

Wolfgang Henderkes

JavaScript Tutor

... für das erfolgreiche Selbststudium!



Technik von gestern: Die Zeche Zollern in Dortmund.

Diese Shareware-Version enthält nur das erste Kapitel mit der Einführung aus dem Tutor und das komplette Lexikon. In der Vollversion sind fünf weitere Kapitel (Funktionen, Bedingungen, Schleifen, Objekte und Beispielprojekt) vorhanden. Die Vollversion können Sie direkt hier [bestellen und downloaden](#).

Inhaltsverzeichnis

HuT	8
Copyright	8
Haftung	9
Impressum	10
Wenn es Sie interessiert.....	11
Über dieses (e)Buch	11
Voraussetzungen an den Leser.....	14
Technische Voraussetzungen	15
Konventionen	17
Einführung.....	18
Die Sprachen des Internet.....	18
Überblick.....	18
HTML	20
JavaScript	23
JScript.....	26
VBScript.....	27
XML	28
JAVA.....	29
Risiken	30
Quiz Sprachen	31
Werkzeuge	32
Der Browser.....	32
Keinen Browser bevorzugen	33
Der Editor.....	35
Text aus Browser kopieren.....	36
Quelltext im Browser anzeigen.....	37
Quelltext im ASCII-Editor	40
Mehrere Browser öffnen.....	41
Quiz Werkzeuge	42
HTML und JavaScript.....	43
Kein Teil von HTML.....	43
JavaScript-Bereich definieren.....	46
Versionen – ohne Bedeutung.....	49
Kommentare	50
Programm ausführen	53
JavaScript-Code in separaten Dateien	54
Anweisungen notieren.....	57
Anweisungsblöcke notieren.....	58
Zuweisungen und Gleichheit	59
Ausdrücke.....	60
Namensgebung.....	61
Sauberer Programmierstil	62
Quiz HTML und JavaScript I	64
Quiz HTML und JavaScript II.....	64
Quiz HTML und JavaScript III.....	64
Variablen.....	65
Was ist eine Variable?	65
Deklaration.....	66
Konvertierung.....	69
Datentyp für Text	71
Datentyp für Zahlen.....	72
Datentyp null	74
Quiz Variablen I	75
Quiz Variablen II	75
Quiz Variablen III	75
Schlüsselwörter.....	76
Reservierte Namen	76
Operatoren	79
... für Text.....	79
... für Zahlen.....	80
... für logische Verknüpfungen.....	82
Verzeichnis der Operatoren	83
Quiz Operatoren I	84
Quiz Operatoren II.....	84
Sonderzeichen	85
Escape-Sequenzen.....	85
Objektbasierte Struktur.....	86
Hierarchische Struktur.....	86

Eigenschaften und Methoden.....	88
Quiz Objekte I.....	90
Quiz Objekte II.....	90
Quiz Objekte III.....	90
Funktionen.....	Fehler! Textmarke nicht definiert.
Frei definierbare Funktionen.....	Fehler! Textmarke nicht definiert.
Funktion deklarieren.....	Fehler! Textmarke nicht definiert.
Funktion aufrufen.....	Fehler! Textmarke nicht definiert.
Einsatz der Variablen.....	Fehler! Textmarke nicht definiert.
Rückgabewerte.....	Fehler! Textmarke nicht definiert.
Argumente übergeben.....	Fehler! Textmarke nicht definiert.
Vordefinierte Funktionen.....	Fehler! Textmarke nicht definiert.
Die escape()-Funktion.....	Fehler! Textmarke nicht definiert.
Die eval()-Funktion.....	Fehler! Textmarke nicht definiert.
Die isFinite()-Funktion.....	Fehler! Textmarke nicht definiert.
Die isNaN()-Funktion.....	Fehler! Textmarke nicht definiert.
Die Number()-Funktion.....	Fehler! Textmarke nicht definiert.
Die parseFloat()-Funktion.....	Fehler! Textmarke nicht definiert.
Die parseInt()-Funktion.....	Fehler! Textmarke nicht definiert.
Die String()-Funktion.....	Fehler! Textmarke nicht definiert.
Die unescape()-Funktion.....	Fehler! Textmarke nicht definiert.
Event-Handler.....	Fehler! Textmarke nicht definiert.
Attribute in Tags.....	Fehler! Textmarke nicht definiert.
Sonderfall Link.....	Fehler! Textmarke nicht definiert.
Fragen & Antworten.....	Fehler! Textmarke nicht definiert.
Quiz Funktionen I.....	Fehler! Textmarke nicht definiert.
Quiz Funktionen II.....	Fehler! Textmarke nicht definiert.
Quiz Funktionen III.....	Fehler! Textmarke nicht definiert.
Quiz Funktionen IV.....	Fehler! Textmarke nicht definiert.
Bedingungen.....	Fehler! Textmarke nicht definiert.
Die if...else-Struktur.....	Fehler! Textmarke nicht definiert.
Logische Vergleiche.....	Fehler! Textmarke nicht definiert.
Mehrere Bedingungen.....	Fehler! Textmarke nicht definiert.
Implizite Bedingung.....	Fehler! Textmarke nicht definiert.
Existenz von Objekten.....	Fehler! Textmarke nicht definiert.
Quiz Bedingungen.....	Fehler! Textmarke nicht definiert.
Schleifen.....	Fehler! Textmarke nicht definiert.
Schleifenarten.....	Fehler! Textmarke nicht definiert.
for-Schleifen.....	Fehler! Textmarke nicht definiert.
while-Schleifen.....	Fehler! Textmarke nicht definiert.
Quiz Schleifen.....	Fehler! Textmarke nicht definiert.
Objekte.....	Fehler! Textmarke nicht definiert.
Die Struktur.....	Fehler! Textmarke nicht definiert.
Vordefinierte Objekte.....	Fehler! Textmarke nicht definiert.
Eigene Objekte erstellen.....	Fehler! Textmarke nicht definiert.
Das window-Objekt.....	Fehler! Textmarke nicht definiert.
window: Übersicht.....	Fehler! Textmarke nicht definiert.
Das anchors-Objekt.....	Fehler! Textmarke nicht definiert.
Das applets-Objekt.....	Fehler! Textmarke nicht definiert.
Das document-Objekt.....	Fehler! Textmarke nicht definiert.
Das document-Objekt: Beispiele.....	Fehler! Textmarke nicht definiert.
Das elements-Objekt.....	Fehler! Textmarke nicht definiert.
Das forms-Objekt.....	Fehler! Textmarke nicht definiert.
Das history-Objekt.....	Fehler! Textmarke nicht definiert.
Das images-Objekt.....	Fehler! Textmarke nicht definiert.
Das links-Objekt.....	Fehler! Textmarke nicht definiert.
Das location-Objekt.....	Fehler! Textmarke nicht definiert.
Das options-Objekt.....	Fehler! Textmarke nicht definiert.
Das navigator-Objekt.....	Fehler! Textmarke nicht definiert.
navigator: Übersicht.....	Fehler! Textmarke nicht definiert.
Objekte ohne Hierarchie.....	Fehler! Textmarke nicht definiert.
Das Array-Objekt.....	Fehler! Textmarke nicht definiert.
Das Boolean-Objekt.....	Fehler! Textmarke nicht definiert.
Das Date-Objekt.....	Fehler! Textmarke nicht definiert.
Das Function-Objekt.....	Fehler! Textmarke nicht definiert.
Das Global-Objekt.....	Fehler! Textmarke nicht definiert.
Das Math-Objekt.....	Fehler! Textmarke nicht definiert.
Das Number-Objekt.....	Fehler! Textmarke nicht definiert.
Das Object-Objekt.....	Fehler! Textmarke nicht definiert.
Das RegExp-Objekt.....	Fehler! Textmarke nicht definiert.
Das Screen-Objekt.....	Fehler! Textmarke nicht definiert.
Das String-Objekt.....	Fehler! Textmarke nicht definiert.
Fragen & Antworten.....	Fehler! Textmarke nicht definiert.
Quiz Objekte IV.....	Fehler! Textmarke nicht definiert.
Beispielprojekt.....	Fehler! Textmarke nicht definiert.
Die Tragödie beginnt.....	Fehler! Textmarke nicht definiert.
Teil I.....	Fehler! Textmarke nicht definiert.

Teil II.....	Fehler! Textmarke nicht definiert.
Teil III.....	Fehler! Textmarke nicht definiert.
Teil IV.....	Fehler! Textmarke nicht definiert.
Teil V.....	Fehler! Textmarke nicht definiert.
Teil VI.....	Fehler! Textmarke nicht definiert.
Teil VII.....	Fehler! Textmarke nicht definiert.
Teil VIII.....	Fehler! Textmarke nicht definiert.
Teil IX.....	Fehler! Textmarke nicht definiert.
Teil X.....	Fehler! Textmarke nicht definiert.
Der Vorhang fällt.....	Fehler! Textmarke nicht definiert.
Lexikon JavaScript.....	Fehler! Textmarke nicht definiert.
Abstract Windowing Toolkit.....	Fehler! Textmarke nicht definiert.
Absturz.....	Fehler! Textmarke nicht definiert.
Active Server Pages.....	Fehler! Textmarke nicht definiert.
ActiveX.....	Fehler! Textmarke nicht definiert.
ADSL.....	Fehler! Textmarke nicht definiert.
Agent.....	Fehler! Textmarke nicht definiert.
America Online.....	Fehler! Textmarke nicht definiert.
animierte Grafiken.....	Fehler! Textmarke nicht definiert.
ANSI.....	Fehler! Textmarke nicht definiert.
ANSI-Zeichensatz.....	Fehler! Textmarke nicht definiert.
Anweisung.....	Fehler! Textmarke nicht definiert.
Anweisungsblock.....	Fehler! Textmarke nicht definiert.
Applet.....	Fehler! Textmarke nicht definiert.
Archie.....	Fehler! Textmarke nicht definiert.
Argument.....	Fehler! Textmarke nicht definiert.
Arpanet.....	Fehler! Textmarke nicht definiert.
Artikel.....	Fehler! Textmarke nicht definiert.
ASCII-Zeichensatz.....	Fehler! Textmarke nicht definiert.
ASP.....	Fehler! Textmarke nicht definiert.
Authenticode.....	Fehler! Textmarke nicht definiert.
Bedingung.....	Fehler! Textmarke nicht definiert.
Befehl.....	Fehler! Textmarke nicht definiert.
BISAC.....	Fehler! Textmarke nicht definiert.
Body.....	Fehler! Textmarke nicht definiert.
Book Industry Systems Advisory Committee.....	Fehler! Textmarke nicht definiert.
Boole'sche Algebra.....	Fehler! Textmarke nicht definiert.
Boolean.....	Fehler! Textmarke nicht definiert.
Browser.....	Fehler! Textmarke nicht definiert.
Btx.....	Fehler! Textmarke nicht definiert.
CGI.....	Fehler! Textmarke nicht definiert.
CGI-Verzeichnis.....	Fehler! Textmarke nicht definiert.
Chat.....	Fehler! Textmarke nicht definiert.
Client-Server Mailbox.....	Fehler! Textmarke nicht definiert.
Common Gateway Interface.....	Fehler! Textmarke nicht definiert.
Comprehensive Perl Archive Network.....	Fehler! Textmarke nicht definiert.
CompuServe.....	Fehler! Textmarke nicht definiert.
Cookie.....	Fehler! Textmarke nicht definiert.
CPAN.....	Fehler! Textmarke nicht definiert.
Crashmail.....	Fehler! Textmarke nicht definiert.
Datentyp.....	Fehler! Textmarke nicht definiert.
Datex-J.....	Fehler! Textmarke nicht definiert.
Delimiter.....	Fehler! Textmarke nicht definiert.
DENIC.....	Fehler! Textmarke nicht definiert.
DE-NIC.....	Fehler! Textmarke nicht definiert.
Deutsches Network Information Center.....	Fehler! Textmarke nicht definiert.
DFÜ.....	Fehler! Textmarke nicht definiert.
DHTML.....	Fehler! Textmarke nicht definiert.
DNS.....	Fehler! Textmarke nicht definiert.
DNS-Server.....	Fehler! Textmarke nicht definiert.
Dokumentaustausch.....	Fehler! Textmarke nicht definiert.
Dokumentenbeschreibungssprache.....	Fehler! Textmarke nicht definiert.
DOM.....	Fehler! Textmarke nicht definiert.
Domain.....	Fehler! Textmarke nicht definiert.
Domain Name Service.....	Fehler! Textmarke nicht definiert.
Download.....	Fehler! Textmarke nicht definiert.
dynamische HTML-Seiten.....	Fehler! Textmarke nicht definiert.
eBook.....	Fehler! Textmarke nicht definiert.
eBook-Reader.....	Fehler! Textmarke nicht definiert.
eBuch.....	Fehler! Textmarke nicht definiert.
Editor.....	Fehler! Textmarke nicht definiert.
elektronisches Buch.....	Fehler! Textmarke nicht definiert.
eMail.....	Fehler! Textmarke nicht definiert.
Entity.....	Fehler! Textmarke nicht definiert.
EWorld.....	Fehler! Textmarke nicht definiert.
Excalibur Mailbox.....	Fehler! Textmarke nicht definiert.
falsch.....	Fehler! Textmarke nicht definiert.

false	Fehler! Textmarke nicht definiert.
FAQ	Fehler! Textmarke nicht definiert.
Fehlermeldung	Fehler! Textmarke nicht definiert.
Fidonet	Fehler! Textmarke nicht definiert.
Firewall	Fehler! Textmarke nicht definiert.
Flame	Fehler! Textmarke nicht definiert.
Frequently Asked Questions	Fehler! Textmarke nicht definiert.
FTP	Fehler! Textmarke nicht definiert.
Funktion	Fehler! Textmarke nicht definiert.
"	Fehler! Textmarke nicht definiert.
Funktionsaufruf	Fehler! Textmarke nicht definiert.
Funktionsdeklaration	Fehler! Textmarke nicht definiert.
Gateway	Fehler! Textmarke nicht definiert.
GIF	Fehler! Textmarke nicht definiert.
globale Variable	Fehler! Textmarke nicht definiert.
Gopher	Fehler! Textmarke nicht definiert.
Graphics Interchange Format	Fehler! Textmarke nicht definiert.
HomePage	Fehler! Textmarke nicht definiert.
HTML	Fehler! Textmarke nicht definiert.
HTML-Editor	Fehler! Textmarke nicht definiert.
HTML-Tag	Fehler! Textmarke nicht definiert.
Hyperlink	Fehler! Textmarke nicht definiert.
Hypertext	Fehler! Textmarke nicht definiert.
Hypertext Markup Language	Fehler! Textmarke nicht definiert.
Hypertext Transfer Protocol	Fehler! Textmarke nicht definiert.
IETF	Fehler! Textmarke nicht definiert.
if-Bedingung	Fehler! Textmarke nicht definiert.
if-then-else	Fehler! Textmarke nicht definiert.
Instanz	Fehler! Textmarke nicht definiert.
interaktiv	Fehler! Textmarke nicht definiert.
International Standard Book Number	Fehler! Textmarke nicht definiert.
Internet	Fehler! Textmarke nicht definiert.
Internet Relay Chat	Fehler! Textmarke nicht definiert.
Internet Server API	Fehler! Textmarke nicht definiert.
Internet Service Provider	Fehler! Textmarke nicht definiert.
Internet-Dienst	Fehler! Textmarke nicht definiert.
Intranet	Fehler! Textmarke nicht definiert.
IP	Fehler! Textmarke nicht definiert.
IP-Adresse	Fehler! Textmarke nicht definiert.
IRC	Fehler! Textmarke nicht definiert.
ISAPI	Fehler! Textmarke nicht definiert.
ISBN	Fehler! Textmarke nicht definiert.
ISO 8859	Fehler! Textmarke nicht definiert.
ISO 9660	Fehler! Textmarke nicht definiert.
ISP	Fehler! Textmarke nicht definiert.
JAVA	Fehler! Textmarke nicht definiert.
JAVA-Applet	Fehler! Textmarke nicht definiert.
JAVA-Bean	Fehler! Textmarke nicht definiert.
JavaScript	Fehler! Textmarke nicht definiert.
JScript	Fehler! Textmarke nicht definiert.
Kerberos	Fehler! Textmarke nicht definiert.
Kermit	Fehler! Textmarke nicht definiert.
Latin	Fehler! Textmarke nicht definiert.
Laufzeitfehler	Fehler! Textmarke nicht definiert.
Lesegerät	Fehler! Textmarke nicht definiert.
Linux	Fehler! Textmarke nicht definiert.
logische Operation	Fehler! Textmarke nicht definiert.
logischer Operator	Fehler! Textmarke nicht definiert.
lokale Variable	Fehler! Textmarke nicht definiert.
Mailbox	Fehler! Textmarke nicht definiert.
Meta-Tag	Fehler! Textmarke nicht definiert.
Methode	Fehler! Textmarke nicht definiert.
Methodenaufruf	Fehler! Textmarke nicht definiert.
MIME	Fehler! Textmarke nicht definiert.
MP3	Fehler! Textmarke nicht definiert.
Multipurpose Internet Mail Extensions	Fehler! Textmarke nicht definiert.
NaN	Fehler! Textmarke nicht definiert.
Netiquette	Fehler! Textmarke nicht definiert.
Newsgroup	Fehler! Textmarke nicht definiert.
NIC	Fehler! Textmarke nicht definiert.
Not a Number	Fehler! Textmarke nicht definiert.
Objekt	Fehler! Textmarke nicht definiert.
Objektinstanz	Fehler! Textmarke nicht definiert.
ODER-Verknüpfung	Fehler! Textmarke nicht definiert.
Online-Dienst	Fehler! Textmarke nicht definiert.
PDF	Fehler! Textmarke nicht definiert.
Perl	Fehler! Textmarke nicht definiert.

