

Ergänzungen

Hinweise

Ulrich Paasch

Christian Moritz

Korrekturen

Jochem Ottersbach

Klemens Kieslinger

Annette Mörsberger

Hans Martens

Informationen verbreiten

**Medien gestalten
und herstellen**

2. Auflage

Die folgenden Ergänzungen, Hinweise und Korrekturen dienen zur Aktualisierung der zweiten Auflage (2004) von „Informationen verbreiten“. In der dritten Auflage (2006) sind diese – und zahlreiche weitere – Aktualisierungen bereits enthalten.

© 2005 by Verlag Beruf und Schule, Postfach 2008, 25510 Itzehoe, Germany

Ergänzungen

Zu 1.4.3 Die Zentraleinheit

Chipsatz und Bus

Der Chipsatz befindet sich auf dem Motherboard und besteht unter anderem aus Northbridge und Southbridge. Eine physikalische Trennung in zwei Bausteine ist jedoch nicht unbedingt notwendig, beide Teile können also in einem Chip untergebracht sein.

Der Chipsatz ist für die Ansteuerung elementarer Computer-Komponenten verantwortlich und regelt den Datentransport zwischen ihnen, zum Beispiel die Übermittlung der Daten zwischen Prozessor und Arbeitsspeicher oder Grafikkarte. Bedeutende Chipsatz-Hersteller sind u. a. Intel und VIA.

Die Northbridge synchronisiert und steuert den Datentransfer zwischen Prozessor (CPU), Arbeitsspeicher (RAM), Cache und AGP- oder PCI-Express-Grafikadapter. Dabei ist der Chip über einen 32-Bit-Bus (seit 2003 auch 64 Bit Busbreite) mit den Komponenten verbunden. Die Verbindung zwischen Northbridge und CPU wird Front-Side-Bus genannt.

Die Southbridge steuert den Datenaustausch mit peripheren Geräten über PCI-, ISA-, EIDE-, S-ATA-, Firewire- oder USB-Schnittstellen. South- und Northbridge sind entweder ebenfalls per PCI oder über eine proprietäre Schnittstelle (wie zum Beispiel VIAs V-Link) verbunden.

Der Front-Side-Bus, kurz FSB, ist die Direktverbindung zwischen Prozessor und North-Bridge. Die Hardware-Schnittstelle ist ein Sockel oder Slot, je nach Bauform des Prozessors. Die Taktfrequenz und die Busbreite des FSB bestimmt die Geschwindigkeit, mit der die Daten vom bzw. zum Prozessor kommen.

Die Signale auf dem Front-Side-Bus laufen je nach Prozessortyp mit 66, 75, 83, 95, 100, 133, 166 oder 200 MHz. Die maximale Transferrate zwischen Prozessor und dem Rest des Systems ist durch den Systemtaktgeber festgelegt.

Das Double-Data-Rate-Verfahren (DDR) verdoppelt die Übertragungsraten, indem pro Takt zwei Datenwörter übertragen werden. Dieses Verfahren wird auch beim AGP-Steckplatz und Arbeitsspeicher (DDR-SDRAM) eingesetzt. Mit dem Pentium 4 geht Intel noch einen Schritt weiter und überträgt gleich vier Datenwörter (Quad-Data-Rate, QDR) pro Takt, der dazu passende Arbeitsspeicher heißt DDR-II.

Einige Front-Side-Bus Varianten

FSB-Takt	Datenwörter Pro Takt		
	1	2	4
66 MHz	FSB66	FSB133	FSB266
75 MHz	FSB75	FSB150	FSB300
100 MHz	FSB100	FSB200	FSB400
133 MHz	FSB133	FSB266	FSB533
166 MHz	FSB166	FSB332	FSB664
200 MHz	FSB200	FSB400	FSB800

Der von Intel 1993 entwickelte PCI-Bus (Peripheral Component Interconnect) ist Industriestandard und fester Bestandteil von Intel-kompatiblen PCs sowie Apple Macintoshs. Das PCI-BIOS unterstützt durch Plug and Play die automatische Erkennung und Konfiguration von PCI-Steckkarten. Die Standards sind: PCI, PCI-X, PCI-Express (PCIe) sowie Mini-PCI und Mini-PCI Express.

