Und das wollen wir uns jetzt mal ansehen ...

Und Ihr Honorar möglicherweise auch!

Anweisungen definieren

Wenn Sie HTML schreiben, versehen Sie den Text üblicherweise mit **Markup**, um ihm eine Struktur zu geben. Hierfür wird der Text mit Elementen, Attributen und Werten angereichert:

```
<h1 class="drink">Mocha Caffe Latte</h1>
<p>Espresso, Milchschaum und Schokosirup, ganz
nach Ihrem Geschmack.</p>
```

Mit HTML wird der Text strukturiert, etwa: >>Ich brauche eine große Überschrift mit dem Inhalt >Mocha Caffe Latte<<. Die Überschrift bezieht sich auf ein Getränk (drink). Danach soll ein Absatz folgen.

CSS ist ein bisschen anders. Mit CSS definieren Sie **Regeln**. Hierbei wählt jede Regel verschiedene Seitelemente aus und weist diesen bestimmte Stile zu:

```
h1.drink {
    color: brown;
}
p {
    font-family: sans-serif;
```

Mit CSS schreiben wir Regeln, die Selektoren wie h1.drink und p verwenden, um festzulegen, auf welche Teile des HTML-Codes der Stil angewandt werden soll.
Alle Überschriften vom Typ >>drink<< sollen braun geschrieben werden ...
... und alle Absätze sollen in einer serifenlosen Schrift dargestellt werden.

Mit JavaScript schreiben Sie **Anweisungen**. Jede Anweisung ist ein kleiner Teil einer Berechnung. Gemeinsam definieren die Anweisungen das Verhalten der Seite:

```
var age = 25;
var name = "Owen";
if (age > 14) {
    alert("Diese Seite
    ist leider nur für Kinder!");
} else {
    alert("Willkommen, " + name + "!");
}
```

Mehrere Anweisungen. Jede Anweisung macht einen Teil der Arbeit, z. B. die Deklaration von Variablen, die bestimmte Werte enthalten sollen.
Hier erstellen wir eine Variable, die das Alter (age) 25 enthalten soll. Zudem brauchen wir eine Variable, die den Wert >>Owen<< enthält.
Oder wir treffen Entscheidungen wie: Ist das Alter des Benutzers größer als 14?
In diesem Fall informieren wir den Benutzer, dass er zu alt für diese Seite ist.
Ansonsten begrüßen wir den Besucher mit Namen, z. B. >>Willkommen, Owen!<< (Hier jedoch nicht, weil Owen schon 25 Jahre alt ist.)

Variablen und Werte

Wie Sie vielleicht schon gemerkt haben, enthalten JavaScript-Anweisungen häufig Variablen. Variablen werden zum Speichern von Werten benutzt. Was für Werte? Hier sind ein paar Beispiele:

`var winners = 2;` ← Diese Anweisung deklariert die Variable `winners` und weist ihr den numerischen Wert `2` zu.

`var name = "Duke";` ← Hier wird der Variablen `name` eine Reihe von Zeichen (ein sogenannter >>String<<) zugewiesen.

`var isEligible = false;` ← Und diese Anweisung weist der Variablen `isEligible` den Wert `false` zu. Werte, die nur wahr (`true`) oder falsch (`false`) sein können, nennen wir >>Boolesche Werte<<. ←

Ausgesprochen: >>Buhl-sche<<.

Neben Zahlen, Strings und Booleschen Werten können Variablen auch andere Werte enthalten, die wir uns ebenfalls bald ansehen. Unabhängig von ihrem Inhalt werden alle Variablen auf die gleiche Weise erzeugt. Die Deklaration einer Variablen wollen wir nun etwas näher betrachten:

Die Deklaration einer Variablen beginnt IMMER mit dem Schlüsselwort `var`. KEINE AUSNAHMEN! Auch wenn sich JavaScript nicht beklagt, wenn Sie `var` nicht benutzen. Wir erklären Ihnen später, warum ... Danach bekommt die Variable einen Namen.

`var winners = 2;` Die Zuweisung wird immer mit einem Semikolon abgeschlossen.

Optional können wir der Variablen einen Wert zuweisen, indem wir ein Gleichheitszeichen, gefolgt von einem Wert ergänzen.

Wir sagen hier optional, weil Sie eine Variable auch ohne Wert deklarieren können. Die Zuweisung können Sie später nachholen. Um eine Variable ohne Wert zu deklarieren, lassen Sie den Zuweisungsteil einfach weg, wie hier:

`var losers;` Ohne das Gleichheitszeichen und den Wert deklarieren Sie einfach nur die Variable zur späteren Verwendung.



Wie Sie sehen, werden Boolesche Werte nicht mit Anführungszeichen umgeben.



Finger weg von der Tastatur!

Jetzt wissen Sie, dass Variablen einen Namen haben und einen Wert besitzen können.

Sie kennen außerdem schon einige Dinge, die Variablen enthalten können, wie Zahlen, Strings und Boolesche Werte.

Aber wie sollen Sie die Variablen nennen? Ist jeder Name okay? Nein – aber die Regeln für gültige Variablennamen sind einfach: Befolgen Sie diese beiden Richtlinien:

- 1** Beginnen Sie die Variablen mit einem Buchstaben, einem Unterstrich oder einem Dollarzeichen.
- 2** Danach können Sie so viele Buchstaben, Ziffern, Unterstriche oder Dollarzeichen verwenden, wie Sie wollen.

Oh, und noch etwas: Wir wollen JavaScript nicht durcheinanderbringen, indem wir eines der eingebauten *Schlüsselwörter* wie **var** oder **function** oder **false** benutzen. Lassen Sie am besten einfach die Finger davon. Einige Schlüsselwörter und ihre Bedeutungen werden wir in diesem Buch noch genauer betrachten. Hier eine Liste, damit Sie einen Überblick bekommen:

break	delete	for	let	super	void
case	do	function	new	switch	while
catch	else	if	package	this	with
class	enum	implements	private	throw	yield
const	export	import	protected	true	
continue	extends	in	public	try	
debugger	false	instanceof	return	typeof	
default	finally	interface	static	var	



Es gibt keine
Dümnen Fragen

F: Was ist ein Schlüsselwort?

A: Ein Schlüsselwort ist ein für JavaScript reserviertes Wort. JavaScript verwendet diese Wörter für eigene Zwecke. Es wäre verwirrend für Sie und den Browser, wenn Sie diese Wörter als Variablennamen verwendeteten.

F: Was passiert, wenn ich ein Schlüsselwort als Teil eines Variablennamens benutze? Kann ich zum Beispiel eine Variable namens ifOnly (also eine Variable, die das Schlüsselwort if enthält) verwenden?

A: Das ist ohne Probleme möglich. Sie dürfen nur das Schlüsselwort als solches nicht benutzen. Außerdem ist es sinnvoll, klaren Code zu schreiben. Allgemein sollten Sie daher Namen wie else vermeiden, weil er leicht mit else verwechselt werden kann.

F: Unterscheidet JavaScript zwischen Groß- und Kleinschreibung? Oder sind meineVariable und MeineVariable das Gleiche?

A: Wenn Sie sich mit HTML auskennen, sind Sie Sprachen ohne diese Unterscheidung vielleicht gewöhnt. Dem Browser ist es schließlich egal, ob Sie <head> oder <HEAD> schreiben. In JavaScript ist das aber anders. Hier wird für Variablen, Schlüsselwörter, Funktionsnamen und so ziemlich alles andere zwischen Groß- und Kleinschreibung unterschieden. Seien Sie also vorsichtig bei der Entscheidung, was Sie wie schreiben.



WEBVILLE TIMES

Peinliche Fehler bei Variablennamen vermeiden!

Bei der Wahl Ihrer Variablennamen haben Sie eine Menge Flexibilität. Hier ein paar Tipps aus Webville, die Ihnen die Namenswahl etwas erleichtern sollen:

Wählen Sie »sprechende« Variablennamen.

Variablennamen wie _m, \$, r und foo haben vielleicht für Sie eine Bedeutung, werden in Webville aber meist nicht so gern geschenkt. Sie werden nicht nur leicht wieder vergessen, Ihr Code wird auch viel lessbarer, wenn Sie Ihre Variablen winkel, aktuellerDruck oder bestandenesExamen nennen.

Verwenden Sie »CamelCase«, wenn Sie zusammengesetzte

Variablennamen benutzen wollen.

Irgendwann kommen Sie an den Punkt, an dem Sie beispielsweise einen zweiköpfigen Feuer speienden Drachen beschreiben wollen. Wie? Mit CamelCase. Hierbei wird der erste Buchstabe jedes Worts (mit Ausnahme des ersten) großgeschrieben: zweikoepfigerFeuerspeiderDrache. CamelCase lässt sich leicht schreiben, ist in Webville weit verbreitet und ist flexibel genug, um passende Variablennamen zu finden. Es gibt zwar auch noch andere Methoden, CamelCase wird jedoch am häufigsten eingesetzt (nicht nur in JavaScript).

Benutzen Sie Variablen, die mit _ und \$ beginnen, nur wenn Sie einen guten

Grund dafür haben.

Mit \$ beginnende Variablennamen sind üblicherweise für JavaScript-Bibliotheken reserviert. Auch wenn manche Autoren ihre Variablen manchmal mit _ beginnen, sollten Sie das nur mit gutem Grund tun (Sie werden wissen, wann das der Fall ist).

Seien Sie vorsichtig.

Seien Sie bei der Benennung von Variablen vorsichtig. Wir werden Ihnen später noch ein paar Tipps hierzu geben. Für den Augenblick gilt: Verwenden Sie »sprechende« Variablennamen, vermeiden Sie Schlüsselwörter und nutzen Sie bei der Deklaration von Variablen immer var.



Spaß mit der Syntax

- Jede Anweisung endet mit einem Semikolon:
`x = x + 1;`
- Ein einzeiliger Kommentar beginnt mit zwei Schrägstrichen. Kommentare sind einfache Notizen von Ihnen für andere Entwickler (oder Sie selbst) zu Ihrem Code.
`// Ich bin ein Kommentar`
- Leerzeichen sind (fast immer) erlaubt.
`x = 2233;`
- Umgeben Sie Strings oder Zeichenketten mit doppelten Anführungszeichen (oder mit einfachen, beides funktioniert, aber entscheiden Sie sich).
- "Du bist klasse!"
'Und du auch!'
- Die Booleschen Werte true und false werden nicht mit Anführungszeichen umgeben.
`rockin = true;`
- Sie müssen Variablen bei der Deklaration noch keinen Wert zuweisen:
`var width;`
- Im Gegensatz zu HTML wird bei JavaScript zwischen Groß- und Kleinschreibung unterschieden. Die Variable `counter` ist etwas anderes als die Variable `Counter`.