PerlS.....	Fehler! Textmarke nicht definiert.
PGP.....	Fehler! Textmarke nicht definiert.
Plug-In.....	Fehler! Textmarke nicht definiert.
PNG.....	Fehler! Textmarke nicht definiert.
Point to Point Protocol.....	Fehler! Textmarke nicht definiert.
Port des Servers.....	Fehler! Textmarke nicht definiert.
PostScript.....	Fehler! Textmarke nicht definiert.
PPP.....	Fehler! Textmarke nicht definiert.
Pretty Good Privacy.....	Fehler! Textmarke nicht definiert.
Programmiersprache.....	Fehler! Textmarke nicht definiert.
Programmverzweigung.....	Fehler! Textmarke nicht definiert.
Programmzeile.....	Fehler! Textmarke nicht definiert.
Protocol Spoofing.....	Fehler! Textmarke nicht definiert.
Provider.....	Fehler! Textmarke nicht definiert.
Proxy-Server.....	Fehler! Textmarke nicht definiert.
Prozedur.....	Fehler! Textmarke nicht definiert.
Real Audio.....	Fehler! Textmarke nicht definiert.
relativer Hyperlink.....	Fehler! Textmarke nicht definiert.
Rich-Content-eMail.....	Fehler! Textmarke nicht definiert.
Robot.....	Fehler! Textmarke nicht definiert.
Rocket-eBook.....	Fehler! Textmarke nicht definiert.
Router.....	Fehler! Textmarke nicht definiert.
Runtime-Error.....	Fehler! Textmarke nicht definiert.
Schlüsselwort.....	Fehler! Textmarke nicht definiert.
Schneckenpost.....	Fehler! Textmarke nicht definiert.
Schrift.....	Fehler! Textmarke nicht definiert.
Second-Level-Domain.....	Fehler! Textmarke nicht definiert.
Secure Hypertext Transfer Protocol.....	Fehler! Textmarke nicht definiert.
Seitenbeschreibungssprache.....	Fehler! Textmarke nicht definiert.
Serial Line Internet Protocol.....	Fehler! Textmarke nicht definiert.
Server Side Includes.....	Fehler! Textmarke nicht definiert.
Servlet.....	Fehler! Textmarke nicht definiert.
SGML.....	Fehler! Textmarke nicht definiert.
Shell Account.....	Fehler! Textmarke nicht definiert.
Site.....	Fehler! Textmarke nicht definiert.
SLIP.....	Fehler! Textmarke nicht definiert.
snail.....	Fehler! Textmarke nicht definiert.
String.....	Fehler! Textmarke nicht definiert.
Sub-Level-Domain.....	Fehler! Textmarke nicht definiert.
Tag.....	Fehler! Textmarke nicht definiert.
TCP/IP.....	Fehler! Textmarke nicht definiert.
T-DSL.....	Fehler! Textmarke nicht definiert.
Telnet.....	Fehler! Textmarke nicht definiert.
Textdatei.....	Fehler! Textmarke nicht definiert.
Third-Level-Domain.....	Fehler! Textmarke nicht definiert.
T-Online.....	Fehler! Textmarke nicht definiert.
Top-Level-Domain.....	Fehler! Textmarke nicht definiert.
true.....	Fehler! Textmarke nicht definiert.
UND-Verknüpfung.....	Fehler! Textmarke nicht definiert.
Unicode.....	Fehler! Textmarke nicht definiert.
Universal Resource Locator.....	Fehler! Textmarke nicht definiert.
Upload.....	Fehler! Textmarke nicht definiert.
Variable.....	Fehler! Textmarke nicht definiert.
Variablendeklaration.....	Fehler! Textmarke nicht definiert.
Vergleichsoperator.....	Fehler! Textmarke nicht definiert.
Verzweigung.....	Fehler! Textmarke nicht definiert.
virtuelle Realität.....	Fehler! Textmarke nicht definiert.
VRML.....	Fehler! Textmarke nicht definiert.
W3-Consortium.....	Fehler! Textmarke nicht definiert.
wahr.....	Fehler! Textmarke nicht definiert.
Web-Browser.....	Fehler! Textmarke nicht definiert.
Web-Master.....	Fehler! Textmarke nicht definiert.
Web-Server.....	Fehler! Textmarke nicht definiert.
Wertzuweisung.....	Fehler! Textmarke nicht definiert.
Western.....	Fehler! Textmarke nicht definiert.
World Wide Web.....	Fehler! Textmarke nicht definiert.
WWW-Browser.....	Fehler! Textmarke nicht definiert.
WWW-Server.....	Fehler! Textmarke nicht definiert.
XML.....	Fehler! Textmarke nicht definiert.
Zeichensatz.....	Fehler! Textmarke nicht definiert.
Zuweisungsoperator.....	Fehler! Textmarke nicht definiert.
Lösungen.....	Fehler! Textmarke nicht definiert.

HuT

Copyright

Copyright 2000: Handel und Technik GbR (HuT). Dieses Werk ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Reproduktion und Kopieren, auch von Einzelteilen, nur mit schriftlicher Genehmigung von HuT.

Für jedes elektronisches Buch, das lizenziert wurde, gilt:

- Der Lizenznehmer ist berechtigt, die Lizenzversion auf beliebig vielen Computern zu installieren. Es darf jedoch zu einem Zeitpunkt immer nur eine Installation verwendet werden.
- Die Weitergabe des auf den Lizenznehmer ausgestellten Passwortes ist nicht erlaubt.
- Die Lizenz beinhaltet die Genehmigung, diese Publikation einmal auszudrucken. Ein erneuter Ausdruck ist nur zulässig, wenn der neue Ausdruck den alten Ausdruck ersetzt. Eine Verfielfältigung und Weitergabe an Dritte ist nicht erlaubt.
- Alle hier nicht genannten Nutzungs- und Verbreitungsmöglichkeiten sind untersagt.

Haftung

Diese Publikation wurde nach bestem Gewissen erstellt und mit Sorgfalt getestet. Sie erhebt jedoch keinen Anspruch auf Vollständigkeit oder Fehlerfreiheit. Für Fehler, die auf falsche oder falsch verstandene Beschreibungen in dieser Publikation zurückzuführen sind, übernehmen HuT und der Autor keine Haftung. Die gesamte Haftung besteht nach Wahl von HuT entweder in der Rückerstattung des bezahlten Kaufpreises oder in der Reparatur oder Ersatz der Software. HuT schließt für sich jede weitere Gewährleistung aus und ist nicht ersatzpflichtig für irgendwelche Schäden, die aufgrund der Benutzung der Software entstehen. Das gilt ebenfalls uneingeschränkt für Schäden aus entgangenem Gewinn, finanziellem Verlust jeder Art, Betriebsunterbrechungen, Verlust von Informationen oder Daten.

Die in dieser Publikation beschriebenen Verfahren und Programme werden ohne Rücksicht auf einen eventuellen Patentschutz mitgeteilt. Die in der Publikation genannten Hard- und Softwarebezeichnungen sowie die Markennamen unterliegen dem Marken-, Waren- oder Patentschutz der jeweiligen Firmen.

Die Wiedergabe von Warenbezeichnungen, Gebrauchs- und Handelsnamen etc. in dieser Publikation berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Waren- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen.

HuT ist nicht verantwortlich für den Inhalt fremder Seiten.

Impressum

Handel und Technik GbR
Wolfgang Henderkes (verantwortlich)
Robert-Bosch-Str. 2
59439 Holzwickede
Telefon: 0 23 01 / 91 74 03
Telefax: 0 23 01 / 91 71 00
eMail: wolfgang.henderkes@hut-online.de
<http://www.hut-online.de>

Wenn es Sie interessiert

Über dieses (e)Buch

Der „JavaScript Tutor“ ist eine Einführung in die Programmiersprache JavaScript, also keine umfassende Beschreibung aller Möglichkeiten und keine vollständige Referenz. Entsprechende weiterführende Publikationen sind bis Mitte des Sommers 2000 separat erhältlich.

Die Zielgruppe dieser Publikation ist der in der Programmierung unerfahrenen Web-Designer, der sich mit der Tatsache konfrontiert sieht, an JavaScript nicht vorbeizukommen. Entsprechend vorsichtig wird der Anfänger an die Programmierung und in diesem konkreten Fall an JavaScript herangeführt.

Der Tutor ist in Kapitel, Kapitelabschnitte und Seiten unterteilt und sollte vom Anfang beginnend Seite für Seite gelesen werden – unter der Voraussetzung, dass Sie keine Programmiererfahrung haben und diesen Tutor zum ersten Male lesen. Die Beispiele in dem Tutor bauen aufeinander auf und nehmen an Komplexität zu. Bei einem erneuten Durcharbeiten des Tutors können Sie auch auf die einzelnen Kapitel zugreifen, um Details zu vertiefen.

Wenn Sie mit dem Internet bereits viele Erfahrungen haben, also Ihren Browser ebenso gut kennen wie HTML und die Erweiterungen mit den Script-Sprachen, dann können Sie aus dem Kapitel „Einführung“ die drei Kapitelabschnitte „Die Sprachen des Internet“, „Werkzeuge“ und „HTML und JavaScript“ überfliegen.

Um gerade für den absoluten Anfänger den Einstieg erträglich zu machen, habe ich sehr viel Zeit darauf verwendet, den Text immer wieder zu überarbeiten – mit dem Ziel, möglichst kurze Sätze mit scharfen Formulierungen zu finden. (Der letzte Satz ist garantiert einer der längsten!) Anders als bei normaler Prosa sind häufig verwendete Begriffe nicht durch Synonyme ersetzt, sondern werden wiederholt, um die Eindeutigkeit sicherzustellen. Die Fachbegriffe, die dann noch verbleiben, sind hoffentlich alle und umfassend in dem Lexikon erklärt.

Auch die direkte Anrede (Tun Sie dies oder jenes ...) verwende ich nur dort, wo es um konkrete Übungen geht. Ansonsten schwebe ich im Passiv.

Dass der Umfang trotzdem über 250 Seiten (DIN A4) liegt, hat mit der Komplexität des Themas zu tun und mit den zahlreichen Beispielen, die jeweils als funktionsfähige HTML-Seite zur Verfügung stehen.

Über dem Beispiel steht jeweils ein Link zur Verfügung, der das Beispiel selbst direkt als HTML-Seite startet.

Alle Beispiele sind an dem hellblauen Rahmen zu erkennen.

Die Übungen sind dagegen grau eingerahmt.

Konzeptionell besteht der „JavaScript Tutor“ aus zwei Teilen:

- einem Tutor mit dem eigentlichen Thema
- einem Lexikon mit den wichtigsten Fachbegriffen

Gute Bücher über JavaScript gibt es schon einige, warum noch ein weiteres? Ein(fach)e Frage, viele Antworten:

1. Fachbücher in elektronischer Form (kurz eBücher) sind noch die Ausnahme.
2. Dieses elektronische Buch gibt es im HTML-Format („JavaScript Tutor - inter@ktiv!“) und als eBuch im PDF-Format („JavaScript Tutor“) und nutzt jeweils die Möglichkeiten der beiden Plattformen. Als HTML-Dokument ist der „JavaScript Tutor - inter@ktiv!“ durch die integrierten Fragen & Antworten ein richtiges interaktives Lernprogramm zum Selbststudium. In dem PDF-Dokument fehlt diese Möglichkeit, dafür hat der Leser die Wahl, das eBuch (komplett) auszudrucken.
3. Neben dem „JavaScript Tutor“ gibt es als separate Publikationen das „JavaScript Referenzhandbuch“ und das „JavaScript Kochbuch“. Damit sollte das Thema JavaScript umfassend behandelt sein.

Vorteil HTML-Format

Das PDF-Format ist kein ungeliebtes Kind und ein HTML-Dokument ist mitnichten das Schönste, was ich mir so vorstellen kann – aber die Welt hat es so gewollt. Da HTML nicht nur ein offener Standard ist, sondern von vielen Organisationen und Firmen weiterentwickelt wird, werden in Zukunft alle elektronischen Bücher von HuT für diese Technik konzipiert.

Der Tutor und das Lexikon verfügen (als eBuch im HTML-Format) jeweils über ein Inhaltsverzeichnis und eine Navigationsleiste zum Blättern der Seiten. Die beiden Anwendungen verhalten sich wie ein gedrucktes Buch, verfügen jedoch über eine Volltextsuche. Der Tutor ist themenbezogen aufgebaut, einige Begriffe sind mit einem [Hyperlink](#) ausgezeichnet, die bei einer Ausführung eine eigene Seite (das Lexikon) mit der Begriffserklärung anzeigen.

Bei der Arbeit (mit dem eBuch im HTML-Format) ist es empfehlenswert, den Tutor und das Lexikon parallel zu starten.

Fazit

Damit sind gleichzeitig einige Unterschiede zwischen den Versionen erklärt – oder anders ausgedrückt:

- HTML-Dokumente enthalten das volle Spektrum der Möglichkeiten.
- PDF-Dokumente enthalten keine „Fragen & Antworten“ als interaktives Quiz. Tutor und Lexikon sind nicht als Datei getrennt.

Wolfgang Henderkes

(Januar 2001)

Voraussetzungen an den Leser

Bei dem „JavaScript Tutor“ handelt es sich um eine Einführung in diese Programmiersprache zum Selbststudium. Der Schwerpunkt liegt ergo auf der Erstellung von HTML-Seiten mit JavaScript-Programmen.

Als Leser kommen alle Web-Designer und technisch Interessierten in Frage, die sich in HTML gut auskennen. Sehr gute Kenntnisse über die Bedienung Ihres Computers und des verwendeten Browsers sind unbedingt erforderlich. Bis auf einige Tipps für eine effektive Arbeit werden keine Funktionen oder Einstellungen der Browser erklärt. Programmierkenntnisse werden nicht vorausgesetzt: Der „JavaScript Tutor“ richtet sich an den Anfänger, der keine oder wenige Erfahrungen mit Programmiersprachen hat.

JavaScript wird gelegentlich als eine leicht zu erlernende Programmiersprache bezeichnet. Sicher, wer Motorrad fahren kann, wird mit dem Fahrrad keine Probleme haben. So braucht sich JavaScript nicht um die Speicherverwaltung oder um Dateioperationen zu kümmern, sondern basiert auf einer HTML-Seite, die vom Browser aufgerufen und kontrolliert wird. Der Anfänger hat häufig erhebliche Probleme, die auf die abstrakte „Denkweise“ in der Programmierung zurückzuführen sind. Tatsache ist, dass JavaScript nicht über alle Möglichkeiten anderer Programmiersprachen wie z.B. JAVA oder C++ verfügt und in diesem Sinne einfacher ist. Für den Anfänger sind die ersten Schritte gleich schwierig. Diejenigen, die bereits einige Erfahrung mit der Programmierung haben, werden JavaScript tatsächlich leichter erlernen.

Den mitgelieferten Code können und sollen Sie als Basis für Ihre Arbeit benutzen. Getesteter Code vermeidet Tippfehler beim Erstellen und liefert eine funktionsfähige Basis. Ein guter Programmierer kann vielleicht den Code aus dem Kopf herschreiben – ein effizienter Programmierer lebt von seinen selbst erstellten und gut organisierten Beispielen. Ist der Code dann noch gut dokumentiert, sollte der Rest ebenfalls gelingen.

Die Beispiele sind entweder im Tutor mit dem kompletten Code aufgeführt, oder in der Beispielsammlung als fertige HTML-Seite zu finden. Wenn Sie Probleme haben sollten, finden Sie dort immer das korrekte Ergebnis.

Mit dem vermittelten Wissen werden Sie in der Lage sein, auch komplexe JavaScript-Programme zu erzeugen und in die HTML-Seiten für das World Wide Web einzubinden.

Technische Voraussetzungen

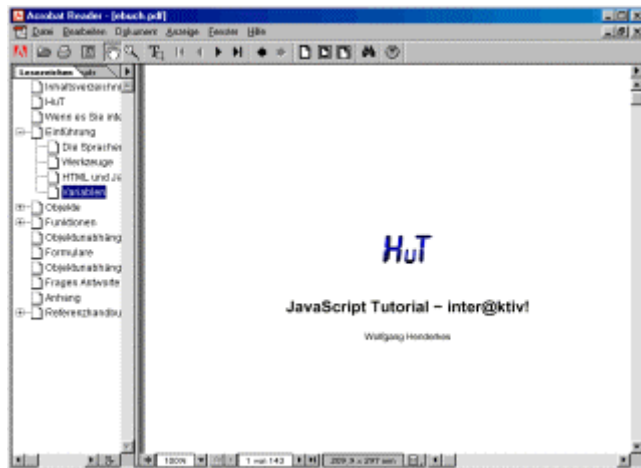
Für die Darstellung des eBooks im HTML-Format und zur Umsetzung der entsprechenden Beispiele benötigen Sie einen Browser der 4. Generation – entweder den MS Internet Explorer oder Netscape Navigator, Opera bis zur Version 4.0 läuft nicht fehlerfrei. Der Browser muss für die Verarbeitung von JavaScript und die Akzeptanz von Cookies eingestellt sein. Grundsätzlich kann jeder andere Browser benutzt werden, der mit vergleichbaren Funktionen ausgestattet ist.

Weiter sollten Sie für Ihre Beispiele einen ASCII-[Editor](#) (wie das Notepad unter MS Windows) verwenden.

Wenn Sie das Dokument im PDF-Format lesen, benötigen Sie dem Browser noch der Acrobat Reader. Der Browser (für HTML-Dokumente) bzw. der Acrobat Reader (für PDF-Dokumente) müssen auf Ihrem Computer installiert sein.



Das eBook im Browser.



Der Adobe Acrobat Reader.



Bei der Arbeit mit dem eBook im HTML-Format ist es zu empfehlen, mehrere Browser zu öffnen. So können Sie in dem ersten Fenster den Tutor, in einem zweiten Fenster das Lexikon anzeigen und in einem weiteren Ihre HTML-Seiten erstellen. Quelltexte aus den Beispielen können Sie über die Zwischenablage Ihres Computers direkt in den ASCII-Editor kopieren. Dann brauchen Sie den Programmcode nur noch anzupassen und vermeiden Tippfehler.

Hardware – nichts besonderes

Ihr Computer sollte wenigstens 256 Farben darstellen können. Sonst erfüllt jeder Computer, der seit Mitte der 90er Jahre verfügbar ist, die notwendigen Voraussetzungen.

Konventionen

Die wichtigsten Schreibregeln und Symbole:

-  Dieses Symbol bedeutet, dass Sie etwas eingeben oder ausführen sollen.
-  Dieses Symbol kennzeichnet die Reaktion des Computers.
- Namen von grafischen Schaltflächen werden als **VERSALIEN** dargestellt.
- Menüpunkte, Dateinamen, Eigennamen, Feldbezeichnungen wie beschriftete Schaltflächen werden in »Klammern« dargestellt.
- Hyperlinks werden in dem eBuch im HTML-Format farbig und im PDF-Format farbig unterstrichen dargestellt. Ein Link mit fetter Schrift zeigt eine Seite im Lexikon an, ein Link mit normaler Schrift verzweigt innerhalb des Tutors. Ein Link zu einer Seite im Internet beginnt mit „www“ oder „ftp“.
- Die "Anführungszeichen" innerhalb der Tags sind Bestandteil der HTML-Befehle und müssen mit eingeben werden.
- Typografische „Anführungszeichen“ im laufenden Text werden zur Hervorhebung benutzt.
- Sofern nicht anders angegeben, wird die linke Maustaste zur Bedienung benutzt. Die rechte Maustaste dient zur Einstellung von Eigenschaften. Wird die rechte Maustaste verwendet, ist das ausdrücklich kommentiert.

Einführung

Die Sprachen des Internet

Überblick

Bevor sich der Unbedarfte auf das Abenteuer Programmieren einlässt, kann ein kleiner Überblick über die Summe der zu erwartenden Sorgen nicht schaden. Als kleinster gemeinsame Nenner des Internet gilt HTML zur Darstellung von Texten, Bildern und Hyperlinks. HTML-gerecht aufbereitete Informationen werden in Seiten organisiert, die anders als ihre gedruckten Pendanten keine begrenzte Länge haben. Zur Anzeige von HTML-Seiten wird eine Web-Browser (oder kurz Browser) genannte Software verwendet.