Das herkömmliche PCI arbeitet durchweg in PCs mit 32 Bit Busbreite und 33 MHz oder 66 MHz Taktfrequenz, in Servern und Workstations auch mit 64 Bit und 66 MHz. Diese Werte bestimmen die Übertragungsraten; sie liegen zwischen 0,13 GByte/s und 0,5 GByte/s. Allerdings sinkt die Rate erheblich, wenn sich bis zu sechs (bei 33 MHz) bzw. drei (bei 66 MHz) Geräte einen PCI-Bus teilen. Schnelle Gigabit-Ethernetkarten oder RAID-Festplattensysteme werden da schon mal ausgebremst.

PCI-X (entwickelt seit 1998) ist eine 64-Bit-Variante mit 66, 100 oder 133 MHz Taktfrequenz. Es beherrscht eine ECC-Fehlerkorrektur und wird in Hochleistungs-PCs (u. a. Apple G5), Servern und Workstations eingesetzt. Die Übertragungsraten können durch Double- und Quad-Data-Rate-Übertragungsverfahren verdoppelt oder vervierfacht werden; sie betragen 0,5 bis 8,5 GByte/s. Ab 133 MHz Taktfrequenz darf nur noch ein Gerät pro Bus angeschlossen sein. In der Praxis werden extreme Anforderungen an die Übertragungsrate zum Beispiel durch mehrkanalige LWL-Netzwerke (Lichtwellenleiter) gestellt. Ein PCI-X-Steckplatz ist nur abwärtskompatibel zum herkömmlichem PCI, wenn die Steckkarte mit 3,3 V und 66 MHz funktioniert.

Der Trend geht eindeutig zu schnellen seriellen Verbindungen, die skalierbar sind und Hotplug unterstützen, siehe USB und Firewire. Auch das neue PCI-Express (PCIe, 2004) arbeitet seriell, es ist daher hardwareseitig nicht kompatibel zu PCI und PCI-X, benutzt aber deren Signalisierungs- und Programmier-techniken. Auf einer Leitung (PCIe x1) können 0,25 GByte/s gleichzeitig in beide Richtungen (voll duplex) übertragen werden und im Gegensatz zu PCI und PCI-X muss diese Bandbreite nicht mit anderen Geräten geteilt werden. Dabei sind die Leistungen skalierbar. Eine Hochleistungsvariante mit 16 Kanälen schafft bis 4 GByte/s voll duplex und heißt PCIe x16. Sie dient zum Anschluss schneller Grafikkarten und soll den etablierten AGP-Standard ablösen.

Ein typisches Mainboard bietet dann zum Beispiel folgende PCI-Steckplätze an: einmal PCIe x16, zwei- bis dreimal PCIe x1 (oder x2 oder x4) sowie drei bis fünf herkömmliche PCI oder PCI-X.

Mini-PCI ist die Variante für Notebooks und sonstige Mobilrechner. Mit einer Übertragungsleistung von 0,12 GByte/s ist sie in die Jahre gekommen und wird von Mini-PCI-Express abgelöst. Auch hier bringt der Umstieg von paralleler auf serielle Technik eine deutliche Leistungssteigerung auf 0,6 GByte/s.

Zu 6.3.2 Workflow-Konzepte

PDF/X

Das Portable Document Format (PDF) hat heute in der Druckvorstufe zwei Funktionen: Es ist Datenaustauschformat bei der Übergabe zum Beispiel vom Auftraggeber an die Druckerei und zugleich Produktionsdatenformat im Ausgabe-Workflow.

Die vom Auftraggeber gelieferten PDF-Dateien sollten möglichst so beschaffen sein, dass sie ohne zeit- und kostenaufwändige Konvertierungen oder Reparaturen („PDF-Basterei“) in den Ausgabe-Workflow übernommen werden können. Sie müssen also alles enthalten, was für den Ausgabe-Workflow erforderlich ist, und dürfen nichts enthalten, was ihn stört.