SPIELEN Sie Browser

Unten finden Sie JavaScript-Code, der einige Fehler enthält.

Finden Sie die Fehler im Code.

Nach dieser Übung können

Sie am Ende dieses

Kapitels nachsehen,

ob Sie alle

gefunden haben.



A

```
// Nach Pointen suchen

var joke = "JavaScript kam in eine Bar ...";
var toldJoke = "false";
var $punchline =
    "Wir haben leider nur Semi-Kola"
var %entage = 20;
var result

if (toldJoke == true) {
    Alert($punchline);
} else
    alert(joke);
}
```

Machen Sie sich nicht zu viele Gedanken darüber, was dieser Code macht. Versuchen Sie einfach nur, die Fehler in Variablen und Syntax zu finden.



B

```
\\" Kinoabend
var zip code = 98104;
var joe'sFavoriteMovie = Alarm im Weltall;
var movieTicket$      =      9;

if (movieTicket$ >= 9) {
    alert("Zu teuer!");
} else {
    alert("Wir sehen heute " + joe'sFavoriteMovie);
}
```



Der richtige Ausdruck

Um sich in JavaScript richtig auszudrücken, benötigen Sie **JavaScript-Ausdrücke**. Ausdrücke werden zu Werten evaluiert (ausgewertet). Einige haben Sie in unseren Codebeispielen bereits gesehen. Nehmen Sie beispielsweise den Ausdruck in dieser Anweisung:

Hier haben wir eine JavaScript-Anweisung, die der Variablen total das Ergebnis des ausgewerteten Ausdrucks zuweist.

```
var total = price - (price * (discount / 100));
```

Hier ist unsere Variable total. Und die Zuweisung. Und das Ganze ist ein Ausdruck.

Zum Multiplizieren verwenden wir *, zum Dividieren gibt es /.

Das Ergebnis dieses Ausdrucks ist der Endpreis (total), gemindert durch den Rabatt (discount), der als Prozentsatz des Preises (price) definiert ist. Ist der Preis 10 und der Rabatt 20, so lautet das Ergebnis 8.

Sofern Sie jemals Matheunterricht hatten, Ihre Kontoauszüge lesen können oder bereits eine Steuererklärung gemacht haben, sollten Ihnen diese numerischen Ausdrücke bekannt vorkommen.

Außerdem gibt es noch String-Ausdrücke. Dies sind ein paar davon:

```
"Lieber " + "Leser" + ", "
```

Hier werden mehrere Teilstrings zur Zeichenkette >>Lieber Leser<< zusammengefasst.

```
"super" + "kali" + youKnowTheRest
```

Genauso. Hier enthält der Ausdruck zusätzlich eine Variable. Dieser Ausdruck evaluiert zu >>superkalifragilisticexpialigeticsh<<.*

```
phoneNumber.substring(0, 3)
```

Und noch ein Beispiel, das einen String ergibt. Wie das funktioniert, werden wir später noch genauer sehen. In diesem Beispiel wird die Vorwahl einer US-Telefonnummer ermittelt.

Es gibt auch Ausdrücke, die zu **true** (wahr) oder **false** (falsch) evaluieren und als Boolesche Ausdrücke bezeichnet werden. Schauen Sie sich die Ausdrücke nacheinander an und überlegen Sie, ob das Ergebnis wahr (true) oder falsch (false) ist.

```
age < 14
```

Ist eine Person jünger als 14, ist der Ausdruck wahr, ansonsten falsch. Hiermit könnten wir beispielsweise überprüfen, ob jemand noch ein Kind ist.

```
cost >= 3.99
```

Liegen die Kosten bei 3.99 oder höher, ist der Ausdruck wahr, ansonsten falsch. Am besten kaufen Sie, wenn der Ausdruck falsch ist.

```
animal == "Bär"
```

Dieser Ausdruck ist wahr, wenn die Variable animal den String >>Bär<< enthält. In diesem Fall könnte es gefährlich werden.

Zudem können Ausdrücke auch noch zu anderen Typen evaluieren, die wir später behandeln werden. Aktuell müssen Sie nur wissen, dass diese Ausdrücke alle zu irgendetwas evaluieren: Ein Wert ist eine Zahl, ein String oder ein Boolescher Wert. Im Folgenden wollen wir sehen, was Sie davon haben.

* Hier gehen wir natürlich davon aus, dass die Variable youKnowTheRest den Wert >>fragilisticexpialigeticsh<< hat.

Ausdrucksübung



Spitzen Sie Ihren Bleistift —

Holen Sie Ihren Bleistift raus und probieren Sie ein paar Ausdrücke durch. Berechnen Sie die Werte und schreiben Sie die Antwort auf. Richtig: AUFSCHREIBEN! Vergessen Sie, dass Ihre Mutter Ihnen verboten hat, in Bücher zu schreiben, und bringen Sie Ihre Antworten gleich hier zu Papier! Und überprüfen Sie (danach!) am Ende dieses Kapitels, ob alles richtig war.

Kann man >>Fahrenheit-nach-Celsius-Rechner<<
sagen?

(9 / 5) * temp + 32

Wie lautet das Ergebnis, wenn temp den Wert 10 hat? _____

Dies ist ein Boolescher Ausdruck.
Der Operator == überprüft, ob
zwei Werte gleich sind.

color == "orange"

Ist dieser Ausdruck wahr oder falsch, wenn color den Wert »pink« hat?

name + ", " + "Sie haben gewonnen!"

Was ist, wenn er den Wert »orange« hat? _____

yourLevel > 5

Hier wird überprüft, ob
der erste Wert größer
ist als der zweite. Um zu
testen, ob der erste Wert
größer oder gleich dem
zweiten ist, können Sie den
Operator >= verwenden.

(level * points) + bonus

Welcher Wert wird berechnet, wenn name den Wert »Martha« hat?

color != "orange"

Wie lautet das Ergebnis, wenn yourLevel den Wert 2 hat? _____

Wie lautet das Ergebnis, wenn yourLevel den Wert 5 hat? _____

Wie lautet das Ergebnis, wenn yourLevel den Wert 7 hat? _____

1000 + "108"

Okay, level hat den Wert 5, points ist 30000 und bonus ist 3300.
Welches Ergebnis bekommen wir? _____

ZUSATZPUNKTE!

Ist dieser Ausdruck wahr (true) oder falsch (false), wenn color den Wert »pink« hat? _____



Serioses Coden

Haben Sie bemerkt, dass der Operator = für Zuweisungen

benutzt wird, während == auf Gleichheit prüft? Wir verwenden also ein Gleichheitszeichen für Zuweisungen. Zwei Gleichheitszeichen testen, ob zwei Werte tatsächlich gleich sind. Beim Programmieren passiert es schnell, dass man beide Operatoren verwechselt.



```
while (juggling) {
    keepBallsInAir();
}
```



Dinge mehr als einmal tun

Viele Dinge tut man mehr als einmal:

Einseifen, ausspülen, wiederholen ...

Auftragen und polieren ...

Alle Bonbons im Glas essen, bis keins mehr übrig ist.

Sicher müssen im Code manche Dinge mehrmals erledigt werden. JavaScript bietet verschiedene Möglichkeiten, den Code in einer Schleife zu wiederholen:

while, for, for in und forEach. Im weiteren Verlauf werden wir uns die verschiedenen Arten von Schleifen genauer ansehen. Zunächst wollen wir uns mit **while** beschäftigen. Ausdrücke wie `scoops > 0`, die zu Booleschen Werten evaluieren, kennen Sie inzwischen. Diese sind der Schlüssel für einen **while**-Ausdruck. Und das geht so:

```
while (scoops > 0) {
    document.write("Noch eine Kugel!");
    scoops = scoops - 1;
}
```

Eine while-Anweisung beginnt mit dem Schlüsselwort while.

while verwendet einen Booleschen Ausdruck, genannt Bedingungsanweisung oder einfach Bedingung.

Ist die Bedingung wahr, wird alles im Codeblock ausgeführt.

Was ist ein Codeblock? Alles zwischen den geschweiften Klammern: { das hier! }.

Ist die Bedingung nach Ausführen des Codeblocks immer noch wahr, springen wir wieder zurück und führen den Codeblock erneut aus. Ist die Bedingung falsch, sind wir fertig.

Wie gesagt: einseifen, ausspülen, wiederholen!

Wie eine while-Schleife funktioniert

Da dies Ihre erste while-Schleife ist, wollen wir uns einmal genau anschauen, was hier passiert. Wie Sie sehen, haben wir den Code um die Deklaration der Variablen scoops (Eiskugeln) erweitert und mit dem Wert 5 initialisiert.

Beginnen wir damit, den Code auszuführen. Zunächst erhält scoops den Wert 5.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



Danach kommen wir zur while-Anweisung. Beim Auswerten der while-Anweisung wird zuerst die Bedingung ausgewertet, um zu sehen, ob sie wahr oder falsch ist.

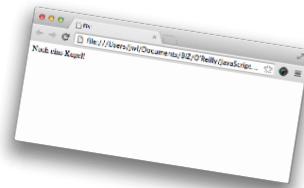
```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

Ist scoops größer als 0? Sieht so aus!



Die Bedingung ist wahr, also beginnen wir, den Codeblock auszuführen. Die erste Anweisung gibt den String »Noch eine Kugel!
« im Browser aus.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



Die nächste Anweisung subtrahiert 1 von der Anzahl der Kugeln und weist scoops den neuen Wert zu.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

1 Kugel weg,
4 übrig!



Dies ist die letzte Anweisung im Block, also springen wir wieder an den Anfang und beginnen von vorne.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

Bei der erneuten Auswertung der Bedingung hat scoops den Wert 4. Aber das ist immer noch mehr als 0.

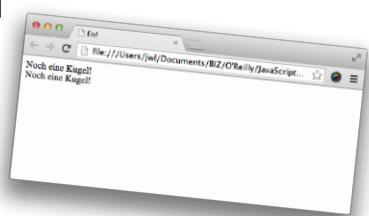
```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

Immer noch reichlich
Eis vorhanden.