Die erste Generation der Browser kann Texte, Bilder und Hyperlinks darstellen, ist aber nicht in der Lage, Programme selbständig auszuführen. Nur mit einer Online-Verbindung lassen sich Programme auf dem Web-Server ausführen. Die Eingaben des Anwenders werden zunächst an den Server gesendet, der ein Programm ausführt und das Ergebnis wieder zurück an den Browser schickt.

Da diese Vorgehensweise in vielen Fällen umständlich und zeitraubend ist, entsteht der Wunsch, Programme ebenfalls auf dem Computer des Anwenders auszuführen. Die Lösung dazu lautet JavaScript. Mit der nächsten Generation der Browser folgt eben diese Fähigkeit, Programme unabhängig von einer Online-Verbindung auszuführen. Bei diesen Programmen handelt es sich um sogenannte Script-Programme oder kurz Scripts. Ein Script ist ein Programm, das wie jedes andere Programm aus einer Folge von Anweisungen besteht. Ein Script wird zeilenweise gelesen und interpretiert, in diesem Fall durch den Browser, in den ein entsprechender Interpreter (auch Script-Modul genannt) integriert ist. In den aktuellen Browsern von Microsoft und Netscape ist dieser Interpreter vorhanden.

Programme in JavaScript werden durch den Browser bei dem Aufruf des Programms ausgeführt. Der Aufruf kann bei dem Start der HTML-Seite erfolgen oder durch eine [Funktion](#) auf der HTML-Seite (z.B. durch eine Schaltfläche in einem Formular). Der Vorteil von JavaScript gegenüber einer zu kompilierenden Programmiersprache (als Alternative zu einer Script-Sprache) liegt in der einfachen Pflege. Jede Änderung ist sofort lauffähig. Außerdem muss sich der Programmierer nicht um die

Anpassung (einen Compiler) für die gewünschten Zielplattformen kümmern. Da JavaScript-Programme von dem Browser ausgeführt werden, laufen sie auf jedem Computer und mit jedem Betriebssystem, für das es einen geeigneten Browser gibt.

HTML

HTML ist keine Programmiersprache, also auch kein Script, sondern eine Seitenbeschreibungssprache. Das klingt nach Haarspalterei, ist aber ungefähr so, als wenn Sie einen Sportwagen mit einem Tretroller vergleichen.

Bevor die Ära der persönlichen Computer beginnt, gibt es fast nur Software, die für einen bestimmten Nutzer und einen bestimmten Zweck bestimmt ist. Für die Daten, die mit dieser Software verwaltet werden, benutzen die Programmhersteller eigene Dateiformate.

Mit dem PC – und natürlich dem Mac – kommt die Standardsoftware. Zwar hat jedes dieser Softwareprodukte ein eigenes Dateiformat, doch können Daten mit anderen Interessierten ausgetauscht werden, sofern sie die gleiche Software verwenden. Typische Vertreter sind Textverarbeitungs-, DTP- und Grafikprogramme, Datenbanken und Tabellenkalkulationen.

Im Internet mit den zahlreichen unterschiedlichen Computersystemen und Betriebssystemen ist ein gemeinsamer, unabhängiger Standard Voraussetzung für einen Datenaustausch und damit für eine grenzenlose Kommunikation. Mit HTML ist das Fundament geschaffen, auf dem alle stehen können. HTML-Seiten bestehen aus ASCII-Text. Deshalb können HTML-Seiten mit jedem einfachen Texteditor wie Notepad unter MS Windows bearbeitet werden. Das ist allerdings die Steinzeitmethode. Besser sind Programme, welche die HTML-Seiten schon bei der Erstellung anzeigen und eine grafische Oberfläche zur Bedienung besitzen, sogenannte HTML-Editoren.

HTML ist die Basis, nicht das Ziel. HTML hat viele Einschränkungen, aber ebenso den wichtigsten aller Vorteile. Es wird von dem herstellerunabhängigen [W3-Consortium](#) gepflegt. Alle andere Software dient zuerst den Interessen ihrer Hersteller. Meister in dieser Disziplin ist Microsoft. Das Softwarehaus schafft es z.B. mit jeder Version ihrer Office-Software, ein neues Dateiformat einzuführen, das von den älteren Version natürlich nicht gelesen werden kann. Zwar können in der Regel ältere Dateiformate in die neue Software geladen werden, allerdings sind damit zwei wesentliche Punkte verbunden:

- Das neue Softwareprodukt ist erfolgreich an den Mann (die Frau) gebracht.
- Die Abhängigkeit von diesem Softwareprodukt wird beibehalten.

Neben der Einarbeitung in immer neue Funktionen und geänderte Oberflächen ist keineswegs garantiert, dass alte Dateiformate in zukünftigen Versionen fehlerfrei gelesen werden können. Fast jeder, der große Datenbestände pflegt, hat schon mal von den Nachteilen proprietärer Software erfahren.

Hat der Computer irgendwann das Zeitliche gesegnet und wird ersetzt durch ein neues Modell, stehen die Chancen gut, dass die alte Software auf dem neuen Computer gar nicht mehr läuft. Hier bietet

eine standardisierte Sprache wie HTML eine echte Alternative. Sie ist plattform- und softwareunabhängig. Der Grund liegt in der Klartextform. HTML enthält keine proprietären Steuerzeichen und keine Dateibereiche, die binär interpretiert werden müssen. Dieses Klartextformat ist so gestaltet, dass Computer und Menschen mit ihren Befehlen klarkommen können.

HTML entwickelt sich natürlich weiter – der Nachfolger wird voraussichtlich XML heißen – und die Wahrscheinlichkeit, dass heutige Computer Ende des nächsten Jahrtausends noch mit HTML arbeiten, ist eher gering. Doch wer heute auf HTML und JavaScript aufbaut, wird es in Zukunft leichter haben, auf neue Versionen umzusteigen.

Aufgabe von HTML

Die Hypertext Markup Language oder kurz HTML ist eine Dokumentenbeschreibungssprache, die zum internationalen, standardisierten Dokumentenaustausch benutzt wird. HTML ist zur Sprache für Dokumente im World Wide Web (WWW) geworden, das vielen zum Synonym für das [Internet](#) geworden ist.

HTML legt die Struktur eines Dokumentes in einer einheitlichen Form fest und benutzt dazu eine hierarchische Ordnung. Die Strukturierung wird durch Elemente vorgenommen, zu denen Kapitel, Unterkapitel, Textabsätze, Listen, Tabellen, Grafiken und Hyperlinks (Querverweise) zu anderen Dokumenten zählen. Diese Elemente werden als logische Elemente des Dokumentes bezeichnet.

Das ursprüngliche Ziel, der Austausch von wissenschaftlichen Dokumenten, wird damit erreicht – schön zu lesen und zu betrachten sind solche Dokumente nicht. Mit der Verbreitung des Internet (oder mit seiner Kommerzialisierung) werden weitere Anforderungen an HTML gestellt. Deshalb beschreibt HTML heute neben den logischen Elementen eines Dokumentes ebenso physische Elemente wie z.B. Textformatierungen (fett, kursiv etc.) und Formatvorlagen.

Bis auf wenige Ausnahmen hat jedes Element einen definierten Anfang und ein definiertes Ende. Anfang und Ende werden durch entsprechende Befehle gekennzeichnet, die in spitze Klammern `<>` gesetzt werden – die sogenannten [HTML-Tags](#) oder einfach nur Tag. (Die Ausnahmen sind Tags, die nur einen Anfang benötigen, aber kein Ende.)

Anders ausgedrückt werden im Rahmen von HTML verschiedene Arten von Tags unterschieden:

- Diejenigen, mit denen angegeben wird, welchen Platz ein Element innerhalb der Struktur des Dokumentes hat, werden logische Tags genannt. In `<h1>Zeichenfolge</h1>` bedeuten die Tags, dass „Zeichenfolge“ eine Überschrift erster Stufe ist.
- Weiterhin gibt es physische Tags, mit denen Formatierungsanweisungen gegeben werden. In `Zeichenfolge` bedeuten die Tags, dass „Zeichenfolge“ im Fettdruck dargestellt werden soll.
- In einem Beispiel wie `<autor>Wolfgang Henderkes</autor>` wird nichts über die Dokumentenstruktur gesagt und es wird keine Formatierungsanweisung gegeben. Diese Tags können

als semantische Tags bezeichnet werden, denn sie sagen etwas über die Bedeutung der Zeichenfolge aus, die von ihnen eingeschlossen wird.

Aufbau von HTML

HTML-Seiten bestehen aus reinem ASCII-Text und können mit jedem ASCII-Editor gelesen und bearbeitet werden. Das bedeutet für den Autor einer WWW-Seite, dass er aus technischer Sicht kaum eine Möglichkeit hat, den Inhalt und das Layout zu schützen.

Das Grundgerüst einer leeren Seite besteht aus dem Kopfbereich `<head> . . . </head>` und einem Körperbereich `<body> . . . </body>`.

Die Elemente stehen im Körperbereich. Ein Element mit einem Textabsatz enthält z.B. eine als fett markierte Textstelle, ein Element mit einer Liste besteht aus mehreren Punkten, ein Element mit einer Tabelle besteht aus mehreren Zellen etc. Elemente können in Unterelemente unterteilt sein – eine Liste kann z.B. Tabellen beinhalten etc.

Jede HTML-Seite kann erweitert werden mit:

- [JavaScript](#)
- [Plug-In\(s\)](#)
- [JAVA](#)
- Programmen, die über das [Common Gateway Interface](#) (CGI-Schnittstelle) aufgerufen werden

Ein wichtiger Punkt für interessante Seiten sind animierte Grafiken, wie sie mit dem [GIF](#)-Format möglich sind. Grafiken, Bilder, Audio- und Videodateien, die auf der HTML-Seite zu sehen und zu hören sind, werden als Verweis auf die Originaldatei eingebunden.

WWW-Browser, die HTML-Seiten am Bildschirm anzeigen, interpretieren die Tags und stellen die Elemente dann als Seite auf dem Bildschirm dar. HTML-Seiten werden innerhalb des WWW über das HTTP-Protokoll übertragen.

JavaScript

JavaScript ist eine Erfindung von Netscape und die am meisten verwendete Script-Sprache in Kombination mit HTML-Seiten. JavaScript hat Ähnlichkeiten mit der objektorientierten Programmiersprache JAVA, aber einen geringeren Befehlsumfang. Übernommen ist z.B. die Syntax der Befehle. Das Konzept der objektorientierten Programmierung findet sich in JavaScript nur teilweise wieder. Nicht nur deswegen gilt: JAVA ist nicht JavaScript. JavaScript wird deswegen als objektbasierte (und nicht als objektorientierte) Programmiersprache bezeichnet.

JavaScript ist trotzdem keine Kleinausgabe einer anderen Sprache (auch nicht von JAVA), hat jedoch Einschränkungen. In JavaScript können keine selbständigen Programme erstellt werden, außerdem gibt es nur begrenzte Funktionen zum Lesen und Schreiben von Daten. JavaScript-Programme können nur bei vorhandenem Interpreter ausgeführt werden, entweder über einen Web-Server oder einen Web-Browser.

JavaScript-Programme werden zur Laufzeit von dem Browser zeilenweise interpretiert und ausgeführt – vorausgesetzt, der Browser kann JavaScript interpretieren und ist für die Verarbeitung von JavaScript-Programmen freigegeben. Deswegen kann jeder Computer mit einem beliebigen Betriebssystem JavaScript-Programme ausführen – sofern ein entsprechender Browser für diese Plattform verfügbar ist.

JavaScript wird in eine HTML-Seite integriert, um die Seite mit zusätzlichen Eigenschaften auszustatten. Das kann ein Hyperlink sein, der seine Farbe ändert, sobald der Mauszeiger darüber fährt, oder eine Berechnung, die nach Eingabe von Daten durch Klicken auf eine Schaltfläche gestartet wird. Zur Ausführung gelangt der JavaScript-Code, indem eine JavaScript-[Funktion](#) mit einem Ereignis der HTML-Seite verknüpft wird – z.B. bei der Betätigung der Schaltfläche oder direkt beim Start der HTML-Seite. Ein JavaScript-Programm wird mit dem `<script>`-Tag in eine HTML-Seite eingebunden. Das folgende Beispiel zeigt ein Programm, das bei dem Start der HTML-Seite automatisch ausgeführt wird und ein Meldfenster öffnet mit dem Text „Herzlich Willkommen!“.

Beispiel

```
<script language="JavaScript">
<!--
alert("Herzlich Willkommen!");
// -->
</script>
```

JavaScript eignet sich nicht für alle Aufgaben, die mit einer normalen Programmiersprache ausgeführt werden können. Dafür ist der Befehlsumfang zu klein, außerdem sorgt die zeilenweise Interpretation für eine geruhsame Arbeitsweise. Aber für die Erstellung von (Inter)-Aktionen z.B. im Zusammenhang mit Formularen ist JavaScript sehr gut geeignet.

Für die Erstellung eines Warenkatalogs mit Bestellmöglichkeit (WebShop) reicht JavaScript noch aus, stößt aber an seine Grenzen. In der Regel sind zusätzliche Programme erforderlich, die über die sogenannte CGI-Schnittstelle des Servers in JavaScript eingebunden werden. Diese Programme sind häufig in der Sprache Perl geschrieben. Grundsätzlich können mit einigen anderen Programmiersprachen entsprechende ausführbare Programme erstellt werden – in der Windows-Welt sind das Programme mit der Endung „.exe“.

JavaScript läuft innerhalb des Browsers und kann im Normalfall nur auf die Objekte zugreifen, die der Browser momentan im Speicher hält. JavaScript kann:

- keine Informationen in beliebigen Dateien auf dem Computer des Benutzers oder dem Server abspeichern, mit Ausnahme von [Cookies](#) und von [ASP](#) (Active Server Pages)
- HTML-Seiten dynamisch generieren, diese aber nicht speichern
- nicht auf Peripheriegeräte zugreifen
- keine dritten Programme ansprechen oder manipulieren mit Ausnahme der ASP

JavaScript-Programme sind wie HTML in ASCII-Text geschrieben und für jeden lesbar. Der Code lässt sich also technisch ebenso wenig schützen wie der HTML-Code.

Die Sprachelemente in JavaScript lassen sich unterteilen in:

- Variablen: Speichern Werte, die aus Zahlen und Text (Zeichenketten) bestehen.
- Operatoren: Führen Berechnungen mit Werten aus.
- Objekte: Verfügen über Eigenschaften und Methoden.
- Bedingte Anweisungen: Anweisungen werden nur unter bestimmten Bedingungen ausgeführt.
- Schleifen: Anweisungen werden mehrmals hintereinander ausgeführt.
- Objektunabhängige Funktionen: Funktionen mit definierten Eigenschaften, die selbst programmiert werden müssen.
- Event-Handler: JavaScript-Anweisungen als HTML-Attribute, um auf Ereignisse (z.B. einen Mausklick) zu reagieren.

Mit JavaScript werden interaktive HTML-Seiten möglich – auf einfachere Art und Weise, als das z.B. mit Perl über das Common Gateway Interface möglich wäre. Wichtiger Unterschied zu CGI-Programmen: JavaScript kann ohne Zugriff auf einen Web-Server eingesetzt werden.

Weil viele Scripts (also Programme) jetzt beim Anwender laufen und damit die Zeit der Datenübertragung und der Berechnung durch den Server ersparen, wird bei einer Online-Verbindung mehr Unabhängigkeit des Browsers vom Server erreicht. Der Browser selbst läuft aber ebenso offline

(also ohne Verbindung zum Internet) und eignet sich als Oberfläche für die Bedienung des gesamten Computers.

JavaScript ist zunächst nur in dem eigenen Browser, dem Netscape Navigator, integriert. Mit der wachsenden Bedeutung des Internet und der Verbreitung des Netscape Navigator erkennt Microsoft die Bedrohung für das eigene Betriebssystem. Als Folge davon kommt Microsoft mit einem konkurrierenden Browser, dem Internet Explorer. Seit der Version 3 wird darin JavaScript unterstützt, aber nicht mit allen Möglichkeiten. Das bedeutet konkret, dass nicht alle Befehle (z.B. der print-Befehl für das Drucken) ausgeführt werden. Bei jeder neuen Technik, welche das Monopol von MS Windows bedrohen könnte, besteht die Taktik von Microsoft darin, eben diese Technik nur teilweise zu unterstützen.

Es ist üblich bei Microsoft, eine vorhandene Technik oder einen Standard nicht nur zu „übernehmen“, sondern zu erweitern und damit weiter zu verwässern, um damit den Anwender zu verunsichern. Microsoft erreicht das, indem zusätzliche Funktionen eingeführt werden, die nur der Internet Explorer unterstützt, wie z.B. die Laufschrift durch das `<marquee>`-Tag.

Durch den gemeinsamen Standard ECMA 262-Script soll(t)en die Streitigkeiten um die Kompatibilität der Browser für die Ausführung von JavaScript behoben werden. ECMA 262-Script ist eine Teilmenge von JavaScript 1.2 und stellt den gemeinsamen Nenner dar, der von allen Browsern korrekt ausgeführt werden sollte. Die ECMA ist ein Zusammenschluss von Firmen zur Erstellung von Normen.

In diesem Tutor sind alle Objekte, Eigenschaften, Methoden und Funktionen nach ECMA 262 aufgeführt. Die eigenmächtigen Erweiterungen von JavaScript werden nicht berücksichtigt.

JScript

JScript ist die Microsoft-Implementierung der Sprachbestimmung ECMA 262. Es handelt sich um eine vollständige Implementierung zusätzlich einiger Erweiterungen, die nur auf dem Microsoft Internet Explorer laufen. Mit dieser Erweiterung wird der Sinn der Standardisierung natürlich wieder unterlaufen.

JScript wird ebenso wie JavaScript in die HTML-Seite eingebunden.

Beispiel

```
<script language="JScript">
<!--
alert("Herzlich Willkommen!");
// -->
</script>
```

Der Code wird von dem Microsoft Internet Explorer ausgeführt und von dem Netscape Navigator ohne Fehlermeldung ignoriert.

Sofern der Code mit der ECMA 262 kompatibel ist, kann er ebenso gut als JavaScript deklariert werden. Wer sich nicht ausschließlich und für alle Zeit in der Microsoft-Welt bewegt, sollte JScript nicht verwenden.

VBScript

VBScript ist eine Eigenentwicklung von Microsoft. Die Script-Sprache lehnt sich an Visual Basic for Applications an, die ebenfalls in den Office-Applikationen verwendet wird, besitzt aber ein paar Einschränkungen.

VBScript gelangt genau wie JavaScript und JScript durch den Browser zum Einsatz. JavaScript und VBScript sind hinsichtlich ihrer Sprachelemente ähnlich, die Syntax der generierten Programme unterscheidet sich jedoch erheblich. VBScript arbeitet nur mit dem Internet Explorer. Eine Browser-unabhängige Programmentwicklung ist mit VBScript daher nicht möglich.

Auch hier gilt: Wer sich nicht ausschließlich und für alle Zeit in der Microsoft-Welt bewegt, sollte VBScript nicht verwenden.