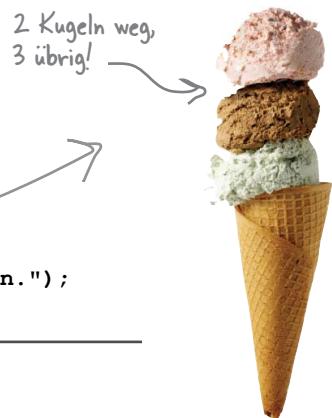
Also geben wir den String »Noch eine Kugel!
« ein weiteres Mal aus.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



Die nächste Anweisung verringert die Anzahl der Kugeln nochmals um 1 und weist `scoops` den neuen Wert zu.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



Dies ist die letzte Anweisung des Blocks, also springen wir wieder zur Bedingung und beginnen von vorne.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

Bei der erneuten Auswertung der Bedingung hat `scoops` nun den Wert 3. Das ist aber immer noch mehr als 0.

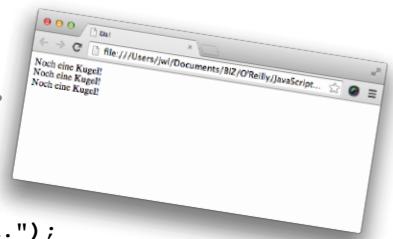
```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

Immer noch genug da!



Und wieder einmal geben wir den String »Noch eine Kugel
« im Browser aus.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



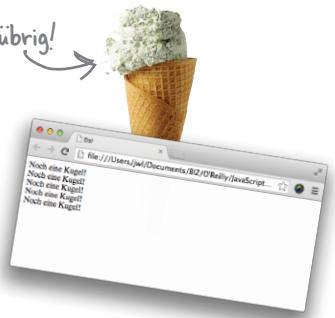
Wie Sie sehen, geht das immer so weiter ... bei jedem Schleifendurchlauf dekrementieren wir (subtrahieren 1 von) scoop, geben einen neuen String im Browser aus und beginnen wieder von vorne.

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



Und weiter ...

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```



... bis sich beim letzten Schleifendurchlauf die Voraussetzungen ändern. Jetzt hat scoop den Wert 0, und die Bedingung evaluiert zu false. Das war's dann, Leute. Die Schleife wird nicht erneut durchlaufen. Stattdessen wird der Codeblock übersprungen und die folgende Anweisung ausgeführt.

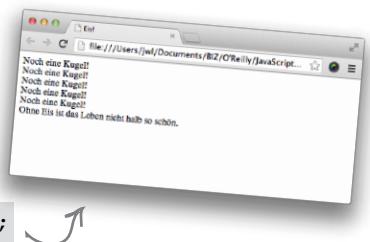
```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```

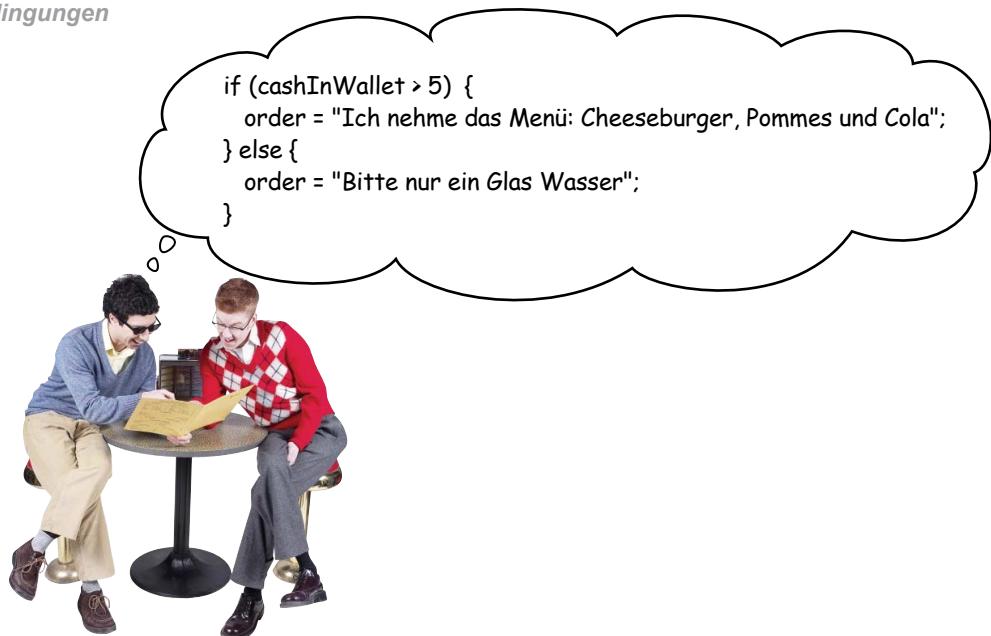
5 Kugeln weg. Nichts mehr übrig!



Jetzt wird die andere document.write-Anweisung ausgeführt, die den String »Ohne Eis ist das Leben nicht halb so schön.« ausgibt. Damit sind wir fertig!

```
var scoops = 5;
while (scoops > 0) {
    document.write("Noch eine Kugel!<br>");
    scoops = scoops - 1;
}
document.write("Ohne Eis ist das Leben nicht halb so schön.");
```





Entscheidungen in JavaScript

Gerade haben Sie gesehen, wie Sie eine `while`-Schleife mit Bedingungen steuern können. Boolesche Ausdrücke können Sie aber auch benutzen, um in einer `if`-Anweisung Entscheidungen zu treffen. Die `if`-Anweisung führt den Codeblock nur aus, wenn die Bedingung wahr ist (ihr Ergebnis `true` ist). Hier ein Beispiel:

Hier sehen Sie das Schlüsselwort `if`, gefolgt von einer Bedingung und einem Codeblock.

Diese Bedingung überprüft, ob wir weniger als 3 Kugeln übrig haben.

```

if (scoops < 3) {
    alert("Gleich ist das Eis alle!");
}

```

Ist das der Fall, wird der Codeblock der `if`-Anweisung ausgeführt.

`alert` nimmt einen String und zeigt ihn als Pop-up-Dialog in Ihrem Browser an. Probieren Sie das ruhig mal!

The screenshot shows a standard browser alert box with the title "JavaScript-Warnmeldung". The message inside reads "Essen Sie schneller. Das Eis schmilzt gleich!". There is an "OK" button at the bottom right of the dialog.

Wir können `if`-Anweisungen auch miteinander kombinieren, indem wir eine oder mehrere `else if`-Anweisungen hinzufügen.

```

if (scoops >= 5) {
    alert("Essen Sie schneller. Das Eis schmilzt gleich!");
} else if (scoops < 3) {
    alert("Gleich ist das Eis alle!");
}

```

Mit `if/else if` können wir mehrere Tests hintereinander ausführen.

Sie können beliebig viele `>>else if<<`-Anweisungen einbauen. Hierbei bekommt jeder Test seinen eigenen Codeblock, der ausgeführt wird, wenn die Bedingung wahr ist.

Und wenn Sie richtig VIELE Entscheidungen treffen müssen

Sie können beliebig viele `if/else`-Anweisungen miteinander verketten. Und falls Sie, wenn alle Bedingungen fehlschlagen, ein letztes `else` brauchen, geht auch das, nämlich so:

```

if (scoops >= 5) { ← Hier überprüfen wir, ob fünf
    alert("Essen Sie schneller. Das Eis schmilzt gleich!");
} else if (scoops == 3) { ← ... oder ob noch genau drei übrig sind ...
    alert("Gleich ist das Eis alle!");
} else if (scoops == 2) {
    alert("Zum Ersten!");
} else if (scoops == 1) {
    alert("Zum Zweiten!");
} else if (scoops == 0) {
    alert("Und fertig!");
}
else { ← ... und wenn es noch 2, 1 oder 0 sind,
    alert("Es ist noch genug Eis da. Immer hereinspaziert.");
}

```

Wenn keine der obigen Bedingungen wahr ist, wird dieser Code ausgeführt.



Es gibt keine
Dümnen Fragen

F: Was genau ist ein Codeblock?

A: Syntaktisch gesehen, ist ein Codeblock (oder einfach ein Block) eine Folge von Anweisungen, die durch geschweifte Klammern zusammengefasst werden. Die in einem Codeblock enthaltenen Anweisungen werden als Gruppe gemeinsam ausgeführt. So werden alle Anweisungen im Codeblock einer while-Schleife ausgeführt, wenn die Bedingung für das while wahr ist. Das Gleiche gilt für einen Block in einer if- oder else if-Anweisung.

F: Ich habe Code gesehen, bei dem die Bedingung kein Boolescher Wert ist, sondern eine einfache Variable, die einen String enthält. Wie funktioniert denn das?

A: Darauf werden wir später genauer eingehen. Die kurze Antwort: Wenn es darum geht, wahr und falsch zu unterscheiden, ist JavaScript ziemlich flexibel. So wird jede Variable, die einen (nicht leeren) String enthält, als wahr angesehen; eine Variable, der kein Wert zugewiesen wurde, gilt dagegen als falsch. Die Details hierzu werden Sie bald kennenlernen.

F: Sie haben gesagt, Ausdrücke könnten auch etwas anderes als Zahlen, Strings oder Boolesche Werte zum Ergebnis haben. Was denn zum Beispiel?

A: Boolesche Werte sind nach dem englischen Mathematiker George Boole benannt, der die Boolesche Logik erfunden hat. Um zu zeigen, dass die Variablen nach George benannt sind, wird dieser Begriff üblicherweise großgeschrieben.

F: Woher kommt der Name Boolescher Wert?

A: Boolesche Werte sind nach dem englischen Mathematiker George Boole benannt, der die Boolesche Logik erfunden hat. Um zu zeigen, dass die Variablen nach George benannt sind, wird dieser Begriff üblicherweise großgeschrieben.



Codemagneten

Unser JavaScript am Kühlschrank ist durcheinandergeraten. Können Sie die Magneten an die richtige Stelle schieben, sodass ein funktionierendes JavaScript entsteht, das die unten stehende Ausgabe erzeugt? Überprüfen Sie Ihre Lösung am Ende des Kapitels, bevor Sie mit der nächsten Seite weitermachen.

Bringen Sie die Magneten in die richtige Reihenfolge, um ein funktionierendes JavaScript-Programm daraus zu machen.

```
document.write("Zum Geburtstag, lieber " + name + ",<br>");
```

```
document.write("Zum Geburtstag viel Glück.<br>");
```

```
var i = 0;  
    var name = "Jørgen";  
    i = i + 1;  
}
```

```
document.write("Zum Geburtstag viel Glück.<br>");
```

```
while (i < 2) {
```

↓ Das Programm sollte diese Ausgabe erzeugen.

```
Zum Geburtstag viel Glück.  
Zum Geburtstag viel Glück.  
Zum Geburtstag, lieber Jørgen,  
Zum Geburtstag viel Glück.
```



Benutzen Sie diesen Platz, um die Magneten anzzuordnen.

Sprechen Sie Ihre Benutzer direkt an

Wir haben bereits darüber geredet, wie Sie Ihre Seiten interaktiver machen können und dass Sie dafür mit Ihren Benutzern kommunizieren müssen. Tatsächlich gibt es hierfür verschiedene Wege, von denen Sie einige bereits kennen. Bevor wir diese Möglichkeiten detailliert betrachten, wollen wir Ihnen einen kleinen Überblick geben:

Warnmeldungen

Wie Sie gesehen haben, bietet Ihnen der Browser eine einfache Möglichkeit, Benutzer mittels der `alert`-Funktion anzusprechen. Hierfür brauchen Sie `alert` nur mit einem String aufzurufen, den der Browser dann als Meldung in einer Dialogbox ausgibt. Wir müssen allerdings zugeben, dass wir diese Möglichkeit etwas zu oft benutzt haben. Sie sollten `alert` möglichst nur benutzen, wenn Sie alles anhalten wollen, um den Benutzer über etwas zu informieren.

Direkt ins Dokument schreiben

Stellen Sie sich Ihre Webseite als Dokument vor (der Browser macht das auch so). Mit der Funktion `document.write` können Sie HTML und alle möglichen anderen Inhalte an beliebiger Stelle in Ihre Seite einfügen. Das sieht man zwar gelegentlich, aber allgemein gilt diese Methode als schlechter Stil. Wir sind in diesem Kapitel kurz so vorgegangen, weil dieser Weg einen guten Startpunkt bietet.

Die Konsole benutzen

Jede JavaScript-Umgebung besitzt auch eine Konsole, die Meldungen Ihres Codes aufzeichnen kann. Um eine Nachricht in das Konsolenprotokoll zu schreiben, verwenden Sie die Funktion `console.log`. Ihr übergeben Sie den String, der in der Konsole ausgegeben werden soll (weitere Details über die Verwendung von `console.log` in Kürze). Betrachten Sie `console.log` als Werkzeug, um Fehler in Ihrem Code zu finden, das Ihre Benutzer typischerweise nie zu sehen bekommen. Als Mittel zur Kommunikation mit den Besuchern taugt es also eher nicht.

Das Dokument direkt manipulieren

Das hier ist die Oberliga. Im Prinzip können Sie direkt mit Ihrer Seite und den Benutzern interagieren, denn mithilfe von JavaScript können Sie direkt auf die eigentliche Webseite zugreifen, ihren Inhalt lesen und ändern oder sogar die zugrunde liegende Struktur und die enthaltenen Stile verändern! Das geschieht über das *Document Object Model* des Browsers (mehr dazu später). Dies ist auch die beste Möglichkeit, um mit Ihren Benutzern zu kommunizieren. Um das Document Object Model zu benutzen, müssen Sie allerdings wissen, wie Ihre Seite strukturiert ist und wie die Programmierschnittstelle benutzt wird, über die die Seite gelesen und geschrieben werden kann. Wir werden uns bald damit befassen. Zunächst gibt es aber noch mehr JavaScript zu lernen.



In diesem Kapitel
verwenden wir diese drei
Methoden.

Die Konsole ist praktisch, um
Programmierfehler aufzustöbern. Bei
Tippfehlern, z. B. einem fehlenden
Anführungszeichen, gibt JavaScript
normalerweise eine Fehlermeldung auf
der Konsole aus, um Ihnen bei der
Fehlersuche zu helfen.

Hier wollen wir hin. Sobald Sie
angekommen sind, können Sie Ihre
Seite auch vielerlei Arten lesen,
ändern und manipulieren.

WER MACHT WAS?

Alle Kommunikationsmethoden sind maskiert zur Party gekommen. Können Sie uns helfen, ihre wahre Identität aufzudecken? Verbinden Sie die Beschreibungen auf der rechten Seite mit den Namen auf der linken. Die erste Verbindung haben wir schon für Sie hergestellt.

document.write

Ich halte den Benutzer an und gebe eine kurze Meldung aus. Der Benutzer muss erst auf »OK« klicken, damit es weitergeht.

console.log

Ich füge etwas HTML und Text in die Seite ein. Ich bin zwar nicht die eleganteste Methode, um Nachrichten an Ihre Benutzer auszugeben, aber ich funktioniere in jedem Browser.

alert

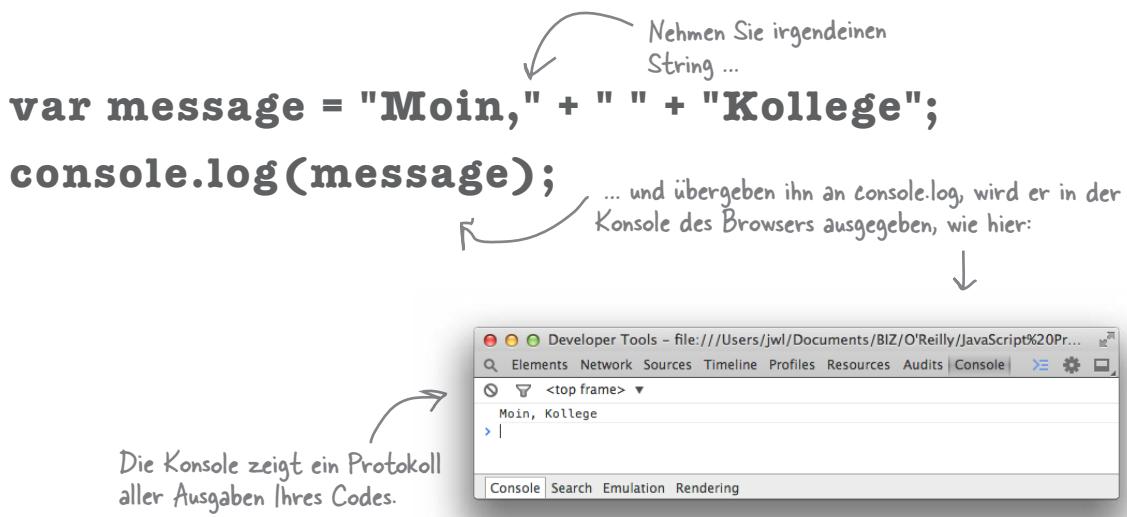
Benutzen Sie mich, um eine Webseite komplett zu kontrollieren: Ich kann Benutzereingaben auslesen, den HTML-Code und CSS-Stile ändern sowie den Inhalt Ihrer Seite aktualisieren.

**Document Object Model
(DOM)**

Ich bin eigentlich nur zur Fehlersuche da. Benutzen Sie mich, und ich werde Informationen auf einer speziellen Entwicklerkonsole ausgeben.

Ein näherer Blick auf console.log

Lassen Sie uns die Funktionsweise von `console.log` jetzt etwas genauer betrachten. Später in diesem Kapitel und auch im weiteren Verlauf dieses Buchs können wir die Konsole dann benutzen, um Ausgaben Ihres Codes zu betrachten und Programmierfehler zu beseitigen. Vergessen Sie aber nicht, dass dieses Feature nur wenigen Webbenutzern bekannt ist, sodass Sie es in der finalen Version Ihrer Webseite eher nicht benutzen sollten. Das Schreiben in die Konsole wird daher hauptsächlich benutzt, um bei der Entwicklung der Seite Fehler aufzuspüren. Davon abgesehen bietet die Konsole eine sehr gute Möglichkeit, Ihrem Code bei der Arbeit zuzusehen, während Sie die Grundlagen von JavaScript lernen.



Es gibt keine Dümnen Fragen

F: Ich habe verstanden, dass `console.log` für die Ausgabe von Strings benutzt werden kann, aber was genau ist die Konsole? Und warum sind »`console`« und »`log`« durch einen Punkt getrennt?

A: Gute Frage. Wir greifen hier ein bisschen vor. Stellen Sie sich die Konsole einfach als Objekt vor, das bestimmte Dinge tut – so Konsolendinge eben. Dazu gehört beispielsweise das Protokollieren (oder »Logging«) von Informationen. Hierfür benutzen wir die Syntax »`console.log`«, der

innerhalb der Klammern die gewünschte Ausgabe übergeben wird. Behalten Sie das schon mal im Hinterkopf – wir kommen später darauf zurück. Diese Informationen zu `console.log` sollten für Sie im Moment erst einmal reichen.

F: Kann die Konsole nur protokollieren?

A: Nein, auch wenn die meisten Leute sie nur dafür benutzen. Es gibt noch ein paar fortgeschrittenere Techniken für die Benutzung der Konsole, die sich aber

zwischen den Browsern unterscheiden können. Obwohl mittlerweile alle modernen Browser eine Konsole haben, ist diese nicht Teil einer formellen Spezifikation.

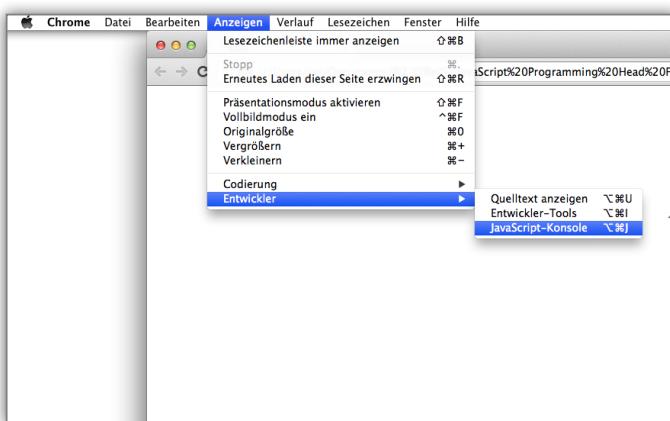
F: Das klingt ja alles ziemlich toll. Aber wo finde ich die Konsole denn überhaupt? Ich benutze sie in meinem Code, aber wo sind die Ausgaben?