XML

XML (eXtensible Markup Language) ist eine vom W3C herausgegebene Empfehlung für ein Dateiformat, das es ermöglicht, elektronische Dokumente im Internet zu erstellen. Mit anderen Worten ist XML eine standardisierte Dokumentenbeschreibungssprache, die HTML in Zukunft ablösen soll. XML ist keine Programmiersprache. XML ist erweiterbar, plattform-, und sprachenunabhängig. Sowohl HTML und XML basieren auf [SGML](#). Der wesentliche Unterschied: Während HTML nur die Methoden für die Darstellung von Dokumenten definiert, liefert XML die Methoden für das Arbeiten mit Daten und ihren Komponenten.

Vereinfacht ausgedrückt geht es bei XML darum, eine neue Sorte von Tags einzuführen. Neu sind diese Tags zumindest für diejenigen, die bisher nur [HTML-Tags](#) kennen. Wer sich mit SGML auskennt, bewegt sich beim Umgang mit XML auf bekanntem Terrain – denn XML ist ein Untermenge von SGML.

Mit der einfachen Struktur von HTML können viele wünschenswerte Anwendungen nicht realisiert werden. Mit XML entsteht jedoch eine Sprache, die ein universelles, datenbankbezogenes Erstellen von Inhalten möglich macht. Die erste offizielle Spezifikation von XML stammt vom Februar 1998, hat bisher aber nur theoretischen Wert, da die gängigen Browser XML (noch) nicht (vollständig) unterstützen. XML wird immer wieder eine wachsende Bedeutung zugesagt.

Aber eins hat die Vergangenheit seit der Geburt des PC immer wieder gezeigt: Ein Standard, der erst mal geschaffen und etabliert ist, hat ein langes Leben. Siehe MS DOS oder die 1,44 MB-Diskette. Bis alle Anwender einen Browser auf ihrem Computer installiert haben, der XML interpretieren kann, wird noch einige Zeit ins Land gehen – selbst dann, wenn er kostenlos ist.

Ob und wann XML sich durchsetzen und HTML ablösen wird, lässt sich noch nicht absehen.

JAVA

JAVA bietet als hochentwickelte Programmiersprache die Möglichkeit, Programme zu erzeugen, die auf jedem Computer laufen – und über das Internet übertragen werden können. Dabei gibt es zwei Möglichkeiten:

1. Mit JAVA lassen sich einerseits selbständig ablaufende Programme schreiben. Der JAVA Quelltext wird mit einem Compiler in einen Byte-Code übersetzt. Diese Übersetzung (und nicht der ursprüngliche ASCII-Quelltext) gelangt zum Anwender, wo er dann ausgeführt wird.

2. Mit JAVA können andererseits sogenannte JAVA-Applets programmiert werden, die für HTML-Seiten konzipiert ist und einen JAVA-fähigen Browser benötigen. Hier gelangt ebenfalls die kompilierte Datei zum Anwender und nicht der ursprüngliche Quelltext. Ausschlaggebend ist hier weder die Hardware noch das Betriebssystem, sondern der Browser als standardisierte Benutzeroberfläche, der unter dem Betriebssystem läuft – oder ohne dasselbe auskommt.

JAVA lehnt sich an die Programmiersprache C++ an. Mit JAVA können mehr oder weniger alle Anwendungen programmiert werden, die in C++ geschrieben werden können. JAVA ist durchgängig objektorientiert aufgebaut, was den Zugriff auf fertige Objekte und damit wiederverwendbare Programmteile ermöglicht.

Die JAVA-Architektur ist so konzipiert, dass der Programmcode ohne Änderung gleichzeitig auf verschiedenen Hardwareplattformen und unterschiedlichen Betriebssystemen lauffähig ist.

Risiken

Ein Script, das durch den Browser des Anwenders ausgeführt wird, ist theoretisch mit einem Risiko behaftet. Jedes ausführbare Programm kann eine Schadensfunktion zur Ausführung bringen. Zwar verbieten JavaScript, JScript und VBScript von sich aus den Zugriff auf Dateien, allerdings besitzen JScript und VBScript Mechanismen für den Zugriff auf die in einer HTML-Seite eingebetteten ActiveX-Objekte. Der Umweg über solche Objekte erlaubt einem Eindringling den Zugriff auf den eigenen Computer. Der Anwender hat die Möglichkeit, die Ausführung von ActiveX-Objekten durch den Browser grundsätzlich zu verbieten.

JavaScript bietet wenig Angriffsfläche für Eindringlinge. Zwar ist in einem Fall eine Sicherheitslücke in einem Browser bekannt geworden, die zu einem gezielten Angriff geführt hat. Hochkompetente Fachleute – wie einige Hacker – machen sich hin und wieder einen Spaß daraus, auf die Schwächen einzelner Programme oder Techniken hinzuweisen. Teilweise nur, um auf Sicherheitsmängel aufmerksam zu machen, aber auch um Schaden anzurichten. Trotzdem halte ich den Einsatz von JavaScript für unbedenklich.

Jede Technik kann missbraucht werden. Die Risiken von JavaScript sind überschaubar. Wer aus Sicherheitsgründen JavaScript in seinem Browser deaktiviert, schließt sich von einem großen Teil des Angebots im Internet aus. Den gelegentlich zu lesenden Hinweis an Autoren und Verleger, für diese Gruppe alternative Seite ohne JavaScript zur Verfügung zu stellen, halte ich nicht nur aus technischen Gründen für absurd.

Ängstliche Naturen dürfen dann bei konsequenter Anwendung ihres Sicherheitsbedürfnisses ihren Computer ebenfalls nicht mehr mit Programmen oder Daten füttern aus Angst vor Viren. Und von dem Gebrauch des Autos ist dringend abzuraten in Anbetracht der zahlreichen Todesfälle, die mit diesem Fortbewegungsmittel verbunden sind. Und selbst ein Wagenheber soll schon als Totschläger benutzt worden sein...

Quiz Sprachen

Und hier die erste Hürde auf dem Weg zum JavaScript-Meister. Welche Antworten sind korrekt?

- a) HTML ist eine Seitenbeschreibungssprache, keine Programmiersprache.
- b) JavaScript ist eine Programmiersprache.
- c) JavaScript-Programme laufen auch ohne einen Web-Browser.
- d) HTML-Seiten können mit JavaScript-Programmen erweitert werden.
- e) Seiten- und Dokumentenbeschreibungssprache sind Synonyme.

Werkzeuge

Der Browser

Es gibt mehrere Browser, von denen die beiden wichtigsten der Internet Explorer (IE) von Microsoft und der Navigator von Netscape sind. Alle anderen Produkte spielen (in der PC-Welt) so gut wie keine Rolle, unabhängig von deren Qualität. Opera verarbeitet bis zur Version 3.60 leider nur JavaScript in der Version 1.1. Deswegen bleibt die Frage nach dem richtigen Werkzeug auf die beiden ersten genannten Produkte beschränkt.

Als Anwender (als Leser von HTML-Dokumenten) sollten Sie darauf achten, dass Sie der aktuellen Entwicklung nicht allzu weit hinterherhinken. Ihr Browser sollte also die Versionsnummer 4.x oder höher tragen. Ältere Browser haben bei der Darstellung von Framesets oder bei der Ausführung von JavaScript-Programmen Schwierigkeiten. Da beide Browser kostenlos verfügbar sind, beschränkt sich Ihr Aufwand auf den Download aus dem Internet und die anschließende Installation.

Welchem Browser beim Surfen (neudeutsch für HTML-Seiten blättern und lesen) der Vorzug gegeben wird, ist mehr oder weniger eine Geschmacksfrage. Die grundlegenden Funktionen sind in beiden Produkten vorhanden, die Ausstattung der Browser ist unterschiedlich.

Keinen Browser bevorzugen

Netscape und Microsoft haben in der Vergangenheit immer wieder versucht, ihre Browser mit Eigenschaften auszustatten, die dem jeweiligen Produkt eine Vormachtstellung einräumen sollte. Dank der Bemühungen des [W3-Consortiums](#), das sich um die Standardisierung von HTML bemüht, ist diese Taktik weitgehend ohne Erfolg geblieben.

Microsoft hat den Internet Explorer in das Betriebssystem MS Windows integriert. Aufgrund der Monopolstellung in dem PC-Bereich – Microsoft hat mit seinem Betriebssystemen einen Marktanteil von ca. 90% – hat das Unternehmen die Möglichkeit, plattformübergreifende Techniken wie HTML, JavaScript und JAVA zu unterlaufen, die für eine Unabhängigkeit der Hardware und Software sorgen (wollen). Diese Möglichkeiten nutzt Microsoft immer wieder, so z.B. durch eine mangelnde Unterstützung von JAVA und JavaScript. Microsofts Taktik, den Browser zu verschenken, ihn in Windows zu verschmelzen, ihn mit eigener proprietärer Technik zu erweitern und offene Standards zu behindern, wird bei Erfolg auch hier zu einer Vormachtstellung und damit erneut zu einer Abhängigkeit von Microsoft führen.

Microsoft kann aufgrund seiner Finanzmacht den Explorer verschenken, um seine Politik durchzusetzen. Den Navigator gab es ursprünglich nur für den Privatanwender kostenlos, an Unternehmen sollte das Produkt lizenziert (verkauft) werden. Aus diesem Grund entschieden sich viele Unternehmen für den Explorer, die Bedeutung von Netscape ging zurück. Inzwischen sind beide Produkte für den privaten wie den kommerziellen Bereich kostenlos verfügbar. Netscape hat außerdem durch die Übernahme von AOL wieder deutlich an Stärke zugelegt.

Zielgruppenbetrachtungen von Marketingstrategen mögen in Anbetracht der Dominanz von MS Windows dazu neigen, HTML-Dokumente zu verfassen, welche auf die Möglichkeiten des Internet Explorer abgestimmt sind und nur dort fehlerfrei laufen. Diese Art von Pragmatismus dient nicht nur Microsofts Zielen, sondern hängt den großen Teil der Anwender ab, die (nur) den Navigator oder einen anderen Browser benutzen.

Und verschiedene HTML-Dokumente für verschiedene Browser zu schreiben ist schlicht Unsinn. Erstens werden damit wieder andere Browser wie z.B. Opera ausgeschlossen, zweitens wird der Pflegeaufwand von zwei oder noch mehr Versionen bald unerfreulich wachsen.

HTML-Dokumente sollten vor der Veröffentlichung wenigstens mit den beiden wichtigsten Browsern (also dem Netscape Navigator und Internet Explorer) getestet werden. Im Idealfall sollte der Test mit mehreren Browsern auf mehreren Plattformen (MS Windows, LINUX, Solaris, MacOS, OS/2) vorgenommen werden.

Aber: Kann uns das nicht egal sein, wer hinter der Hand steckt, die uns da so freundlich füttert?
Schließlich interessiert sich (buchstäblich) kein Schwein dafür, woher der volle Fressnapf kommt – bis es geschlachtet wird!

Deswegen gilt für alle Entwickler und Autoren: Keinen Browser bevorzugen!

Der Editor

Um HTML-Seiten zu erstellen, benötigen Sie entweder ein geeignetes Programm – einen HTML-Editor – oder ein einfaches Textverarbeitungsprogramm, mit dem ASCII-Text geschrieben und gespeichert werden kann. Um JavaScript-Programme zu erstellen, reicht das Textverarbeitungsprogramm. Anders ausgedrückt: Derzeit kenne ich kein geeignetes Werkzeug für die Erstellung von JavaScript-Programmen, das den Komfort anderer Entwicklungsumgebungen bietet. Insbesondere programmiererfahrene Anwender werden sich über die rudimentären Arbeitsmittel wundern, z.B. gibt es keinen Debugger. Für die Anfänger: Ein Debugger ist ein Hilfsmittel, um den Programmcode zeilenweise auszuführen und jederzeit die Variablen auf ihren Inhalt zu kontrollieren. Damit wird die Fehlersuche stark vereinfacht.

In jedem modernen Betriebssystem ist ein Editor integriert, mit dem der Anwender ASCII-Text schreiben und speichern kann. Unter MS Windows steht dafür das Notepad zur Verfügung. Unter MS Windows 95/98 ist das Notepad in der Startleiste unter Programme/Zubehör/Editor zu finden. Grundsätzlich kann jedes Textverarbeitungsprogramm benutzt werden, sofern es den Text als ASCII-Text (Textdokument) speichern kann.

➔ Wenn Sie unter MS Windows nicht mit dem Notepad, sondern mit einem Textverarbeitungsprogramm (WordPad, Word etc.) arbeiten, um Programmcode zu erstellen, achten Sie darauf, den Text als ASCII-Text zu speichern. Andernfalls enthält Ihr Programmcode unbrauchbare Formatierungszeichen.

In den Fällen, wo es Bestandteil dieses Tutors ist, HTML-Elemente wie Formulare zu entwerfen, bietet der Netscape Composer als [HTML-Editor](#) die geeignete Hilfe. Der Composer gehört zum Lieferumfang des kostenlosen Navigator. Er erfüllt zwar nicht alle Wünsche, nimmt dem Web-Designer aber viele lästige Aufgaben ab. Anwender, die einen PC mit Windows 98 benutzen, können ebenso FrontPage Express als HTML-Editor benutzen.

Text aus Browser kopieren

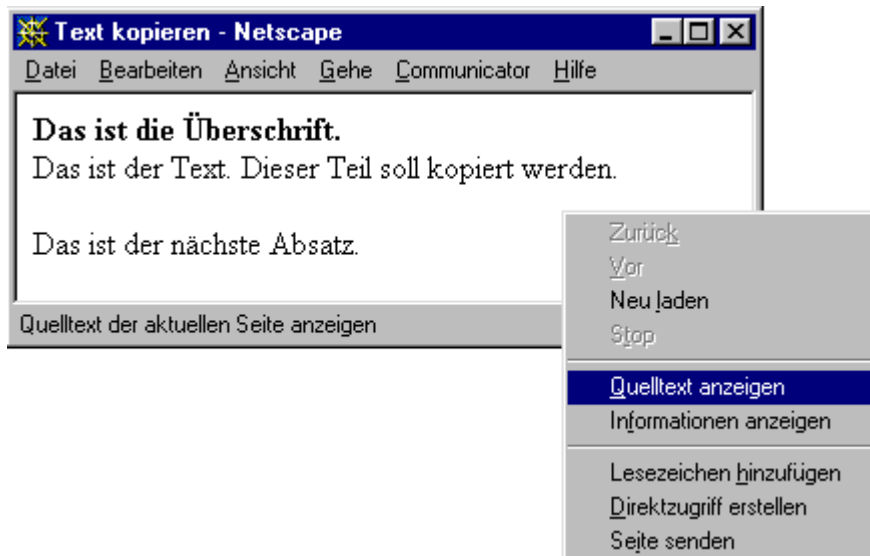
Wenn Sie mit dem Internet bereits Erfahrung gesammelt haben, werden Sie wissen, wie der dargestellte Text aus der Seite kopiert werden kann. Andernfalls sei hier noch einmal darauf hingewiesen:



Text kopieren im Netscape Navigator.

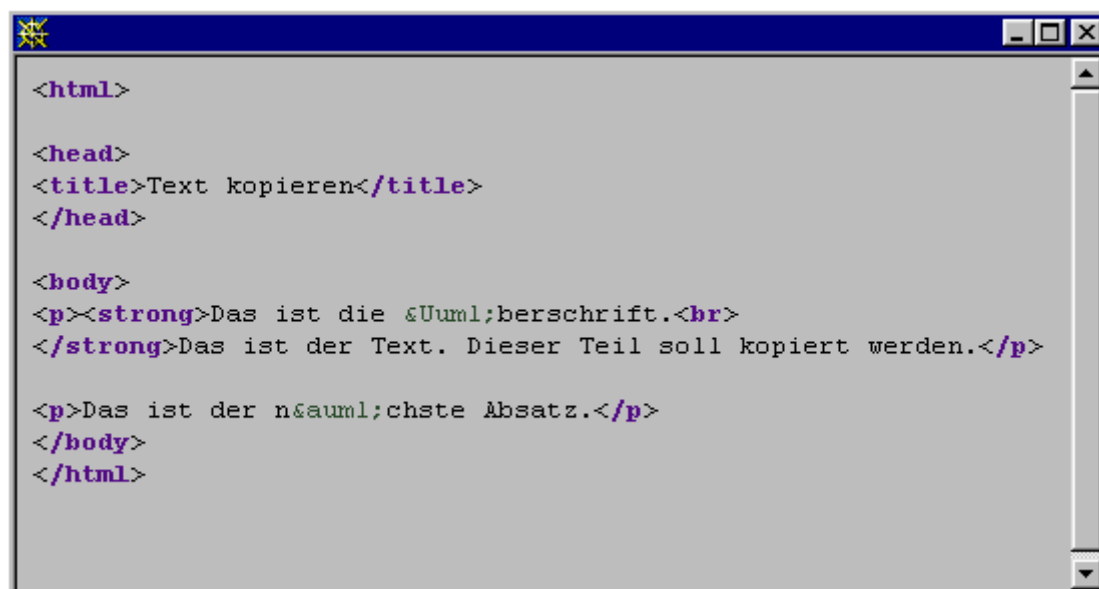
Mit der Maus den gewünschten Textabschnitt markieren. Unter MS Windows mit dem Menü »Bearbeiten/Kopieren« den Text kopieren. Danach liegt der Text in der Zwischenablage und kann in andere Programme eingefügt werden, z.B. in einen ASCII-Editor.

Quelltext im Browser anzeigen



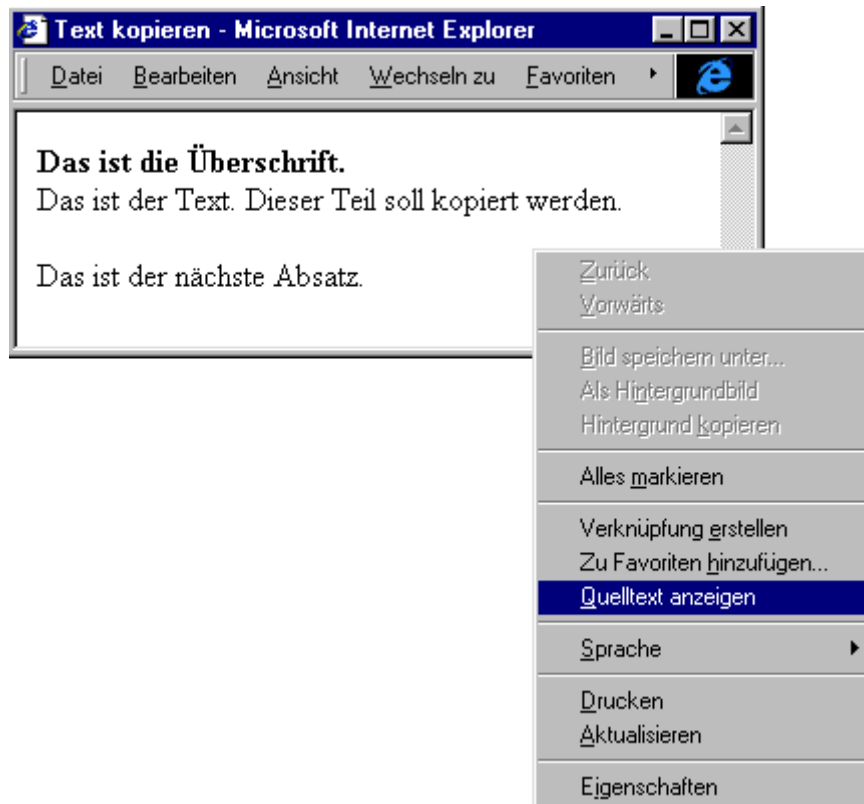
Menü (auf dem PC) mit der rechten Maustaste anzeigen.