A: In den meisten Browsern müssen Sie ein spezielles Konsolenfenster öffnen. Details hierzu finden Sie auf der folgenden Seite.

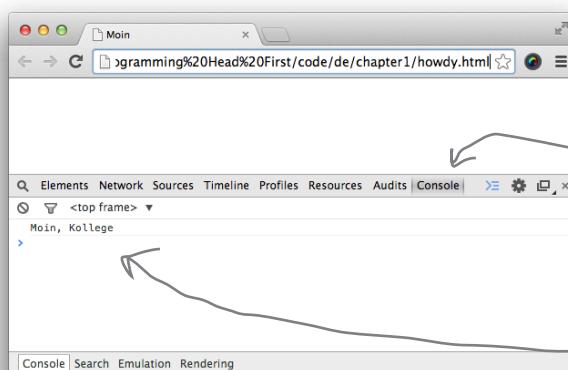
Die Konsole öffnen

Jeder Browser hat seine eigene Implementierung der Konsole, und um es noch ein bisschen schwieriger zu machen, ändert sich diese noch dazu recht häufig – zwar nicht im großen Stil, aber so, dass das hier Gezeigte möglicherweise bei Ihnen etwas anders aussieht.

Hier zeigen wir Ihnen, wie Sie mit Google Chrome (Version 34) für den Mac auf die Konsole zugreifen können. Für die anderen größeren Browser finden Sie diese Informationen unter <http://wickedlysmart.com/hfsconsole>. Sobald Sie die Konsole des Browsers verstanden haben, ist es relativ einfach, auch die anderen Browser zu nutzen. Probieren Sie das ruhig einmal – schon allein, um sich mit der Materie ein wenig vertraut zu machen.



Um auf dem Mac auf die Konsole von Chrome zuzugreifen, benutzen Sie das Menü Anzeigen > Entwickler > JavaScript-Konsole.



Standardmäßig erscheint die Konsole im unteren Teil des Browserfensters.

Stellen Sie sicher, dass die Registerkarte >Console< ausgewählt ist.

Hier werden die Nachrichten, die Sie an `console.log` übergeben, anzeigt.

Die anderen Register sind im Moment nicht so wichtig. Sie sind auch nützlich, aber wir wollen uns zunächst auf die Registerkarte Console konzentrieren, in der die `console.log`-Meldungen aus Ihrem Code angezeigt werden.

Die erste richtige JavaScript-Applikation

Jetzt wollen wir Ihre neuen JavaScript-Kenntnisse und `console.log` endlich praktisch anwenden. Wir brauchen ein paar Variablen, eine while-Schleife und eine if/else-Anweisung. Dazu noch ein bisschen Kosmetik, und schon haben wir eine seriöse Businessapplikation. Bevor Sie sich aber auf den Code stürzen, sollten Sie kurz überlegen, wie Sie den Klassiker »99 Flaschen Bier« (99 bottles of beer) programmieren würden.

```
var word = "Flaschen";
var count = 99;

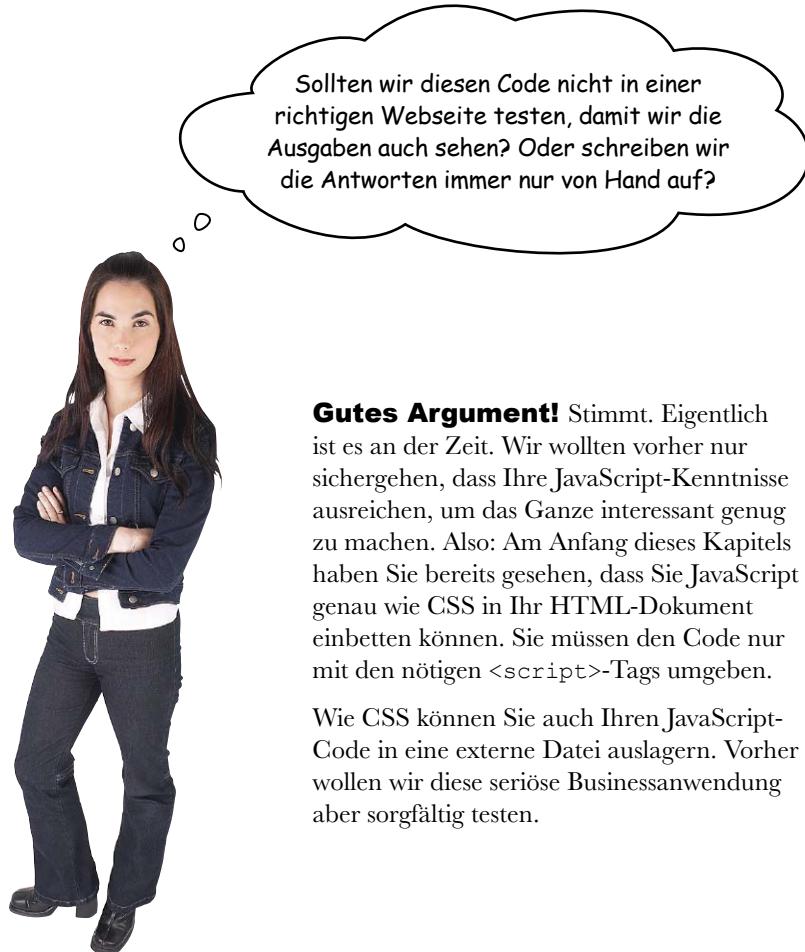
while (count > 0) {

    console.log(count + " " + word + " Bier im Regal,");
    console.log(count + " " + word + " Bier.");
    console.log("Nimm' eine runter, reich' sie herum");
    count = count - 1;
    if (count > 0) {
        console.log(count + " " + word + " Bier im Regal.");
    } else {
        console.log("Keine " + word + " Bier im Regal.");
    }
}
```



KOPF-
NUSS

Unser Code enthält einen kleinen Fehler. Das Programm läuft korrekt, aber die Ausgabe ist nicht zu 100 % perfekt. Versuchen Sie, das Problem zu finden und zu lösen.



Sollten wir diesen Code nicht in einer richtigen Webseite testen, damit wir die Ausgaben auch sehen? Oder schreiben wir die Antworten immer nur von Hand auf?

Gutes Argument! Stimmt. Eigentlich ist es an der Zeit. Wir wollten vorher nur sichergehen, dass Ihre JavaScript-Kenntnisse ausreichen, um das Ganze interessant genug zu machen. Also: Am Anfang dieses Kapitels haben Sie bereits gesehen, dass Sie JavaScript genau wie CSS in Ihr HTML-Dokument einbetten können. Sie müssen den Code nur mit den nötigen `<script>`-Tags umgeben.

Wie CSS können Sie auch Ihren JavaScript-Code in eine externe Datei auslagern. Vorher wollen wir diese seriöse Businessanwendung aber sorgfältig testen.



Okay, dann wollen wir den Code mal in den Browser laden ... Folgen Sie den unten stehenden Anweisungen und starten Sie Ihre Businessapplikation! Das Ergebnis sehen Sie unten:

↑ Den Code und die Beispieldateien für dieses Buch finden Sie unter http://examples.oreilly.de/german_examples/hfjavascriptprogrer/Beispielcode.zip.

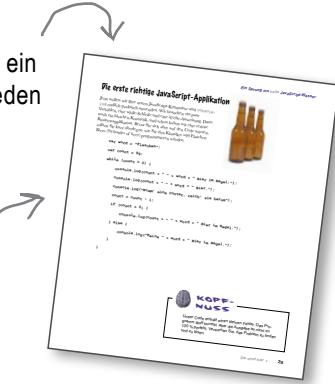
- 1 Werfen Sie einen Blick auf den unten stehenden HTML-Code: Hier kommt Ihr JavaScript hin. Tippen Sie den HTML-Code ab und packen Sie das JavaScript-Programm von den vorigen zwei Seiten zwischen die <script>-Tags. Hierfür reicht ein Editor wie NotePad (Windows) oder TextEdit (Mac). Speichern Sie die Seite auf jeden Fall als »reinen Text«. Wenn Sie lieber mit einem HTML-Editor wie Dreamweaver, Coda oder WebStorm arbeiten, können Sie natürlich auch diese benutzen.

```
<!doctype html> ← Geben Sie das hier ein.

<html lang="de">

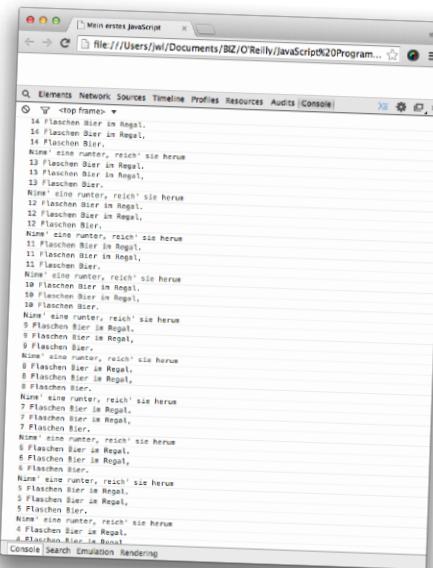
  <head>
    <meta charset="utf-8">
    <title>Mein erstes JavaScript</title>
  </head>
  <body>
    <script>
      ← Hier sind die <script>-Tags. Damit wissen Sie auch, wo Ihr Code hinkommt.

    </script>
  </body>
</html>
```



- 2 Speichern Sie die Datei als »index.html«.
- 3 Laden Sie die Datei in Ihren Browser. Hierfür können Sie die Datei entweder direkt in das Browserfenster ziehen, oder Sie benutzen die Menüoption Datei > Öffnen (bzw. Datei > Datei öffnen) im Browser Ihrer Wahl.
- 4 Auf der Webseite selbst wird nichts angezeigt, da wir alle Ausgaben per console.log auf der Konsole protokollieren. Sie müssen also die Konsole Ihres Browsers öffnen. Danach dürfen Sie sich für diese seriöse Businessapplikation selbst auf die Schulter klopfen.