Um den Quelltext der dargestellten Seite anzuzeigen, klicken Sie mit der rechten Maustaste (unter MS Windows und LINUX) in das Fenster auf eine leere Stelle. Das dann angebotene Menü hängt von dem jeweiligen Browser ab. Wählen Sie den Menüpunkt »Quelltext anzeigen« oder einen ähnlichen Menüpunkt in Ihrem Browser.

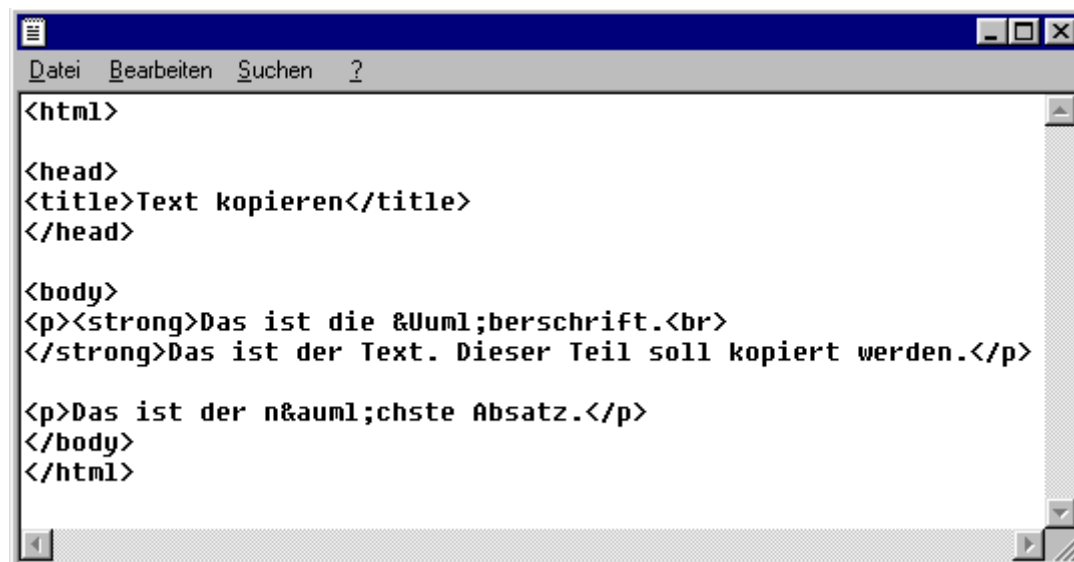


Dieselbe Seite im Navigator als Quelltext.

Das ist der Quelltext, der unter Netscape in dieser Form nicht editierbar ist. Das bedeutet, Sie haben keinen direkten Zugriff auf den ASCII-Text und damit nicht auf die Tags!



Dieselbe HTML-Seite im Internet Explorer.



Dieselbe HTML-Seite im Notepad als Quelltext.

Eine weitere Möglichkeit, den Quelltext anzuzeigen, wird über die Menüführung angeboten. Im Internet Explorer wird dazu das Menü „Ansicht/Quelltext anzeigen“ aufgerufen. Im Navigator das Menü „Ansicht/Seitenquelltext“.

Als Editor wird im Internet Explorer unter Windows (als Standardeinstellung) das Notepad geöffnet, das als ASCII-Editor funktioniert. Wenn Sie wollen, können Sie hier den Text sowie Tags editieren oder kopieren.

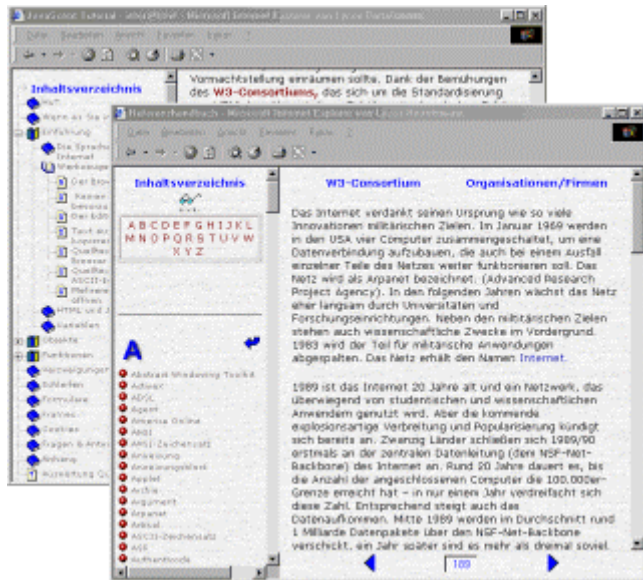
Quelltext im ASCII-Editor

Es gibt drei Möglichkeiten, den Quelltext einer HTML-Seite in einem ASCII-Editor zu bearbeiten:

1. Unter MS Windows und bei Verwendung des Internet Explorer bekommen Sie durch Klick mit der rechten Maustaste in das Fenster ein Menü angeboten, mit dem der Quelltext angezeigt werden kann. Normalerweise wird der Quelltext mit dem Notepad geöffnet.
2. Sie starten einen ASCII-Editor und öffnen die entsprechende HTML-Seite.
3. Sie starten einen ASCII-Editor: „Klicken und ziehen“ Sie die HTML-Seite in das geöffnete Fenster des ASCII-Editors.

In allen Fällen haben Sie Zugriff auf den gesamten Quelltext inklusive der Tags.

Mehrere Browser öffnen



Parallelbetrieb der Browser.

Jeder Browser lässt sich mehrfach öffnen. Dabei ist es gleichgültig, ob in dem Browser dieselbe Datei oder eine andere angezeigt wird. Verschiedene Browser wie der Netscape Navigator oder der Internet Explorer können parallel benutzt werden.

Quiz Werkzeuge

Richtig oder falsch?

- a) Die Browser von Netscape und Microsoft ab der Version 4 sind für die Verarbeitung von JavaScript geeignet.
- b) Der Programmcode muss als ASCII-Text gespeichert werden.
- c) Es können mehrere Browserfenster geöffnet werden.
- d) Der Quelltext der HTML-Seite ist für den Anwender frei zugänglich.
- e) Der Programmcode in einer HTML-Seite ist für den Anwender frei zugänglich.

HTML und JavaScript

Kein Teil von HTML

JavaScript ist kein Bestandteil von HTML, sondern eine eigene Programmiersprache mit dem Zweck, HTML-Seiten im World Wide Web mit zusätzlichen Eigenschaften auszustatten.

JavaScript ist im Textformat (Zeichen des ASCII-Zeichensatzes) geschrieben und ist damit wie HTML plattformunabhängig. Der JavaScript-Code ist in Anweisungen und Kommentare unterteilt. Der JavaScript-Code wird als lesbarer Text direkt in der HTML-Seite zwischen dem einleitenden Tag `<script>` und dem abschließenden Tag `</script>` platziert.

[Beispiel](#) `b_0001.htm`

 Öffnen Sie einen Editor und kopieren Sie den folgenden Code in Ihren Editor:


```
<html>

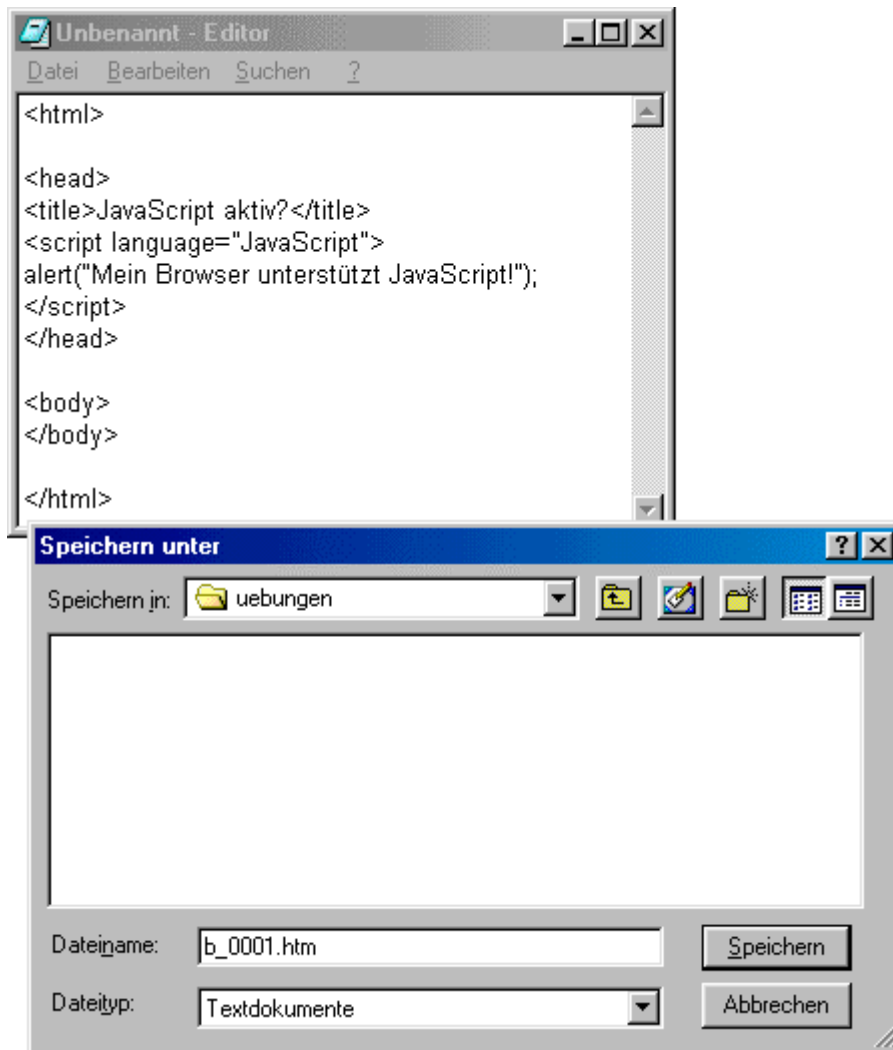
<head>
<title>JavaScript aktiv?</title>
<script language="JavaScript">
alert("Mein Browser unterstützt JavaScript!");
</script>
</head>

<body>
</body>


</html>
```

 Erstellen Sie auf Ihrem Datenträger einen Ordner, z.B. mit dem Namen »uebungen«.


 Speichern Sie die Datei in diesem Ordner unter dem Namen »b_0001.htm«. Als Dateityp muss im Notepad unter MS Windows »Textdokumente« gewählt werden. (Diese Angabe kann in einem anderen Editor anders lauten.)



HTML-Seite mit dem Dateinamen »b_0001.htm« anlegen.

 Starten Sie die HTML-Seite.

➔ Beachten Sie, dass die Seite in dem aktuellen Fenster Ihres Browsers angezeigt wird. Mit der Schaltfläche ZURÜCK in der Kopfleiste des Browsers wird der JavaScript Tutor wieder angezeigt.

 Sofern Ihr Browser JavaScript unterstützt und die Ausführung von JavaScript-Programmen zulässt, erscheint beim Start der HTML-Seite die Meldung „Mein Browser unterstützt JavaScript!“.

➔ Erscheint die Meldung nicht, nehmen Sie bitte die entsprechende Einstellung in Ihrem Browser vor:

- in dem Internet Explorer 5 in dem Menü »Extras/Internetoptionen«
- in dem Navigator 4.x in dem Menü »Bearbeiten/Einstellungen« in der Kategorie »Erweitert«.

➔ Bei den weiteren Beispielen in diesem Tutor finden Sie nicht immer den kompletten Code zum Kopieren. Wenn Sie nur fertige Beispiele kopieren, erlernen Sie nicht das Programmieren. Aber es ist sinnvoll, sich ein „Rahmengerüst“ zu schaffen, das als Basis für Ihre Arbeit dient, z.B. eine leere HTML-Seite mit den gültigen Tags für ein JavaScript-Programm wie bei der Vorlage: [Leere Seite](#)

➔ Grundsätzlich gilt: Bevor Sie eigenständigen Code verfassen, sollten Sie sich erst nach frei verfügbaren JavaScript-Beispielen umsehen. Selbst wenn Ihr Problem damit nicht vollständig gelöst wird: Es ist leichter, vorhandenen Code anzupassen, als neuen zu erstellen.

➔ Legen Sie alle weiteren Übungen jeweils mit der nächst höheren Nummer in Ihrem Übungsordner ab.

JavaScript-Bereich definieren

JavaScript-Programme laufen in einer HTML-Seite ab und werden in dieser Datei definiert, sozusagen angemeldet.

Beispiel

```
<script language="JavaScript">
alert("Mein Browser unterstützt JavaScript!");
</script>
```

oder

```
<script language="JavaScript1.2">
alert("Mein Browser unterstützt JavaScript!");
</script>
```

Erklärung zum Beispiel:

1. Mit `<script language="JavaScript">` wird ein JavaScript-Programm innerhalb einer HTML-Seite definiert. Mit dem Attribut `language=" "` wird die Script-Sprache und eventuell die Sprachversion angegeben, in diesem Fall eben JavaScript, alternativ mit der Angabe der Versionsnummer 1.2. Die JavaScript-Anweisung `alert("Mein Browser unterstützt JavaScript!");` gibt nach dem Start der HTML-Seite ein Meldungsfenster mit dem Text „Mein Browser unterstützt JavaScript!“ am Bildschirm aus. Das JavaScript-Programm endet mit dem abschließenden Tag `</script>`.
2. Browser, die kein JavaScript beherrschen, kennen das `</script>`-Tag nicht und ignorieren es. Kennt der Browser das `</script>`-Tag, beherrscht jedoch kein JavaScript, wird der Script-Bereich übergangen. Das gilt theoretisch auch, wenn der Browser die mit im Attribut `language=" "` angegebene Sprachversion nicht beherrscht.
3. Wenn die Sprachversion angegeben wird, sollte nur der Browser das Script ausführen, der die angegebene Version unterstützt. Damit sollten Laufzeitfehler verhindert werden, wenn ein Befehl unbekannt sein sollte. Der Browser, der nur eine ältere Sprachversion unterstützt, sollte das Programm gar nicht ausführen. Das ist die Theorie. Wer sich die Mühe macht, die gerade aufgestellten Behauptungen zu überprüfen, wird feststellen: Alles geht. Details dazu finden Sie unter [Versionen – ohne Bedeutung](#).

➔ Da der Navigator und der Internet Explorer seit der Version 4.x JavaScript 1.2 (mit leider vielen Inkompatibilitäten) unterstützen und hinreichend lang und kostenlos verfügbar sind, verwende ich die Sprachversion in diesem Tutor nicht weiter.

Es gibt zwar keine Vorschrift, an welcher Stelle einer HTML-Seite ein JavaScript-Bereich definiert werden muss. Es ist jedoch zu empfehlen, diesen Bereich im Kopfbereich der HTML-Seite zu definieren. Dadurch wird der Code vom Browser beim Start der HTML-Seite eingelesen und steht zur Verfügung, wenn er ausgeführt werden soll. Genauer ausgedrückt wird dadurch verhindert, dass durch eine bereits teilweise aufgebaute Seite ein Programm aufgerufen wird, das noch gar nicht zur Verfügung steht.

Wenn das Programm Informationen in die HTML-Seite schreiben soll, dann muss das Programm allerdings an der Stelle stehen, wo die Information ausgegeben werden soll.

[Beispiel b_0002.htm](#)

```
<html>

<head>
<title>JavaScript-Bereich definieren</title>
</head>

<body>
In Kürze können Sie sagen:
<p>
<script language="JavaScript">
<!--
alert("Ich verstehe auch JavaScript!");
//-->
</script>
Allerdings gilt die alte Regel in der IT-Branche: Die ersten 10 Jahre sind die schlimmsten!
</body>

</html>
```

In jeder HTML-Seite können mehrere JavaScript-Programme definiert werden. Die Programme haben jeweils ein einleitendes und abschließendes `</script>`-Tag. Die Programme dürfen nacheinander angemeldet werden oder wie in dem folgenden Beispiel an verschiedenen Stellen.

[Beispiel](#) b_0003.htm

```
<html>

<head>
<title>JavaScript-Bereich definieren</title>
<script language="JavaScript">
<!--
alert("Herzlich Willkommen!");
//-->
</script>
</head>

<body>
In Kürze können Sie sagen:
<p>
<script language="JavaScript">
<!--
alert("Ich verstehe auch JavaScript!");
//-->
</script>
Allerdings gilt die alte Regel in der IT-Branche: Die ersten 10 Jahre sind die schlimmsten!
</body>

</html>
```

Versionen – ohne Bedeutung

Die verschiedenen Versionen von JavaScript sind hinsichtlich ihrer praktischen Auswirkung mehr oder weniger bedeutungslos. Der Netscape Navigator und der Internet Explorer unterstützen seit der Version 4 offiziell die JavaScript Sprachversion 1.2, Opera bis zur Version 3.6 die JavaScript Sprachversion 1.1.

Opera führt trotzdem JavaScript-Programme aus, die im `<script>`-Tag mit der Version 1.2 angemeldet werden. Microsoft hat in dem Internet Explorer nur einen Teil des Sprachstandards von JavaScript 1.2 implementiert, akzeptiert ebenfalls klaglos die Anmeldung `<script language="JavaScript1.2">` und ignoriert die nicht bekannten Methoden und Eigenschaften einfach – bzw. gibt eine Fehlermeldung aus (letzteres hängt von der Konfiguration des Browsers ab.)

Deswegen verzichte ich auf die Angabe einer Sprachversion in dem `<script>`-Tag. Der Katastrophe nächster Teil stellt sich mit der Frage, welche Sprachversion mit welchem Umfang nun in einem Tutor wie diesem dokumentiert werden soll. Eigentlich sind alle denkbaren Antworten unbefriedigend. Ich habe mich daher für eine pragmatische Lösung entschieden: Dokumentiert wird, was der Navigator 4.7 und der Internet Explorer 5.0 unter MS Windows können. Damit fallen zahlreiche Methoden und Eigenschaften weg, die von JavaScript 1.2 jedoch unterstützt werden.

Kommentare

Kommentare in HTML

HTML bietet die Möglichkeit, an beliebigen Stellen innerhalb einer HTML-Seite Kommentare – z.B. für die eigene Dokumentation – einzufügen. Kommentare werden vom Browser nicht angezeigt.

Beispiel

```
<!-- Hier steht ein einzeiliger HTML-Kommentar -->
```

oder

```
<!-- Erste Zeile des HTML-Kommentars  
Letzte Zeile des HTML-Kommentars -->
```

Kommentare werden durch die Zeichenfolge `<!--` eingeleitet. Dahinter folgt der beliebig lange Kommentar. Innerhalb des Kommentars können HTML-Tags erwähnt werden, ohne diese maskieren zu müssen. Der Kommentar wird durch die Zeichenfolge `-->` beendet.

➔ Wenn in der HTML-Seite Tags angezeigt und nicht ausgeführt werden sollen, müssen diese Zeichen durch eine Zeichenfolge ersetzt werden, maskieren genannt:

- das Zeichen `<` wird durch die Zeichenfolge `<` ersetzt
- das Zeichen `>` wird durch die Zeichenfolge `>` ersetzt
- das Zeichen `&` wird durch die Zeichenfolge `&` ersetzt
- das Zeichen `"` wird durch die Zeichenfolge `"` ersetzt

Beim Einbinden von JavaScript-Programmen in eine HTML-Seite wird häufig empfohlen, die JavaScript-Anweisungen als mehrzeiligen HTML-Kommentar zu kennzeichnen, da einige ältere Browser das `<script>`-Tag nicht kennen und den Code nicht ausführen, sondern als Text auf dem Monitor darstellen. Die letzte Zeile des HTML-Kommentars (!) sollte dann mit den beiden Schrägstrichen `//` beginnen, die einen JavaScript-Kommentar (!) kennzeichnen. Durch die beiden Schrägstriche wird verhindert, dass der Browser das Ende des HTML-Kommentars als JavaScript-Anweisung ausführt.

Beispiel

```
<script language="JavaScript">
<!-- JavaScript-Programm als HTML-Kommentar ausweisen
alert("Mein Browser unterstützt JavaScript!");
//-->
</script>
```

Ich halte diese Forderung schlicht für Unfug der höheren Art. Eine HTML-Seite mit einem JavaScript-Programm wird ohne einen geeigneten Browser kaum funktionieren. Folglich ist es egal, ob in der HTML-Seite dann auch noch der Quelltext lesbar ist. Diejenigen, die mit einem alten Browser arbeiten, werden dadurch vielleicht endlich wach und bemühen sich um eine neue Version, die von Microsoft und Netscape ja kostenlos zu haben ist. Und für diejenigen, die mit einer alten Kiste arbeiten, bleibt immerhin die Erkenntnis, dass mit einem Tretroller eben kein Formel-1-Rennen zu gewinnen ist.

Dass ich trotzdem in den Beispielen den JavaScript-Code in HTML-Kommentare setze, hängt mit meiner Arbeitstechnik zusammen. Da Softwareentwicklung Evolution und nicht Revolution ist, beginne ich meine Arbeit damit, eine ähnliche Lösung (aus meiner Bespielsammlung) zu verändern. Damit ich meine Arbeit bei eventuell auftretenden Fehlern zurückverfolgen kann, kopiere ich den letzten funktionstüchtigen Stand an eine andere Stelle der HTML-Seite, wo er als Kommentar keinen Schaden anrichten kann. Dort steht er jederzeit wieder zur Verfügung, falls nötig.

[Beispiel](#) `b_0004.htm`

```
<html>

<head>
<title>JavaScript-Bereich definieren</title>
<!--
alert("Herzlich Willkommen!");
//-->
</head>

<body>
In Kürze können Sie sagen:
<p>
<script language="JavaScript">
<!--
alert("Ich verstehe auch JavaScript!");
//-->
</script>
Allerdings gilt die alte Regel in der IT-Branche: Die ersten 10 Jahre sind die schlimmsten!
</body>

</html>
```

➔ Eine Vorlage für eine [leere Seite](#) mit HTML-Kommentar.

Kommentare in JavaScript

Kommentare dienen zur Erklärung der einzelnen Anweisungen und werden vom JavaScript-Interpreter des Browsers ignoriert. Ein einzeiliger JavaScript-Kommentar beginnt mit zwei Schrägstrichen //, dahinter steht der eigentliche Text. Ein mehrzeiliger Kommentar beginnt mit einem Schrägstrich und einem Sternchen /* und endet mit der umgekehrten Kombination */.

Beispiel

```
alert("Mein Browser unterstützt JavaScript!"); // Meldung ausgeben
```

oder

```
/* Dies ist ein mehrzeiliger Kommentar, der zur Dokumentation  
auch umfangreichen Text beinhalten kann. */
```

Programm ausführen

Es gibt zwei Möglichkeiten, ein JavaScript-Programm auszuführen:

- beim Start der HTML-Seite
- über ein Ereignis, bzw. über eine [Funktion](#)

In unserem Beispiel wird der JavaScript-Code automatisch beim Einlesen der HTML-Seite ausgeführt:

```
<script language="JavaScript">  
alert("Mein Browser unterstützt JavaScript!");  
</script>
```

In vielen Fällen soll ein Programm aber ereignisabhängig ausgeführt werden, z.B. durch Klicken einer Schaltfläche, nachdem vorher in einem Formular einige Bestellungen durchgeführt wurden. In diesem Fall wird über die Schaltfläche eine bestimmte Funktion in dem JavaScript-Programm aufgerufen. Mehr dazu im Kapitel Funktionen in diesem Tutor.

JavaScript-Code in separaten Dateien

Der JavaScript-Code kann in einer separaten Datei notiert werden. Damit ist es möglich, den gleichen Code in mehreren HTML-Seiten zu verwenden. Die Datei mit dem Code muss in jeder HTML-Seite nur angemeldet (referenziert) werden:

```
<script language="JavaScript" src="dateiname.js">
</script>
```

Mit dem Attribut `src=` (`src = source = Quelle`) wird die Quelle angegeben, der Dateiname wird den HTML-Regeln entsprechend gewählt. Als Dateikennung muss `.js` vergeben werden. Die JavaScript-Datei mit dem Quellcode muss eine ASCII-Datei sein.

➔ Eine JavaScript-Datei erzeugen Sie ebenso wie eine HTML-Seite mit einem ASCII-Editor.


➔ Den Quelltext der JavaScript-Datei können Sie betrachten und editieren, indem Sie:

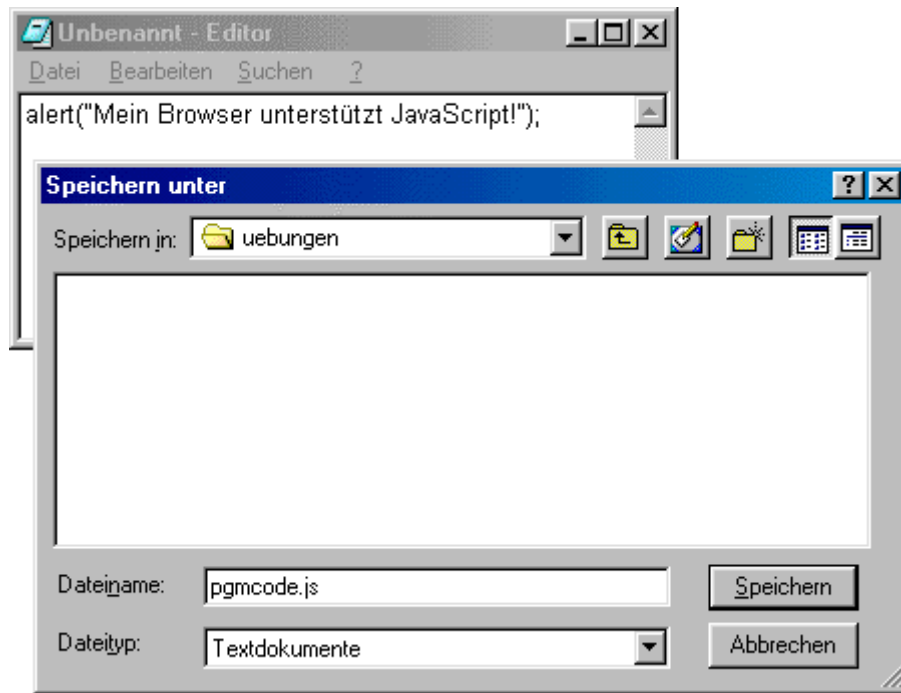
- die JavaScript-Datei doppelklicken, sofern die Dateikennung ».js« mit einem ASCII-Editor verknüpft ist.
- einen ASCII-Editor starten und dann die JavaScript-Datei öffnen.

[Beispiel](#) b_0005.htm


 Kopieren Sie den folgenden Code in Ihren Texteditor und erstellen Sie eine JavaScript-Datei:

```
alert("Mein Browser unterstützt JavaScript!");
```

 Speichern Sie die Datei unter dem Namen »pgmcode.js« in Ihrem Übungsordner. Wenn Sie das Notepad unter MS Windows verwenden, muss als Dateityp »Textdokumente« gewählt werden. (Diese Angabe kann in einem anderen Editor anders lauten.)



Eine JavaScript-Datei anlegen.


 Kopieren Sie den folgenden Code in Ihren Texteditor und erstellen Sie eine HTML-Seite:

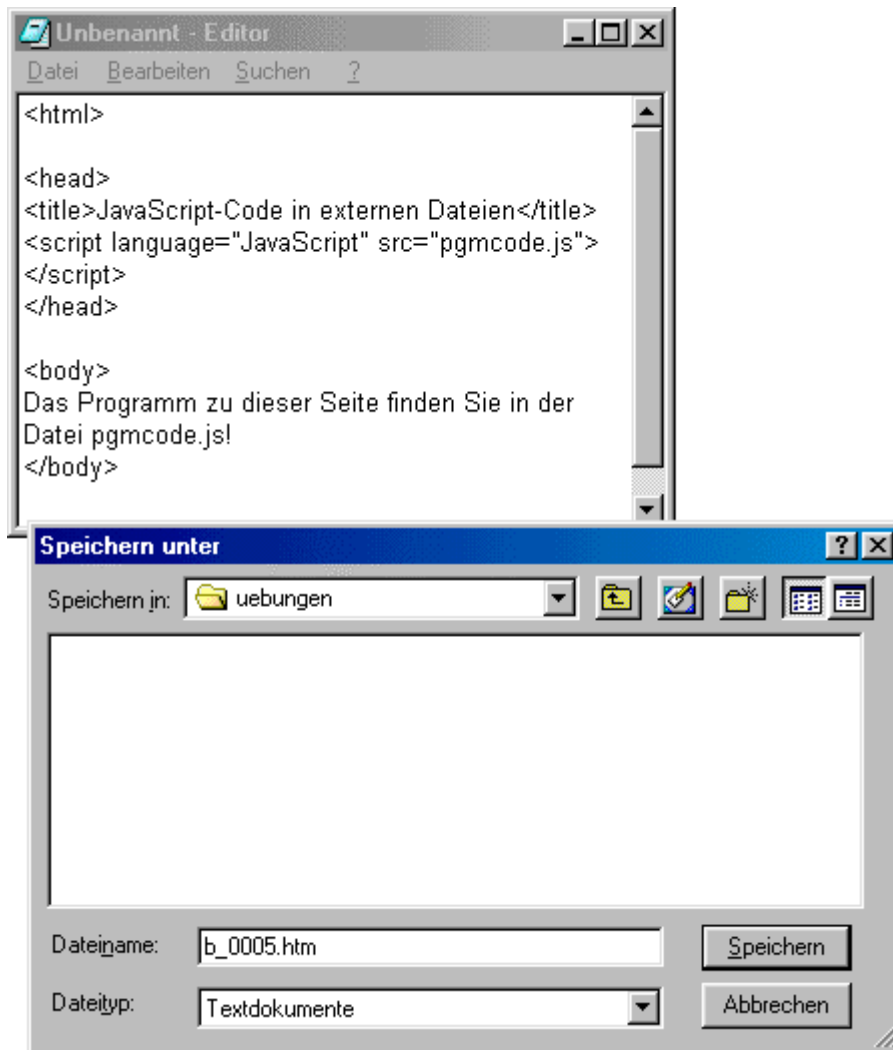
```
<html>

<head>
<title>JavaScript-Code in externen Dateien</title>
<script language="JavaScript" src="pgmcode.js">
</script>
</head>

<body>
Das Programm zu dieser Seite finden Sie in der Datei pgmcode.js!
</body>


</html>
```

 Speichern Sie die Datei unter dem Namen »b_0005.htm« in Ihrem Übungsordner.



HTML-Seite mit dem Dateinamen »b_0005.htm« anlegen.

 Starten Sie die Datei.

 Bei dem Start der Datei erscheint die Meldung „Mein Browser unterstützt JavaScript!“ – anschließend wird der Text in der HTML-Seite angezeigt.

Anweisungen notieren

Eine JavaScript-Anweisung ist eine Aktion. Um eine Aufgabe zu lösen, sind der Aufgabe entsprechend viele Anweisungen nötig. Eine Anweisung besteht aus einer definierten Anordnung von **Befehlen**, Symbolen und **Schlüsselwörter** in einer Zeile. Eine neue Zeile leitet in der Regel eine neue Anweisung ein, aus Gründen der Übersichtlichkeit ist diese Vorgehensweise sehr zu empfehlen. (Es ist jedoch erlaubt, alle Anweisungen in einer Zeile zu notieren.)

Eine Anweisung sollte explizit mit einem Semikolon (;) abgeschlossen werden, dem JavaScript-Endezeichen. Eine Anweisung ist entweder eine **Wertzuweisung** oder ein **Funktions-** oder **Methodenaufruf**. Die Anweisungen werden zeilenweise vom JavaScript-Interpreter in Maschinencode umgesetzt und ausgeführt.

Beispiel

```
alert("Mein Browser unterstützt JavaScript!");
```

Erklärung zum Beispiel:

Hier handelt es sich um einen Methodenaufruf.

Beispiel

```
var zahl_1 = 55;
```

Erklärung zum Beispiel:

Hier handelt es sich um eine Wertzuweisung.

Ein Problem bei der Programmierung besteht darin, bei der Erstellung des Codes konzeptionell weit vorausdenken, weil auf Programmteile Bezug genommen werden muss, die noch gar nicht geschrieben sind. Es wäre als schöner, das Ergebnis seiner Arbeit zu kennen, bevor die Arbeit begonnen hat.

Anweisungsblöcke notieren

Eine von geschweiften Klammern eingeschlossene Gruppe von JavaScript-Anweisungen wird als Anweisungsblock bezeichnet. Anweisungsblöcke werden in [Funktionen](#) und [Bedingungen](#) verwendet. In dem folgenden Beispiel beginnt die erste Anweisung mit der Definition einer Funktion, die aus einem Anweisungsblock mit vier Anweisungen besteht.

Beispiel

```
function Addition()  
{  
  var zahl_1 = 55;  
  var zahl_2 = 60;  
  var summe = zahl_1 + zahl_2;  
  alert("Die Summe von " + zahl_1 + " + " + zahl_2 + " = " + summe);  
}
```

Erklärung zum Beispiel:

1. Ein Anweisungsblock beginnt und endet mit einer geschweiften Klammer `{}`. Anweisungsblöcke werden notiert, wenn mehr als zwei Anweisungen ausgeführt werden sollen. Anweisungsblöcke können verschachtelt sein.
2. Um die Übersicht zu bewahren, ist es zu empfehlen, Anweisungsblöcke durch Einrückungen zu kennzeichnen.

Zuweisungen und Gleichheit

Das Gleichheitszeichen (=) wird in JavaScript für eine Wertzuweisung verwendet. Die JavaScript-Anweisung `var zahl_1 = 55;` besagt „Weise der Variablen zahl_1 den Wert 55 zu“.

Um zwei Werte miteinander zu vergleichen, werden zwei Gleichheitszeichen (==) verwendet. Mehr dazu später.

Ausdrücke

Ein Ausdruck in JavaScript ist eine mathematische Operation, z.B. eine Addition von zwei Zahlen. Ein Ausdruck kann sich außerdem auf Boole'sche Werte und auf Zeichenketten beziehen. (Mehr dazu in dem Kapitel „Operatoren“ in diesem Tutor.) Ausdrücke enthalten Symbole wie das Pluszeichen „+“ etc. Jede gültige Kombination von Werten, Variablen und Operatoren bildet einen Ausdruck.

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 + zahl_2;
```

Erklärung zum Beispiel:

Zunächst werden zwei Variablen mit je einer Zahl deklariert. In der JavaScript-Anweisung `var summe = zahl_1 + zahl_2;` bildet „zahl_1 + zahl_2“ den Ausdruck. In diesem Fall werden zwei Variablen addiert. Dieser Ausdruck wird deswegen als numerischer Ausdruck bezeichnet.

Namensgebung

Für selbst definierte Variablen und **Funktionen** müssen Sie jeweils einen Namen vergeben.

JavaScript unterscheidet zwischen Groß- und Kleinschreibung. Weiter gelten folgende Regeln:

- Namen dürfen keine Leerzeichen enthalten
- Namen dürfen nur aus Buchstaben und Ziffern bestehen
- Namen dürfen keine deutschen Umlaute und kein ß enthalten
- Namen dürfen als einzige Sonderzeichen den Unterstrich (_) und das Dollarzeichen (\$) enthalten
- Namen dürfen nicht mit einem Schlüsselwort identisch sein
- das erste Zeichen darf keine Ziffer sein
- Groß- und Kleinbuchstaben sind erlaubt

Einige gültige Variablennamen:

- `zahl_1`
- `Zahl_1`
- `zahl1`
- `_zahl`

Einige ungültige Variablennamen:

- `1zahl` // Beginnt mit einer Zahl.
- `zahl1&zahl2` // Kaufmännisches Und-Zeichen (&).

Sauberer Programmierstil

Vielleicht ist es Ihnen beim Betrachten der Quelltexte aufgefallen, dass die Anweisungen in der **Funktion** eingerückt sind. Für die korrekte Ausführung des Programms ist es unerheblich, ob Einrückungen verwendet werden, oder ob alle Programmzeilen am Zeilenanfang stehen. Für die Lesbarkeit des Quelltextes sind Einrückungen sehr wichtig.

Beispiel

```
function Addition()  
{  
  var zahl_1 = 55;  
  var zahl_2 = 60;  
  var summe = zahl_1 + zahl_2;  
  alert("Die Summe von " + zahl_1 + " + " + zahl_2 + " = " + summe);  
}
```

Erklärung zum Beispiel:

Das Programm wird korrekt ausgeführt. Beinhaltet das Programm mehr als eine Funktion, führt diese Schreibweise zur Unübersichtlichkeit.

Beispiel

```
function Addition() {  
  var zahl_1 = 55;  
  var zahl_2 = 60;  
  var summe = zahl_1 + zahl_2;  
  alert("Die Summe von " + zahl_1 + " + " + zahl_2 + " = " + summe);  
}
```

Erklärung zum Beispiel:

Besser ist es, den gesamten Anweisungsblock einzurücken, z.B. mit einem Tabulator.

Beispiel