Hier sind die Ausgaben Ihres Codes. Er erzeugt den kompletten Songtext für >>99 Flaschen Bier<< und gibt ihn auf der Browserkonsole aus.



Wie bekomme ich den Code in meine Seite? (... mal die Möglichkeiten zählen ...)

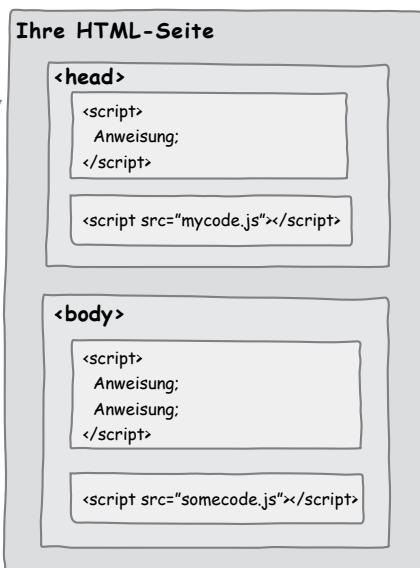
Mittlerweile wissen Sie, dass Sie das `<script>`-Element mit Ihrem JavaScript-Code im `<head>`- oder im `<body>`-Teil Ihrer Seite platzieren können. Daneben gibt es weitere Möglichkeiten, den Code einzubinden. Wir werden alle einmal ansehen (und überlegen, warum eine Methode möglicherweise besser ist als eine andere):

Sie können Ihren Code direkt in die Seite einbetten. Am häufigsten wird der Code per `<script>`-Element in den `<head>`-Teil der Seite eingefügt. Das scheint logisch und macht den Code leicht auffindbar. Trotzdem ist das nicht immer der beste Ort. Warum? Lesen Sie weiter ...

Oder Sie können den Code im body-Teil des Dokuments platzieren.

Umgeben Sie Ihren Code hierfür mit `<script>`-Tags und fügen Sie ihn im `<body>` der Seite (typischerweise direkt vor dem schließenden `</body>`-Tag) ein.

Das ist schon etwas besser. Warum? Wenn der Browser Ihre Seite lädt, holt er sich zunächst alle im `<head>`-Teil referenzierten Daten – bevor die Informationen aus dem `<body>`-Teil abgerufen werden. Befindet sich Ihr Code im `<head>`, müssen die Benutzer möglicherweise warten, bis die Seite dargestellt wird. Wird der Code dagegen erst nach dem HTML im `<body>` der Seite geladen, können die Benutzer den Seiteninhalt schon sehen, während der Code noch geladen wird. Es gibt aber eine noch bessere Methode. Lesen Sie weiter ...



Oder Sie schreiben den Code in eine eigene Datei und binden diese im `<head>` der Seite des Dokuments ein. Das funktioniert genau wie das Einbinden einer CSS-Datei. Der einzige Unterschied ist, dass Sie hier das `src`-Attribut des `<script>`-Tags benutzen, um die URL Ihrer JavaScript-Datei anzugeben.

Befindet sich Ihr Code in einer externen Datei, ist er leichter (unabhängig vom HTML-Code) zu pflegen und kann in mehrere Seiten eingebunden werden. Der Nachteil ist, dass der Code hierfür vor dem `<body>` der Seite geladen werden muss. Gibt es vielleicht noch eine bessere Methode? Lesen Sie weiter ...

Schließlich können Sie eine externe Datei auch innerhalb des `<body>`-Teils der Seite einbinden.

Damit sind beide Vorteile vereint. Einerseits haben wir eine schöne, wartbare JavaScript-Datei, die Sie in beliebigen Seiten wiederverwenden können. Andererseits wird sie erst am Ende des `<body>`-Teils eingebunden, wodurch der Inhalt schneller dargestellt werden kann. Nicht schlecht.

Trotz der erdrückenden Beweislast finde ich, dass der Code auch gut in den `<head>`-Teil passt.



Wir müssen euch leider trennen

Es tut immer weh, wenn jeder seiner Wege geht, aber es muss sein. Es ist Zeit, Ihrem JavaScript ein Zuhause in seiner eigenen Datei zu geben. Und das geht so ...

- ① Öffnen Sie die Datei index.html und wählen Sie den gesamten Code, also alles zwischen den `<script>`-Tags aus. Ihre Auswahl sollte so aussehen:

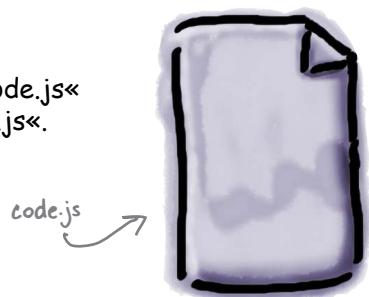
```

<!doctype html>
<html lang="de">
<head>
<meta charset="utf-8">
<title>Mein erstes JavaScript</title>
</head>
<body>
<script>
    var word = "Flaschen";
    var count = 99;
    while (count > 0) {
        console.log(count + " " + word + " Bier im Regal,");
        console.log(count + " " + word + " Bier.");
        console.log("Nimm' eine runter, reich' sie herum");
        count = count - 1;
        if (count > 0) {
            console.log(count + " " + word + " Bier im Regal.");
        } else {
            console.log("Keine " + word + " Bier im Regal.");
        }
    }
</script>
</body>
</html>

```

Wählen Sie nur den Code, aber nicht die `<script>`-Tags aus.
Dort, wo Sie hingehen, werden sie nicht mehr gebraucht.

- ② Erstellen Sie jetzt mit Ihrem Editor eine Datei namens »code.js« und fügen Sie den Code dort ein. Dann speichern Sie »code.js«.



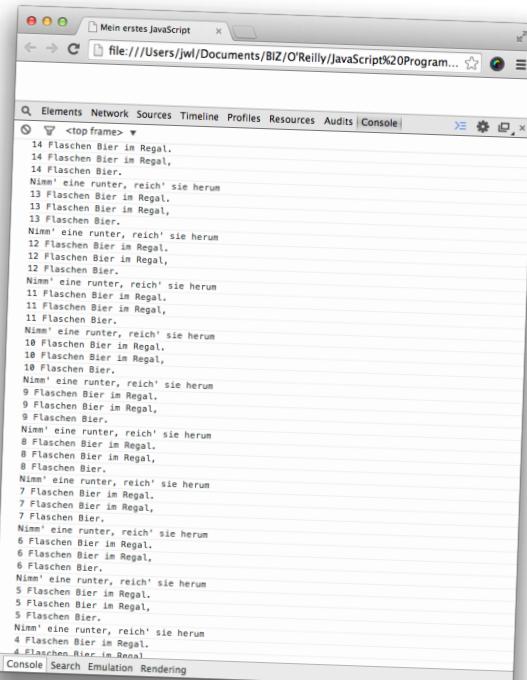
- ③ Nun müssen wir »index.html« mit einer Referenz auf »code.js« versehen, damit Ihr Code beim Laden der Seite eingebunden werden kann. Entfernen Sie hierfür den JavaScript-Code aus »index.html«, lassen Sie die `<script>`-Tags aber stehen. Dann erweitern Sie das öffnende `<script>`-Tag um ein `src`-Attribut, mit dem »code.js« eingebunden wird:

```
<!doctype html>
<html lang="de">
<head>
<meta charset="utf-8">
<title>Mein erstes JavaScript</title>
</head>
<body>
<script src="code.js">
</script> Hier stand Ihr Code.
</body>
</html> Ob Sie's glauben oder nicht – das schließende </script>-Tag wird gebraucht, auch wenn zwischen den Tags kein Code steht.
```

Benutzen Sie das `src`-Attribut des `<script>`-Elements, um Ihre JavaScript-Datei einzubinden.

- ④ Operation geglückt – Patient lebt. Rufen Sie zum Testen erneut die Datei »index.html« in Ihrem Browser auf. Das Ergebnis sollte aussehen wie zuvor. Beachten Sie, dass wir bei der Verwendung von `src="code.js"` davon ausgehen, dass »index.html« und »code.js« im selben Verzeichnis liegen.

Das Ergebnis sollte genauso aussehen wie vorhin. Nur liegen HTML und JavaScript jetzt in eigenen Dateien. Fühlt sich das nicht viel sauberer, wartbarer und stressfreier an?





Anatomie eines <script>-Elements

Inzwischen wissen Sie, wie Sie Code anhand des <script>-Elements in Ihre Seite einfügen können. Um diesen Nagel ganz einzuschlagen und nichts zu vergessen, wollen wir uns genau ansehen, wie das <script>-Element im Detail funktioniert:

Über das type-Attribut können Sie dem Browser mitteilen, dass Sie JavaScript schreiben. Da der Browser standardmäßig davon ausgeht, empfehlen wir Ihnen, das Attribut einfach wegzulassen. Das machen die Leute, die die Standards schreiben, schließlich auch so.

Das öffnende <script>-Tag.

<script type="text/javascript">

Vergessen Sie beim öffnenden Tag nicht die schließende spitze Klammer.

alert("Hallo Welt!");

Zwischen den <script>-Tags darf nur gültiges JavaScript stehen!

</script>

Das Skript braucht ein schließendes </script>-Tag. Immer!

Und wenn Sie eine externe JavaScript-Datei in Ihr HTML-Dokument einbinden wollen, benutzen Sie das <script>-Element wie hier gezeigt:

Verwenden Sie ein src-Attribut, um die URL der JavaScript-Datei anzugeben.

<script src="myJavaScript.js">

Versehen Sie JavaScript-Dateien mit der Endung >>.js<<.

</script>

Wenn Sie eine externe JavaScript-Datei einbinden, darf zwischen den <script>-Tags kein JavaScript stehen.

Vergessen Sie auch hier keinesfalls das schließende </script>-Tag! Es wird ebenfalls gebraucht, wenn Sie eine externe Datei einbinden.



Aufgepasst

Sie können nicht gleichzeitig einbinden und einbetten.

Wenn Sie mal eben etwas Code zwischen <script>-Tags schreiben, die bereits ein src-Attribut besitzen, wird Ihr Code nicht funktionieren. Sie brauchen zwei separate <script>-Elemente.

```
<script src="goodies.js">  
var = "kleiner Hack";  
</script>
```

FALSCH



JavaScript im Gespräch

Interview der Woche: JavaScripts geheime Beichte

Von Kopf bis Fuß: Willkommen, JavaScript. Wir wissen, Sie sind gerade sehr beschäftigt mit der Arbeit in all den Webseiten. Wir freuen uns, dass Sie trotzdem die Zeit gefunden haben, mit uns zu reden.