```
function Addition()  
{  
  var zahl_1 = 55;  
  var zahl_2 = 60;  
  var summe = zahl_1 + zahl_2;  
  alert("Die Summe von " + zahl_1 + " + " + zahl_2 + " = " + summe);  
}
```

Erklärung zum Beispiel:

Noch übersichtlicher wird der Code, wenn beide geschweiften Klammern ebenfalls eingerückt sind.

Quiz HTML und JavaScript I

Richtig oder falsch – welche Antworten treffen zu?

- a) `<script language="JavaScript1.2">` und `<script language="JavaScript 1.2">` sind identische Tags.
- b) `<--` kennzeichnet den einleitenden JavaScript-Kommentar.
- c) `//` kennzeichnet einen einzeiligen JavaScript-Kommentar.
- d) Browser, die das `<script>`-Tag nicht kennen, stellen den Quelltext in der HTML-Seite dar.
- e) Ein JavaScript-Bereich muss im Kopfbereich der HTML-Seite stehen.

Quiz HTML und JavaScript II

Richtig oder falsch – welche Antworten treffen zu?

- a) Eine JavaScript-Datei ist eine reine Textdatei.
- b) Eine JavaScript-Datei kann von mehreren HTML-Seiten benutzt (referenziert) werden.
- c) Eine JavaScript-Datei dient auch zur Speicherung von Daten.
- d) Eine externe JavaScript-Datei kann mit einem Texteditor bearbeitet werden.
- e) Eine JavaScript-Datei kann keine HTML-Kommentare beinhalten.

Quiz HTML und JavaScript III

Richtig oder falsch – welche Antworten treffen zu?

- a) Das Gleichheitszeichen (`=`) dient für eine Wertzuweisung.
- b) Eine Anweisung sollte explizit mit einem Semikolon (`;`) abgeschlossen werden.
- c) Eine von geschweiften Klammern eingeschlossene Gruppe von JavaScript-Anweisungen wird als Anweisungsblock bezeichnet.
- d) Einrückungen im Programmcode dienen nur der besseren Übersicht.
- e) Namen für Variablen dürfen nicht mit einem Schlüsselwort identisch sein.

Variablen

Was ist eine Variable?

In JavaScript werden Variablen verwendet, um Werte zu speichern. Die Variablen bieten die Möglichkeit, Werte durch die Angabe von Namen abzufragen und zu verändern. Der Variablenname ist unter Berücksichtigung der [Namensgebung](#) frei wählbar.

Der Variablen wird mit dem Gleichheitszeichen (=) ein Wert zugewiesen. Jede neue Wertzuweisung bewirkt, dass der bis dahin gespeicherte Wert verloren geht. Ein Wert kann beinhalten:

- eine Zahl (Datentyp „Numerisch“), z.B. 55
- ein Text (Datentyp „Zeichenkette“), z.B. "Das ist Text."
- einen Boole'schen Wert (Datentyp „Boolean“), z.B. true oder false
- ein Objekt, z.B. window.document.images[0]

Diese unterschiedlichen Werte (Zahl, Text etc.) werden als Datentypen bezeichnet. JavaScript ist eine lose typisierte Sprache, d.h. einzelne Datentypen der Variablen müssen (oder können) nicht explizit deklarieren werden.

Als Sonderfall können die beiden Variablen mit den Werten „null“ und „undefined“ betrachtet werden.

Deklaration

Es ist zwar nicht zwingend erforderlich, aber sinnvoll, die Variablen vor ihrer Verwendung explizit zu deklarieren. Die explizite Deklaration wird mit dem Schlüsselwort `var` vorgenommen. Aus Gründen der Übersichtlichkeit sollten die Variablen an einer Stelle deklariert werden, sofern es mit dem gewünschten Programmablauf zu vereinbaren ist.

Eine Variable wird implizit deklariert, indem das `var`-Schlüsselwort weggelassen wird.

Beispiel

```
var zahl_1 = 55; // Eine Variable vom Typ Zahl.  
var zahl_2 = 60; // Eine Variable vom Typ Zahl.  
var summe = zahl_1 + zahl_2; // Eine Addition zweier Zahlen ergibt wieder  
eine Zahl.  
var meldung = "Herzlich Willkommen!"; // Eine Variable vom Typ "String"  
enthält Text.  
var bezahlt = true; // Die Variable enthält den Datentyp "Boolean".
```

Erklärung zum Beispiel:

1. Bei der Wertzuweisung unterscheiden sich Zahlen und Text durch das Anführungszeichen. Der Text steht in Anführungszeichen, die Zahl wird ohne Anführungszeichen notiert.

2. Rechts neben dem Gleichheitszeichen darf eine Operation stehen. Das Ergebnis dieser Operation wird in der Variablen gespeichert.

Beispiel

```
var bilderLaden = checkImages(); // Funktionsaufruf  
var eingabe = window.prompt("Geben Sie die Bestellmenge ein:", ""); //  
Methode des Objekts window
```

Rechts neben dem Gleichheitszeichen darf eine Methode oder Funktion stehen, die ein Ergebnis liefert. Dieses Ergebnis wird in der Variablen gespeichert.

Beispiel

```
var zahl_1 = 55;  
zahl_1 = zahl_1 + 1;
```

Erklärung zum Beispiel:

Eine Variable kann rechts und links von Gleichheitszeichen stehen. Zu dem Wert der Variablen `zahl_1` wird eine 1 addiert. Das Ergebnis wird wieder in der Variablen `zahl_1` gespeichert.

Beispiel

```
var zahl_1 = 55;
var zahl_2 = zahl_1;
```

Erklärung zum Beispiel:

Rechts neben dem Gleichheitszeichen darf statt einem Wert eine weitere Variable stehen. Dann wird der Wert dieser Variablen in der Variablen gespeichert, die links vom Gleichheitszeichen steht. In diesem Fall haben beide Variablen den gleichen Wert.

Beispiel

```
var zahl_1;
var summe = 3 * zahl_1;
```

Erklärung zum Beispiel:

1. Eine Variable darf ohne Wertzuweisung deklariert werden.
2. Wird der Variablen vor dem Gebrauch kein Wert zugewiesen, bekommt sie den Status `undefined` und die Programmausführung führt zu einem Laufzeitfehler. Die mathematische Operation führt zu dem Ergebnis `NaN` (Not a Number), weil die Variable `zahl_1` hier keinen Wert hat.

Beispiel

```
var zahl_1 = null;
var summe = 3 * zahl_1;
```

Erklärung zum Beispiel:

Alternativ kann der Variablen der Wert `null` zugewiesen werden. Die Variable `summe` wird zu 0.

Beispiel

```
zahl_1 = ""; // Die Variable zahl_1 ist implizit deklariert.
var summe = a + b; // Löst einen Fehler aus, da a und b nicht deklariert wurden.
```

Erklärung zum Beispiel:

1. Eine Variable wird implizit deklariert (ohne das Schlüsselwort `var` zu verwenden), indem ihr ein Wert zugewiesen wird.

2. Eine Variable kann nicht verwendet werden, die noch nie deklariert wurde. Wird sie trotzdem verwendet, tritt ein Laufzeitfehler auf.

➡ JavaScript unterscheidet zwischen lokalen und globalen Variablen. Lokale Variablen gelten nur in einem Programmteil, globale Variablen stehen dem gesamten Programm zur Verfügung. Näheres zu lokalen und globalen Variablen sowie den Auswirkungen von impliziten Variablendeklarationen in Funktionen auf den Gültigkeitsbereich finden Sie bei dem [Einsatz der Variablen](#) im Kapitel Funktionen in diesem Tutor.

Konvertierung

Variablen in JavaScript verfügen über keine unterschiedlichen Datentypen. Variablen werden für die verschiedenen Datentypen wie Zahlen, Texte etc. nicht besonders gekennzeichnet, in anderen Programmiersprachen ist das aber durchaus üblich. Anders ausgedrückt darf eine Variable erst mal alles beinhalten.

Immerhin wird eine Zahl bei der Deklaration als Zahl gekennzeichnet, indem sie nicht in Anführungszeichen gesetzt wird wie eben ein Text. Wie eine Variable in dem Programm behandelt wird, hängt von dem weiteren Code ab. Eine Zahl kann in einen Text konvertiert werden. In einigen Fällen wird in JavaScript eine automatische Konvertierung einer Variablen in einen anderen Typ vorgenommen. Zahlen können z.B. in einen Text eingefügt werden. Bei einer Zahl, die nicht als Zahl deklariert wurde, stehen dafür Konvertierungsfunktionen zur Verfügung.

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var meldung_1 = "Die Zahlen ";
var meldung_2 = " werden als Text behandelt.";
var ausgabe = meldung_1 + zahl_1 + " und " + zahl_2 + meldung_2;
```

Erklärung zum Beispiel:

Nach der Programmausführung enthält die Variable `ausgabe` den Text: „Die Zahlen 55 und 60 werden als Text behandelt.“ Die numerischen Daten wurden in eine Zeichenkette umgewandelt.

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var zahl_3 = "75";
var ausgabe = zahl_1 + zahl_2 + zahl_3;
```

Erklärung zum Beispiel:

Nach der Programmausführung enthält die Variable `ausgabe` den Text: „11575“. Der Grund liegt in den Datentypen:

- Die "75" steht in Anführungszeichen und ist eine Zeichenkette (also ein Text), die 55 und die 60 sind dagegen Zahlen.
- Da zuerst zwei Zahlen mit dem Operator + addiert werden, lautet das (unsichtbare)

Zwischenergebnis 115.

- Da die "75" jedoch eine Zeichenkette ist, bewirkt der folgende Operator + jetzt keine mathematische Addition, sondern eine Verkettung von zwei Zeichenketten. Das Ergebnis lautet „11575“.

Beispiel

```
var zahl_1 = 55;  
var zahl_2 = 60;  
var zahl_3 = "75";  
var ausgabe = zahl_3 + zahl_1 + zahl_2;
```

Erklärung zum Beispiel:

Nach der Programmausführung enthält die Variable `ausgabe` den Text „755560“. Da sich der erste Operator + auf eine Zeichenkette und eine Zahl bezieht, entsteht eine Verkettung, der mit dem zweiten Operator + eine weitere Zeichenkette hinzugefügt wird.

Beispiel

```
var zahl_1 = 55;  
var zahl_2 = 60;  
var zahl_3 = "75";  
zahl_3 += zahl_1 + zahl_2;
```

Erklärung zum Beispiel:

Eine besondere Form der Notation ist in diesem Fall durch += gegeben. Nach der Programmausführung enthält die Variable `zahl_3` den Text „75115“.

Datentyp für Text

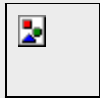
Der als Zeichenkette bezeichnete Text wird in Apostrophe oder Anführungszeichen gesetzt. Apostrophe werden bei Zeichenketten verwendet, die Anführungszeichen enthalten. Eine Zeichenkette ist also eine Anordnung von Zeichen und Ziffern, in vielen Fällen lesbarer Text. Der entsprechende Ausdruck dafür lautet String. In JavaScript ist ein String ein eigenes Objekt.

Beispiel JavaScript

```
var text_1 = "Das ist normaler Text."  
var text_2 = '"Ich bin wieder da!" ruft der Marius.'  
var text_3 = "123"  
var text_4 = ""
```

Erklärung zum Beispiel:

Alle vier Variablen enthalten Text. Das prägende Merkmal sind die Apostrophe oder Anführungszeichen. Enthält ein String-Objekt kein Zeichen, handelt es sich um eine Nullzeichenfolge ("").



Variable deklarieren

- Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:
 - Vergeben Sie als Titel der HTML-Seite „Variable deklarieren“.
 - Deklarieren Sie eine Variable „meldung“.
 - Weisen Sie der Variablen den Text „Herzlich Willkommen!“ zu.
 - Geben Sie die Variable über den »alert()«-Befehl aus, wenn die HTML-Seite gestartet wird.



Vergleichen Sie Ihre Lösung mit dem [Muster b_0006.htm](#).

Datentyp für Zahlen

JavaScript unterstützt Ganzzahlen und Fließkommazahlen – statt des Dezimalkommas wird jedoch der Dezimalpunkt verwendet. Die Zahlen können positiv, 0 oder negativ sein. Zur eindeutigen Unterscheidung zwischen Text und Zahl wird eine Zahl nicht in Anführungszeichen gesetzt.

Beispiel

```
var zahl_1 = -13.5;  
var zahl_2 = 5.5;
```

Eine Fließkommazahl kann einen Dezimalpunkt und ein „e“ (oder ein „E“) als Darstellung für die Exponentialschreibweise enthalten. Weiter gibt es die Sonderfälle:

- NaN (Not A Number)
- plus unendlich
- minus unendlich
- plus 0
- minus 0

Ganzzahlen können mit der Basis 10 (dezimal), 8 (oktal) oder 16 (hexadezimal oder kurz hex) dargestellt werden.

Ganzzahlen im Oktalsystem werden mit einer „0“ zu Beginn dargestellt und können die Ziffern von 0 bis 7 enthalten. Eine oktale Zahl mit dem Buchstaben „e“ (oder ein „E“) erzeugt einen Fehler.

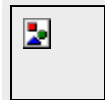
Ganzzahlen im Hexadezimalsystem werden mit „0x“ (oder „0X“) eingeleitet und enthalten die Ziffern 0 bis 9 und die Buchstaben A bis F (als Groß- oder Kleinbuchstaben) enthalten. Der Buchstabe „e“ ist eine mögliche Ziffer in der hexadezimalen Schreibweise und gibt keinen Exponenten an. Die Buchstaben A bis F werden verwendet, um die Zahlen 10 bis 15 zur Basis 16 darzustellen.

Ganze Zahlen im Oktal- und Hexadezimalsystem können negativ sein. Eine Zahl, die mit einer einzelnen „0“ beginnt und einen Dezimalpunkt enthält, ist eine dezimale Fließkommazahl. Eine Zahl, die mit „0x“ oder „00“ beginnt und einen Dezimalpunkt enthält, ist eine Hexadezimalzahl oder eine Oktalzahl – alle Stellen rechts vom Dezimalpunkt werden ignoriert.

Beispiel

- Die Dezimalzahlen (als Fließkommazahl mit dem Dezimalpunkt) `.0001` und `0.0001` und `1e-4` und `1.0e-4` sind äquivalent.

-
- Die Dezimalzahl $1.23e2$ ist äquivalent zu 123 .
 - 55 ist eine Ganzzahl im Dezimalzahlsystem.
 - 0678 ist eine Ganzzahl im Dezimalzahlsystem und äquivalent zu 678 .
 - 0370 ist eine Ganzzahl im Oktalsystem und äquivalent zu der Dezimalzahl 248 .
 - Bei der Oktalzahl 00.0001 werden die Stellen rechts vom Dezimalpunkt ignoriert, sie ist äquivalent zu der Dezimalzahl 0 .
 - Die Hexadezimalzahl $0xF$ ist äquivalent zu der Dezimalzahl 15 .
 - Die Hexadezimalzahl $0x10$ ist äquivalent zu der Dezimalzahl 16 .
 - Die Hexadezimalzahl $0xFF$ ist äquivalent zu der Dezimalzahl 255 .
 - Die Hexadezimalzahl $0x2e3$ ist äquivalent zu der Dezimalzahl 739 .
 - Bei der Hexadezimalzahl $0x5.25e2$ werden die Stellen rechts vom Dezimalpunkt ignoriert – sie ist äquivalent zu der Dezimalzahl 5 .



Variable deklarieren

- Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:
 - Vergeben Sie als Titel der HTML-Seite „Variable deklarieren“.
 - Deklarieren Sie eine Variable „zahl_1“ mit dem Wert „55“.
 - Deklarieren Sie eine Variable „zahl_2“ mit dem Wert „60“.
 - Deklarieren Sie eine Variable „summe“ mit der Addition von „zahl_1“ und „zahl_2“.
 - Geben Sie über den »alert()«-Befehl die Meldung „Die Summe von $55 + 60 = 115$.“ aus, wenn die HTML-Seite gestartet wird. Die Zahlen sollen in dem »alert()«-Befehl durch die Variablen ausgedrückt werden.



Vergleichen Sie Ihre Lösung mit dem [Muster b_0007.htm](#).

Datentyp null

Ein Wert null ist einer, der keinen Wert hat und nichts bedeutet.

Beispiel

```
var zahl_1 = 10;
var zahl_2 = null;
var summe = zahl_1 * zahl_2; // Die Variable summe enthält den Wert 0.
```

oder

```
var zahl_1 = 10;
var zahl_2 = null;
var summe = zahl_1 + zahl_2; // Die Variable summe enthält den Wert 10.
```

Quiz Variablen I

Welche Antworten treffen zu oder wie variabel sind Sie?

- a) JavaScript kennt keine unterschiedlichen Variablen für die verschiedenen Datentypen.
- b) Eine Zeichenfolge in Anführungszeichen wird als Text behandelt.
- c) Ein Text und eine Zeichenkette bezeichnen den selben Datentyp.
- d) Die Wertzuweisung einer Variablen wird mit einem Semikolon abgeschlossen.
- e) Für Fließkommazahlen wird der Dezimalpunkt verwendet.

Quiz Variablen II

Und hier der Katastrophe nächster Teil: Was ist richtig?

- a) Eine Variable muss explizit deklariert werden.
- b) Eine Variable kann als Wertzuweisung eine Operation beinhalten.
- c) Eine Variable kann ein Objekt beinhalten.
- d) Eine Variable kann einen Boole'schen Wert beinhalten.
- e) Rechts neben dem Gleichheitszeichen darf eine Methode oder Funktion stehen.

Quiz Variablen III

Was ist richtig?

- a) zahl_1 und Zahl_1 sind gültige Variablennamen.
- b) 3zahlen ist ein gültiger Variablenname.
- c) Variablennamen dürfen den Unterstrich enthalten.
- d) Variablennamen dürfen Leerzeichen enthalten.
- e) Variablennamen dürfen keine deutschen Umlaute enthalten.

Schlüsselwörter

Reservierte Namen

Wie jede Programmiersprache reserviert JavaScript bestimmte Schlüsselwörter, die für ihre eigene Verwendung bestimmt sind. Die folgenden Schlüsselwörter dürfen nicht als Namen für die eigenen Variablen und Funktionen verwendet werden:

Schlüsselwort	Bedeutung
abstract	Noch keine Verwendung.
boolean	Noch keine Verwendung.
break	Abbruch einer Schleife oder in einer bedingten Anweisung mit switch.
byte	Noch keine Verwendung.
case	Fallunterscheidung in bedingter Anweisung mit switch.
catch	Noch keine Verwendung.
char	Noch keine Verwendung.
class	Noch keine Verwendung.
const	Noch keine Verwendung.
continue	Schleifendurchlauf erzwingen.
debugger	Noch keine Verwendung.
default	Fallunterscheidung in bedingter Anweisung mit switch.
delete	Element eines Arrays oder Eigenschaft eines selbst definierten Objekts löschen.
do	Schleife mit do-while.
double	Noch keine Verwendung.
else	Bedingte Anweisung.
enum	Noch keine Verwendung.
export	Noch keine Verwendung.
extends	Noch keine Verwendung.
false	Boole'scher Wert.
final	Noch keine Verwendung.
finally	Noch keine Verwendung.
float	Noch keine Verwendung.
for	Schleife mit for.
function	Deklaration einer Funktion.
goto	Noch keine Verwendung.
if	Bedingte Anweisung.

implements	Noch keine Verwendung.
import	Noch keine Verwendung.
in	Schleife mit for.
instanceof	Noch keine Verwendung.
int	Noch keine Verwendung.
interface	Noch keine Verwendung.
long	Noch keine Verwendung.
native	Noch keine Verwendung.
new	Deklaration von Objekten.
null	Datentyp null ist ein Sonderfall einer Variablen.
package	Noch keine Verwendung.
private	Noch keine Verwendung.
protected	Noch keine Verwendung.
public	Noch keine Verwendung.
return	Rückgabewert in einer Funktion.
short	Noch keine Verwendung.
static	Noch keine Verwendung.
super	Noch keine Verwendung.
switch	Bedingte Anweisung.
synchronized	Noch keine Verwendung.
this	Bezug auf aktuelles Objekt.
throw	Noch keine Verwendung.
throws	Noch keine Verwendung.
transient	Noch keine Verwendung.
true	Boole'scher Wert.
try	Noch keine Verwendung.
typeof	Datentyp angeben.
var	Explizite Variablendeklaration.
void	Noch keine Verwendung.
while	Schleife mit while.
with	Anweisung mit aktuellem Objekt durchführen.

Die Schlüsselwörter dieser Liste wurden mit dem Navigator 4.7 und dem Internet Explorer 5.0 unter MS Windows 98 getestet. Alle Schlüsselwörter dürfen in dem Navigator nicht verwendet werden, ein Teil der Schlüsselwörter wird von dem Internet Explorer jedoch akzeptiert. Wie sich der Fehler äußert, hängt von dem verwendeten Browser und seinen Einstellungen ab. Möglich sind Meldungen über Laufzeitfehler wie eine (scheinbare) Ausführung ohne Fehlermeldung – im letzteren Fall wird das Programm eben wegen des Fehlers gar nicht ausgeführt und bringt keine Fehlermeldung.

Weiter sollten alle Namen von Objekten, Eigenschaften, Methoden und Funktionen in JavaScript nicht verwendet werden. Zwar generieren nicht alle dort benutzten Wörter einen Fehler, trotzdem ist von ihrem Gebrauch abzuraten.

Schlüsselwörter testen

Für den Text von Schlüsselwörtern kann folgendes Programm verwendet werden. Die Variable `abc` wird jeweils durch das zu testende Schlüsselwort ersetzt:

```
<html>

<head>
<title>Schlüsselwörter</title>
<script language="JavaScript">
var abc = "Diese Variable ist kein Schlüsselwort!";
alert(abc);
</script>
</head>

<body>
</body>

</html>
```

Ist die Variable kein Schlüsselwort, erscheint die Meldung „Diese Variable ist kein Schlüsselwort!“. Wie oben erwähnt, sollte der Test wenigstens mit dem Navigator und dem Internet Explorer durchgeführt werden.

Operatoren

... für Text

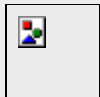
Mehrere Zeichenketten werden mit dem Operator + zu einer Zeichenkette zusammengefügt.

Beispiel

```
var text1 = "Aller Anfang ";  
var text2 = "ist schwer!";  
var summe = text1 + text2; // summe = Aller Anfang ist schwer!
```

Da der Operator + ebenfalls für die Addition von Zahlen zuständig ist, gilt folgende Regel:

- Zahl + Zahl ergibt eine Zahl
- Zeichenkette + Zeichenkette ergibt eine Zeichenkette
- Zahl + Zeichenkette ergibt eine Zeichenkette



Operatoren

- Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:
 - Vergeben Sie als Titel der HTML-Seite „Operatoren“.
 - Kopieren Sie den Quelltext ohne den Kommentar aus dem Beispiel oben.
 - Geben Sie die Variable „summe“ über den »alert()«-Befehl aus, wenn die HTML-Seite gestartet wird.

- Vergleichen Sie Ihre Lösung mit dem [Muster b_0008.htm](#).

... für Zahlen

JavaScript stellt verschiedene mathematische Berechnungen zur Verfügung, darunter die vier Grundrechenarten:

- addieren durch den Operator +
- subtrahieren durch den Operator -
- multiplizieren durch den Operator *
- dividieren durch den Operator /

Bei der Berechnung gelten die gleichen Regeln wie in der Mathematik:

- Punktrechnung geht vor Strichrechnung
- ein Ausdruck in runden Klammern hat Vorrang

Zur Berechnung des Rests einer Division (Modulo-Division) wird der Operator % verwendet, z.B. ergibt `3 % 2` // summe hat Rest 1.

Weitere Funktionen (Sinus, Cosinus etc.) stehen als Methoden des »Math«-Objekts zur Verfügung, mathematische Konstanten (Pi etc.) als Eigenschaften.

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 + zahl_2; // Addition: summe = 115
```

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 - zahl_2; // Subtraktion: summe = -5
```

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 * zahl_2; // Multiplikation: summe = 3300
```

Beispiel

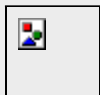
```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 / zahl_2; // Division: summe = 0.92
```

Beispiel

```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = zahl_1 + zahl_2 * 10 ; // Punktrechnung geht vor
Strichrechnung: summe = 655
```

Beispiel


```
var zahl_1 = 55;
var zahl_2 = 60;
var summe = (zahl_1 + zahl_2) * 10 ; // Die Klammer hat Vorrang: summe =
1150
```



Operatoren

 Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:

- Vergeben Sie als Titel der HTML-Seite „Operatoren“.
- Kopieren Sie den Quelltext ohne den Kommentar aus dem letzten Beispiel oben.
- Geben Sie die Variable „summe“ mit einem beschreibenden Text der mathematischen Operation über den »alert()«-Befehl aus, wenn die HTML-Seite gestartet wird.

 Vergleichen Sie Ihre Lösung mit dem [Muster b_0009.htm](#).

... für logische Verknüpfungen

Logische Operatoren dienen zur Berechnung der Boole'schen Werte true und false. Eine logische Operation liefert als Ergebnis wieder einen Boole'schen Wert.

Das logische UND verknüpft wenigstens zwei Bool'sche Werte:

false && false ergibt false

false && true ergibt false

true && false ergibt false

true && true ergibt true

Das logische ODER verknüpft wenigstens zwei Bool'sche Werte:

false || false ergibt false

false || true ergibt true

true || false ergibt true

true || true ergibt true

Das logische NICHT betrifft nur einen Boole'schen Wert:

!false ergibt true

!true ergibt false

Verzeichnis der Operatoren

In der folgenden Tabelle stehen die gültigen Operatoren in JavaScript:

Operator	Bedeutung
$x+=y$	$x=x+y$
$x-=y$	$x=x-y$
$x*=y$	$x=x*y$
$x/=y$	$x=x/y$
$x\%=y$	$x=x\%y$ liefert den Restwert einer Teilung.
$x++$	$x=x+1$
$x--$	$x=x-1$
$y=x++$	$y=x$ und danach $x=x+1$
$y=++x$	$x=x+1$ und danach $y=x$
$y=x--$	$y=x$ und danach $x=x-1$
$y=--x$	$x=x-1$ und danach $y=x$
$-x$	$x=-x$
$\&\&$	Logisches UND.
$\ \ $	Logisches ODER.
$!$	Logisches NICHT.
$==$	Vergleichsoperator »ist-gleich«.
$!=$	Vergleichsoperator »ist-nicht-gleich«.
$>$	Vergleichsoperator »größer-als«.
$<$	Vergleichsoperator »kleiner-als«.
$>=$	Vergleichsoperator »größer-gleich«.
$<=$	Vergleichsoperator »kleiner-gleich«.

Quiz Operatoren I

Welche Antworten sind richtig?

- a) Für die vier Grundrechenarten dienen die vier Operatoren „+“ und „-“ und „*“ und „/“.
- b) Zur Berechnung des Rests einer Division wird der Operator „%“ verwendet.
- c) Das Symbol „+“ ist ein gültiger Operator auch für Zeichenketten.
- d) Das Symbol „+“ ist ein gültiger Operator auch für Boole'sche Werte.
- e) `var summe = 55 + 60 * 10 ;` ergibt 655.

Quiz Operatoren II

Welche Antworten sind richtig?

- a) Zeichenkette + Zeichenkette ergibt eine Zeichenkette.
- b) Zahl + Zahl ergibt eine Zahl.
- c) Zahl + Zeichenkette ergibt eine Zahl.
- d) `var meldung = "Das ist " + "richtig!"` ist eine gültige Operation.
- e) `var meldung = "Das ist " + "55!"` ist eine gültige Operation.

Sonderzeichen

Escape-Sequenzen

JavaScript enthält Sonderzeichen, mit denen Zeichen oder Formatierungen in Zeichenfolgen eingefügt werden können, die nicht direkt eingegeben werden können. Die Sonderzeichen beginnen mit einem umgekehrten Schrägstrich (Backslash). Mit diesem sogenannten Escape-Zeichen wird darauf verwiesen, dass es sich bei dem folgenden Zeichen um ein Sonderzeichen handelt.

Escape-Sequenz	Zeichen
\b	Rückschritt.
\f	Seitenvorschub.
\n	Zeilenvorschub (neue Zeile).
\r	Wagenrücklauf.
\t	Horizontaler Tabulator.
\'	Einfaches Anführungszeichen.
\"	Doppeltes Anführungszeichen
\\	Umgekehrter Schrägstrich (Backslash).

Mit Hilfe der Escape-Sequenzen kann der Text zwischen den Tags `<pre>` und `<xmp>` und den Meldefenstern mit der »`alert()`«, »`confirm()`«- und »`prompt()`«-Methode gesteuert werden.

Objektbasierte Struktur

Hierarchische Struktur

JavaScript bezeichnet die einzelnen Datenelemente als Objekte und ordnet sie bis auf einige Ausnahmen in einer hierarchischen Struktur ein. Das wichtigste Objekt ist das Fenster des Browsers, »window« genannt. Es ist selbst keinem Objekt untergeordnet, verfügt aber über untergeordnete Objekte.

Mit Hilfe des »window«-Objektes wird das Fenster des Browsers kontrolliert. »document« ist ein Unterobjekt des »window«-Objektes und der Anzeigebereich innerhalb des Fensters, in dem die HTML-Seite angezeigt wird.

Für alle Unterobjekte, Eigenschaften und Methoden des »window«-Objektes kann das „window“ weggelassen werden kann. Statt `window.alert("Herzlich Willkommen");` reicht einfach `alert("Herzlich Willkommen");`

Ein Objekt bezeichnet allgemein ein abstraktes Modell (z.B. der Mensch). Tatsächlich verwendet werden die sogenannten Instanzen eines Objekts (z.B. meine Freundin). Es ist üblich, von Objekten zu sprechen, wenn Instanzen gemeint sind.

Eine Objektinstanz wie »window« existiert automatisch und kann mehrfach (mehrere geöffnete Fenster) vorhanden sein – hier werden die einzelnen Instanzen anhand des Namens der Instanz identifiziert. Eine Objektinstanz wie »Screen« existiert in JavaScript ebenfalls automatisch, ist jedoch einmalig. Eine Objektinstanz wie »Array« muss erst erzeugt werden, um damit arbeiten zu können.

Bei dem Objekt »window« steht ein Teil des Fensters für die Anzeige der HTML-Seite zur Verfügung. Dieser Bereich wird über das untergeordnete Objekt namens »document« angesprochen. Um ein Unterobjekt in der Hierarchie ansprechen zu können, wird es mit einem Punkt getrennt an das übergeordnete Objekt angehängt. In diesem Fall also »window.document«.

Die HTML-Seite (das Dokument) selbst kann weitere Unterobjekte beinhalten. Ein Datenelement kann z.B. ein Formular sein, das als Unterobjekt »forms« definiert ist. Das Formular enthält selbst wieder Datenelemente, also weitere untergeordnete Objekte. In JavaScript gibt es dafür das Unterobjekt »elements«. Es ist denkbar, dass in einem Formular mehrere (hierarchisch gleichgestellte) Elemente vorkommen, wie z.B. mehrere Eingabefelder. Deswegen werden die einzelnen Unterobjekte durchnummeriert (die Zählung beginnt bei 0) und sind somit eindeutig identifizierbar. Sofern die

Objekte einen eindeutigen Namen tragen, kann jedes Objekt ebenso über den Namen angesprochen werden. Da in einer HTML-Seite mehrere (hierarchisch gleichgestellte) Formulare vorkommen können, gilt die Namensregelung ebenfalls für die Formulare.

In der Hierarchie lautet die korrekte Ansprache also `window.document.forms[0].elements[0]`, wenn das erste Formular mit dem ersten Element in der HTML-Seite gemeint ist. Die beiden Anweisungen sind äquivalent, sofern der Name des Formulars „Bestellformular“ lautet und der Name des Eingabefeldes „F1“:

```
var menge_1 = window.document.Bestellformular.F1.value;
var menge_1 = window.document.forms[0].elements[0].value;
```

Ein weiteres Objekt mit einer hierarchischen Struktur hat den Namen »navigator« und gibt Informationen über den verwendeten Browser. Die Beziehung zwischen den beiden Objekten »window« und »navigator« kann einerseits als unabhängig bezeichnet werden – beide Objekte stehen zueinander in keinem Unter- oder Überordnungsverhältnis. Andererseits werden die Eigenschaften und Methoden des »navigator«-Objekts nur dann zur Anwendung kommen, wenn sie im Rahmen des »window«-Objekts benutzt werden können. Diese Eigenart gilt ebenfalls für die folgenden Ausnahmen.

Ausnahmen

Neben den hierarchisch geordneten Objekten gibt es Ausnahmen, die nicht in die Hierarchie passen, z.B. Objekte für Datums- und Zeitangaben, für mathematische Berechnungen etc. Diese Objekte sind kein Unterobjekt eines anderen Objekts und besitzen selbst keine Unterobjekte:

- Array
- Boolean
- Date
- Function
- Global
- Math
- Number
- Object
- RegExp
- Screen
- String

Eine weitere Ausnahme stellen eigenständig erstellte Objekte dar.

Eigenschaften und Methoden

In der Regel verfügen Objekte über Eigenschaften und Methoden. Eine Eigenschaft ist ein konkreter Wert. Eine Methode ist eine **Funktion**, die an ein bestimmtes Objekt gebunden ist. (Das Beispielobjekt „Mensch“ verfügt über Eigenschaften wie einen Namen, eine Größe, ein Gewicht etc. Die Methode, das Gewicht abzufragen, besteht darin, auf die Waage zu steigen.) JavaScript unterstützt neben den integrierten (hierarchischen und nicht hierarchischen) Objekten auch eigenständig erstellte Objekte.

Eigenschaften

Ein Objekt kann über vorgegebene Eigenschaften verfügen. Die Eigenschaft wird mit einem Punkt an das Objekt angehängt. Eine Eigenschaft des »window.document«-Objekts ist die Hintergrundfarbe »bgColor«. Die Eigenschaft kann z.B. in eine Variable geschrieben werden:

```
var farbe = window.document.bgColor;
```

Alle Eigenschaften können gelesen und angezeigt werden. Die Hintergrundfarbe wird z.B. mit der folgenden Anweisung angezeigt:

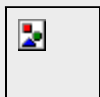
```
alert(window.document.bgColor);
```

Das »Math«-Objekt hat z.B. eine Eigenschaft »Math.PI«. Damit steht die mathematische Konstante „Pi“ für Berechnungen zur Verfügung, ohne deren genauen Wert zu kennen. Die Konstante wird z.B. mit der folgenden Anweisung angezeigt:

```
alert(Math.PI);
```

Einige Eigenschaften können nicht nur gelesen, sondern auch geändert werden. Um eine Eigenschaft zu ändern, wird ihr ein Wert zugewiesen. Die mathematische Konstante „Pi“ kann nicht geändert werden, die Hintergrundfarbe einer HTML-Seite schon. Durch die folgende Anweisung wird der Hintergrund schwarz gesetzt:

```
window.document.bgColor = "000000";
```



Eigenschaften von Objekten



Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:

- Vergeben Sie als Titel der HTML-Seite „Eigenschaften von Objekten“.
- Setzen Sie den Hintergrund der HTML-Seite über eine JavaScript-Anweisung schwarz.



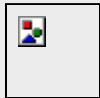
Vergleichen Sie Ihre Lösung mit dem [Muster b_0010.htm](#).

Methoden

Ein Objekt kann Methoden besitzen. Methoden sind mit frei definierten Funktionen vergleichbar, sie sind jedoch (im Gegensatz zu den frei definierten Funktionen) an ein konkretes Objekt gebunden. Methoden sind Aktionen, die im direkten Zusammenhang mit dem Objekt stehen. Das »window«-Objekt verwendet die »alert()«-Methode, um eine Meldung auszugeben. Um eine Methode aufzurufen – also die Aktion ausführen zu lassen – wird die Methode mit einem Punkt das entsprechende Objekt angehängt. Die Anweisung lautet dann:

```
window.alert();
```

Methoden haben am Ende immer runde Klammern, in denen bei vielen Methoden noch zusätzliche Informationen angegeben werden müssen, sogenannte Argumente. Je nach dem, was die Methode bewirkt, kann sie ein Ergebnis zurückliefern.



Methoden von Objekten

- Erstellen Sie eine HTML-Seite mit folgenden Bedingungen:
- Vergeben Sie als Titel der HTML-Seite „Methoden von Objekten“.
 - Geben Sie eine Meldung mit dem Text „Das ist eine Methode“ aus.

Vergleichen Sie Ihre Lösung mit dem [Muster b_0011.htm](#).

Quiz Objekte I

Sozusagen von Subjekt zu Objekt: Welche Aussagen sind korrekt?

- a) Die einzelnen Datenelemente werden in JavaScript als Objekte bezeichnet.
- b) Die Struktur ist grundsätzlich hierarchisch ausgelegt.
- c) Ein Objekt bezeichnet allgemein ein abstraktes Modell.
- d) Tatsächlich verwendet werden die Instanzen eines Objekts.
- e) Es ist üblich, von Objekten zu sprechen, auch wenn Instanzen gemeint sind.

Quiz Objekte II

Welche Aussagen sind objektiv korrekt?

- a) Objektinstanzen stehen teilweise automatisch zur Verfügung und müssen teilweise erst erzeugt werden.
- b) Ein Objekt wie »window« existiert automatisch.
- c) Bei dem Objekt »window« steht ein Teil des Fensters für die Anzeige des HTML-Dokuments zur Verfügung.
- d) Objekte können nicht automatisch existieren.
- e) Automatisch existierende Objekte stehen in keiner Hierarchie.

Quiz Objekte III

Welche Aussagen sind objektiv korrekt?

- a) Objekte verfügen nicht immer über Eigenschaften und Methoden.
- b) JavaScript unterstützt eigenständig erstellte Objekte.
- c) JavaScript unterstützt nur hierarchische Objekte.
- d) Eigenschaften können nur gelesen und nicht gesetzt werden.
- e) Die Eigenschaft wird mit einem Semikolon an das Objekt angehängt.