JavaScript: Kein Problem. Ich bin dieser Tage tatsächlich *sehr* beschäftigt: Die Leute benutzen JavaScript mittlerweile auf fast jeder Webseite — für einfache Menüeffekte bis hin zu ausgewachsenen Spielen. Es ist echt verrückt!

Von Kopf bis Fuß: Erstaunlich, wenn man bedenkt, dass Sie noch vor ein paar Jahren nur eine »halbwüchsige, schwachbrüstige Skriptsprache« waren. Und jetzt sind Sie überall.

JavaScript: Daran mag ich gar nicht denken. Seitdem hat sich aber viel getan, und viele kluge Köpfe haben daran gearbeitet, mich ständig zu verbessern.

Von Kopf bis Fuß: Wie das denn? Ihre grundsätzlichen Sprachmerkmale haben sich doch kaum verändert...

Von Kopf bis Fuß: Verbesserungen gibt es an vielen Stellen. Zum einen bin ich heutzutage blitzschnell. Und obwohl man mich als Skriptsprache bezeichnet, bin ich fast so schnell und leistungsfähig wie native kompilierte Sprachen.

Von Kopf bis Fuß: Und was noch?

JavaScript: Meine Möglichkeiten, Dinge im Browser zu tun, wurden dramatisch erweitert. Mit den für alle Browser verfügbaren Bibliotheken kann ich Ihren Aufenthaltsort ermitteln, Video und Audio abspielen, Grafiken direkt in der Webseite erstellen und vieles mehr. Dafür müssen Sie JavaScript natürlich auch schreiben können.

Von Kopf bis Fuß: Ein paar Kritikpunkte gegen Sie als Sprache sollten wir noch klären. Einige weniger nette Kommentare bezeichneten Sie, glaube ich, als »ziemlich zusammengestückelte« oder »gehackte Sprache«.

JavaScript: Da kann ich gut mit leben. Ich bin immerhin eine, wenn nicht *die* meistverwendete Programmiersprache der Welt. Ich habe bereits gegen viele Konkurrenten gekämpft – und gewonnen. Erinnern Sie sich noch an »Java im Browser«? So ein Witz! VBScript? Ha! JScript? Flash? Silverlight? Die Liste ist noch lang. So schlecht kann ich wohl nicht sein.

Von Kopf bis Fuß: Sie wurden auch schon mal als »zu vereinfachend« kritisiert. Was sagen Sie dazu?

JavaScript: Das ist eine meiner größten Stärken. Starten Sie einfach einen Browser, geben Sie ein paar Zeilen JavaScript ein, und

schon läuft das Programm. Das ist wahre Stärke. Und auch Anfänger kommen gut mit mir klar. Manche Leute sagen, JavaScript ist die beste Programmiersprache für Anfänger überhaupt.

Von Kopf bis Fuß: Aber diese Einfachheit hat doch ihren Preis, oder?

JavaScript: Das ist ja das Tolle. Ich bin insofern einfach, als dass Sie schnell loslegen können. Trotzdem stecke ich randvoll mit modernen Programmierkonzepten.

Von Kopf bis Fuß: So? Was denn zum Beispiel?

JavaScript: Nun, wie wäre es beispielsweise mit dynamischen Datentypen, First-Class-Funktionen und Closures?

Von Kopf bis Fuß: Keine Ahnung, was das ist.

JavaScript: Hmm ... schon okay. Wenn Sie das Buch weiterlesen, werden Sie verstehen, was ich meine.

Von Kopf bis Fuß: Dann geben Sie uns zumindest einen Überblick.

JavaScript: Lassen Sie mich nur so viel sagen: JavaScript wurde für eine dynamische Webumgebung entwickelt, in der der Benutzer mit einer Seite interagiert, in der Daten nach Bedarf erzeugt und geladen werden und in der viele verschiedene Events passieren. All dies schlägt sich in der Sprache und der Art zu programmieren nieder. Sie werden das wirklich besser verstehen, wenn Sie das Buch weiterlesen und etwas mehr Verständnis von JavaScript haben.

Von Kopf bis Fuß: Ja, ich glaube, das haben Sie schon gesagt. Sie sind also die perfekte Sprache, richtig?

JavaScript (bekommt feuchte Augen ...)

JavaScript: Wissen Sie, ich bin nicht, wie die meisten Sprachen, innerhalb der efeubewachsenen Mauern einer Hochschule aufgewachsen. Ich wurde in die wirkliche Welt hineingeboren und musste entweder schnell schwimmen lernen oder untergehen. Davon abgesehen, bin ich sicher nicht perfekt; auch ich habe meine schlechten Seiten.

Von Kopf bis Fuß mit einem Anflug von Dagmar-

Berghoff-Lächeln: Wir haben Sie heute von einer neuen Seite kennengelernt. Ich denke, das rechtfertigt ein weiteres Interview zu einem späteren Zeitpunkt. Noch ein Gedanke zum Abschied?

JavaScript: Beurteilen Sie mich nicht nach meinen Nachteilen. Lernen Sie das Gute und bleiben Sie dabei!

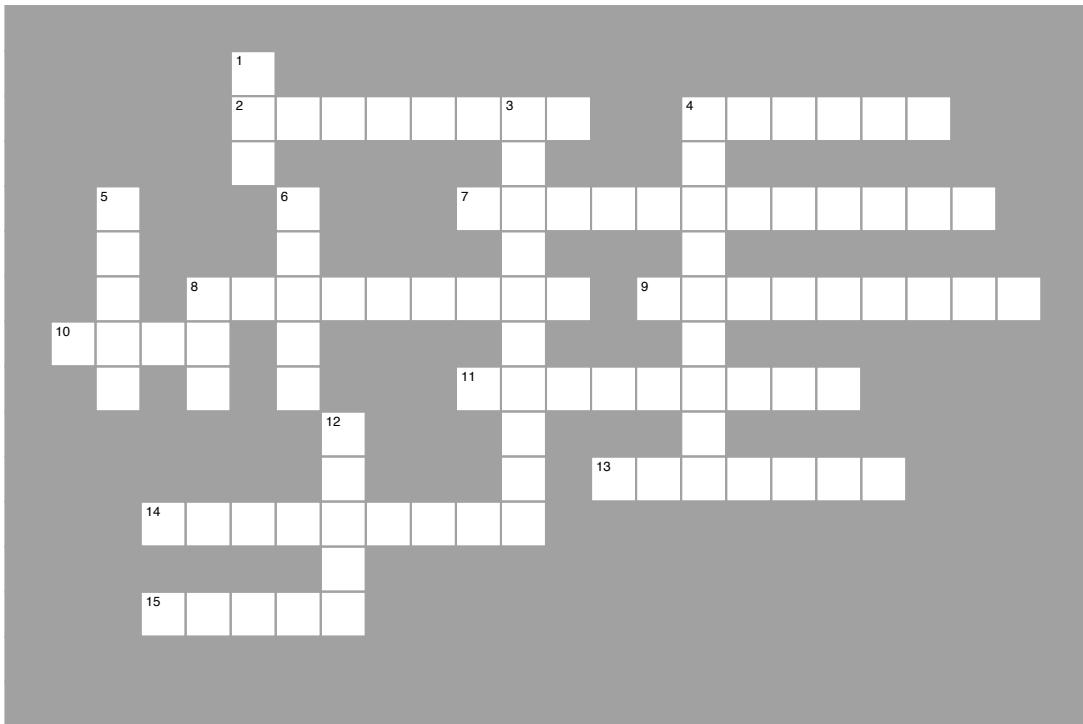
Punkt für Punkt

- Mit JavaScript können Sie Webseiten mit **Verhalten** versehen.
- Browser-Engines können JavaScript deutlich schneller ausführen als noch vor ein paar Jahren.
- Browser beginnen mit der Ausführung von JavaScript, sobald sie den Code in der Seite finden.
- JavaScript wird anhand des `<script>`-Elements in die Seite eingefügt.
- Sie können JavaScript direkt in die Webseite einbetten oder eine externe JavaScript-Datei per HTML einbinden.
- Soll eine externe JavaScript-Datei eingebunden werden, versehen Sie das `<script>`-Tag mit einem entsprechenden **src**-Attribut.
- HTML **deklariert** die Struktur und den Inhalt Ihrer Seite; JavaScript **berechnet** Werte und versieht die Seite mit Verhalten.
- JavaScript-Programme bestehen aus einer Reihe von **Anweisungen**.
- Eine der häufigsten JavaScript-Anweisungen ist die Deklaration von Variablen. Mit dem Schlüsselwort **var** wird die Variable deklariert; mit dem Operator **=** kann ihr ein Wert zugewiesen werden.
- Für die Benennung von Variablen in JavaScript gibt es nur wenige Regeln und Richtlinien. Halten Sie sich daran!
- Vermeiden Sie JavaScript-Schlüsselwörter bei der Benennung von Variablen.
- JavaScript-Ausdrücke berechnen Werte.
- Drei häufige Arten von Ausdrücken sind **numerische**, **String-** und **Boolesche** Ausdrücke.
- Mit **if/else**-Anweisungen können Sie in Ihrem Code Entscheidungen treffen.
- **while/for**-Anweisungen ermöglichen das wiederholte Ausführen von Code innerhalb einer Schleife.
- Verwenden Sie **console.log** anstelle von **alert**, um Nachrichten auf der Konsole auszugeben.
- Konsolenmeldungen sollten vornehmlich für die Ausgabe von Meldungen zur Fehlersuche verwendet werden, da die meisten Benutzer die Konsolenmeldungen wahrscheinlich nie zu sehen bekommen.
- JavaScript wird hauptsächlich verwendet, um Webseiten mit Verhalten zu versehen. Es kommt aber auch in Programmen wie Adobe Photoshop, OpenOffice oder Google Apps zum Einsatz und wird sogar als serverseitige Programmiersprache immer beliebter.



JavaScript-Kreuzworträtsel

Es ist Zeit, Ihre Dendriten mit einem kleinen Rätsel ein wenig zu dehnen, damit von diesem Kapitel auch etwas hängen bleibt.



WAAGERECHT

2. $3 + 4$ ist ein Beispiel für einen _____.
4. Innerhalb einer HTML-Seite steht der JavaScript-Code in einem _____-Element.
7. Mit einer if/else-Anweisung können Sie eine _____ treffen.
8. Alle JavaScript-Anweisungen enden mit einem _____.
9. Bei jedem Schleifendurchlauf wird eine _____ ausgewertet.
10. Es stehen 99 Flaschen _____ im Regal.
11. JavaScript versieht Ihre Seiten mit _____.
13. Mithilfe des Operators + können Sie mehrere _____ miteinander verketten.
14. Eine Zeile JavaScript-Code nennt man eine _____.
15. Variablen werden benutzt, um _____ zu speichern.

SENKRECHT

1. Um eine Variable zu deklarieren, verwenden Sie das Schlüsselwort _____.
3. Um Fehler in Ihrem Code zu finden, benutzen Sie _____.
4. Heutzutage läuft JavaScript deutlich _____ als früher.
5. Bei der Benennung von JavaScript-Variablen wird zwischen Groß- und _____-Schreibung unterschieden.
6. Um peinliche Namensfehler zu vermeiden, verwenden Sie _____-Case.
8. Um eine externe JavaScript-Datei einzubinden, versehen Sie Ihr <script>-Element mit einem _____-Attribut.
12. Um Dinge in JavaScript wiederholt auszuführen, können Sie eine _____-Schleife verwenden.

SPIELEN Sie Browser, Lösung



Unten finden Sie JavaScript-Code,
der einige Fehler enthält. Es ist
Ihre Aufgabe, die Fehler im Code
zu finden. Hier ist unsere Lösung.

A

```
// Nach Pointen suchen
var joke = "JavaScript kam in eine Bar....";
var toldJoke = "false";
var $punchline = "Wir haben nur Semi-Kola"
var %entage = 20;
var result;

if (toldJoke == true) {
    Alert($punchline);
} else
    alert(joke);
```

Umgeben Sie Ihre Strings mit einem doppelten ("") oder einfachen ('') Anführungszeichen. Nicht mischen!

Boolesche Werte werden nicht mit Anführungszeichen versehen, es sei denn, Sie brauchen wirklich einen String.

Es ist erlaubt, aber nicht empfohlen, Variablennamen mit einem \$ beginnen zu lassen.

Vergessen Sie nicht, alle Anweisungen mit einem Semikolon zu beenden!

Das % ist in Variablennamen nicht erlaubt.

Noch ein fehlendes Semikolon.

Das sollte alert heißen, nicht Alert. JavaScript unterscheidet zwischen Groß- und Kleinschreibung.

Hier fehlt die öffnende geschweifte Klammer.

B

```
\\" Kinoabend
var zip code = 98104;
var joe'sFavoriteMovie = Alarm im Weltall;
var movieTicket$ = 9;

if (movieTicket$ >= 9) {
    alert("Zu teuer!");
} else {
    alert("Wir sehen heute " + joe'sFavoriteMovie);
```

Kommentare beginnen mit //, nicht mit \\\

Variablennamen dürfen keine Leerzeichen enthalten.

Der String >>Alarm im Weltall<< muss mit Anführungszeichen umgeben werden.

Anführungszeichen sind in Variablennamen nicht erlaubt.

Die if/else-Bedingung funktioniert nicht, weil der Variablenname ungültig ist.



Spitzen Sie Ihren Bleistift

Lösung

Holen Sie Ihren Bleistift raus und probieren Sie ein paar Ausdrücke durch. Berechnen Sie die Werte und schreiben Sie die Antwort auf. Richtig: AUFSCHREIBEN! Vergessen Sie, dass Ihre Mutter Ihnen verboten hat, in Bücher zu schreiben, und bringen Sie Ihre Antworten gleich hier zu Papier! Hier ist unsere Lösung.

Kann man >>Fahrenheit-nach-Celsius-Rechner<< sagen?

`(9 / 5) * temp + 32`

Dies ist ein Boolescher Ausdruck.
Der Operator `==` überprüft, ob
zwei Werte gleich sind.

`color == "orange"`

`name + ", " + "Sie haben gewonnen!"`

`yourLevel > 5`
Hier wird überprüft, ob
der erste Wert größer
ist als der zweite. Um zu
testen, ob der erste Wert
größer oder gleich dem
zweiten ist, können Sie den
Operator `>=` verwenden.

`(level * points) + bonus`

`color != "orange"`

Der Operator `!=` testet, ob die
beiden Werte NICHT gleich sind.

ZUSATZPUNKTE!

`1000 + "108"`

Sind mehrere Antworten möglich? Nur
eine ist richtig. Welche wählen Sie?
"1000108"

Wie lautet das Ergebnis, wenn `temp` den Wert 10 hat?

50

Ist dieser Ausdruck wahr oder falsch, wenn `color` den Wert
»pink« hat? false

Was, wenn er den Wert »orange« hat? true

Welcher Wert wird berechnet, wenn `name` den
Wert »Martha« hat? "Martha, Sie haben gewonnen!"

Wie lautet das Ergebnis, wenn `yourLevel` den Wert 2 hat? false

Wie lautet das Ergebnis, wenn `yourLevel` den Wert 5 hat? false

Wie lautet das Ergebnis, wenn `yourLevel` den Wert 7 hat? true

Okay, `level` hat den Wert 5, `points` ist 30000 und `bonus` ist 3300.

Welches Ergebnis bekommen wir? 153300

Ist dieser Ausdruck wahr (true) oder falsch (false), wenn
`color` den Wert »pink« hat? true



Seriöses Coden

Haben Sie bemerkt, dass der
Operator `=` für Zuweisungen
benutzt wird, während `==` auf Gleichheit
testet? Das heißt, wir verwenden ein
Gleichheitszeichen für Zuweisungen. Zwei
Gleichheitszeichen testen, ob zwei Werte
tatsächlich gleich sind. Beim Programmieren
kann es schnell passieren, dass man
die beiden Operatoren verwechselt.



Codemagneten, Lösung

Unser JavaScript am Kühlschrank ist durcheinandergeraten. Können Sie die Magneten an die richtige Stelle schieben, sodass ein funktionierendes JavaScript entsteht, das die unten stehende Ausgabe erzeugt? Hier ist unsere Lösung.

Hier sehen Sie die Magneten wieder in der richtigen Reihenfolge:



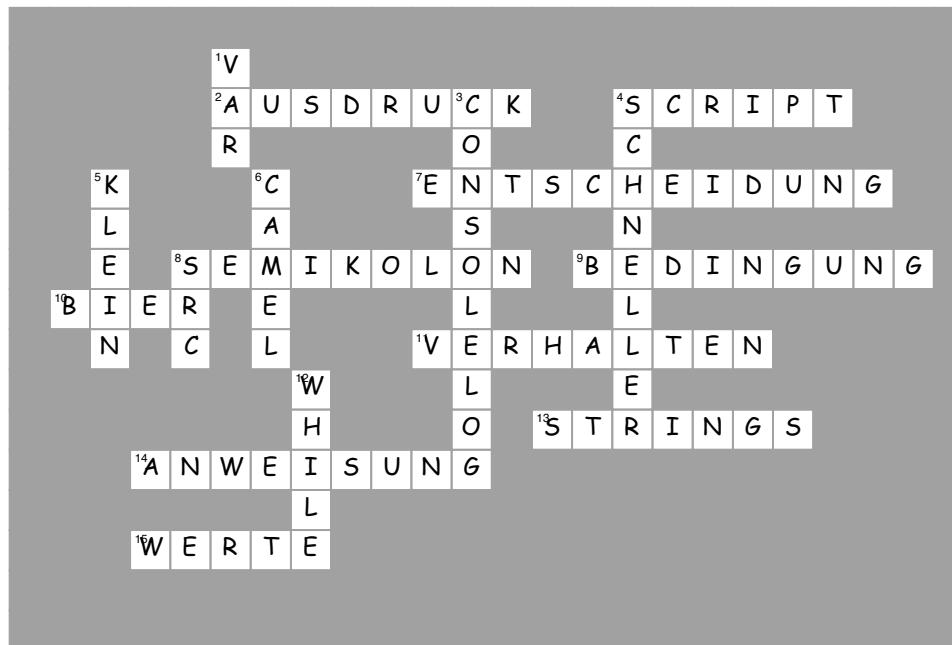
```
var name = "Jørgen";  
  
var i = 0;  
  
while (i < 2) {  
  
    document.write("Zum Geburtstag viel Glück.<br>");  
  
    i = i + 1;  
  
}  
  
document.write("zum Geburtstag, lieber " + name + ",<br>");  
  
document.write("Zum Geburtstag viel Glück.<br>");
```

Das Programm sollte diese
Ausgabe erzeugen.
↓

The screenshot shows a browser window titled "Herzlichen Glückwunsch". The address bar indicates the file is located at "file:///Users/jwl/Documents/BIZ/O...". The page content displays the following text:
Zum Geburtstag viel Glück.
Zum Geburtstag viel Glück.
Zum Geburtstag, lieber Jørgen,
Zum Geburtstag viel Glück.



JavaScript Kreuz- worträtsel, Lösung



WER MACHT WAS? — LÖSUNG

Alle Kommunikationsmethoden sind maskiert zur Party gekommen. Können Sie uns helfen, ihre wahre Identität aufzudecken? Verbinden Sie die Beschreibungen auf der rechten Seite mit den Namen auf der linken. Die erste Verbindung hatten wir schon für Sie hergestellt.

`document.write`

Ich halte den Benutzer an und gebe eine kurze Meldung aus. Der Benutzer muss erst auf »OK« klicken, damit es weitergeht.

`console.log`

Ich füge etwas HTML und Text in die Seite ein. Ich bin zwar nicht die eleganste Methode, um Nachrichten an Ihre Benutzer auszugeben, aber ich funktioniere in jedem Browser.

`alert`

Benutzen Sie mich, um eine Webseite komplett zu kontrollieren: Ich kann Benutzereingaben auslesen, den HTML-Code und CSS-Stile ändern sowie den Inhalt Ihrer Seite aktualisieren.

Document Object Model (DOM)

Ich bin eigentlich nur zur Fehlersuche da. Benutzen Sie mich, und ich werde Informationen auf einer speziellen Entwicklerkonsole ausgeben.