Diese Bedingungen sind in den internationalen Standards PDF/X-1a, PDF/X-2 und PDF/X-3 festgelegt; das X steht für *Exchange*, also (Daten-)Austausch. In Westeuropa wird überwiegend mit PDF/X-3 gearbeitet.

Wesentliche Gemeinsamkeiten und Unterschiede:

- PDF/X-1a und PDF/X-3 regeln den vollständigen digitalen Datenaustausch (*complete exchange*, *blind exchange*); alle Daten, die zur Ausgabe gebraucht werden, müssen in die PDF-Datei integriert sein.
- PDF/X-2 regelt den unvollständigen digitalen Datenaustausch (*partial exchange*, *open exchange*); hier sind auch Verweise zum Beispiel auf Bilder oder Schriften möglich, die nicht Bestandteil der PDF-Datei sind.
- PDF/X-1a erlaubt nur CMYK und Sonderfarben.
- PDF/X-2 und PDF/X-3 lassen neben CMYK und Sonderfarben auch RGB-Farbräume und CIELAB zu.

Abgesehen von den zugelassenen Farbräumen stimmen die Spezifikationen PDF/X-1a und PDF/X-3 weitgehend überein:

- Alle Bilder und Schriften müssen als Objekte in der PDF-Datei enthalten sein.
- Das beschnittene Endformat (*TrimBox*) muss definiert sein, das unbeschnittene Format (*BleedBox*), soweit es produktionstechnisch erforderlich ist.
- Der Überfüllungsschlüssel (*Trapped Key*) muss auf „ja“ oder „nein“ (*true* oder *false*) gesetzt sein; der Wert „unbekannt“ (*unknown*) ist nicht erlaubt.
- Die Ausgabebedingungen und das entsprechende ICC-Profil müssen im *OutputIntend Dictionary* angegeben bzw. eingebettet sein.

- Bei Verwendung von RGB oder CIELAB in PDF/X-3-Dateien muss auch das entsprechende ICC-Profil eingebettet sein.

Unzulässig sind nach beiden Spezifikationen:

- Kommentare innerhalb des beschnittenen bzw. unbeschnittenen Seitenformats
- Verschlüsselungen
- LZW- und JBIG2-Komprimierung
- Transferkurven, also Funktionen, die bei der Ausgabe im RIP Tonwertveränderungen zur Anpassung an ein Ausgabegerät auslösen

Rastereinstellungen sind zwar erlaubt, aber nicht verbindlich für die Ausgabe.

Wichtig ist auch die Wahl der richtigen PDF-Version. Es darf nicht einfach die jeweils neueste (zurzeit PDF1.6) benutzt werden, da die entsprechenden Tools zur Verarbeitung im Ausgabe-Workflow immer mit einer gewissen zeitlichen Verzögerung auf den Markt kommen und auch dann nicht sofort in allen Betrieben vorhanden sind.

Zurzeit werden die PDF-Versionen 1.3 (X-1a:2001, X-2:2002, X-3:2002) und 1.4 (X-1a:2003, X-2:2003, X-3:2003) verwendet.

Der Weg vom offenen Layout-Dokument zur PDF/X-Datei geht in drei logischen Schritten vor sich, die einzeln oder in einem automatisierten Ablauf durchgeführt werden:

- Erzeugung einer PostScript Datei mithilfe des PostScript-Treibers
- Umwandlung in PDF-Datei mit eingebetteten Schriften mit dem Acrobat-Distiller oder einem anderen PDF-Erzeugungsprogramm
- Prüfung der PDF-Datei und Umwandlung in PDF/X, zum Beispiel mit dem PDF-Inspektor

Der PDF-Inspektor ist Bestandteil von Adobe Acrobat Professional; eine funktionstüchtige Demo-Version gibt es unter www.callas.de. Der PDF-Inspektor enthält neben der eigentlichen PDF/X-Funktionalität auch eine Reihe von Preflight-Funktionen, mit denen sich unter anderem die Auflösungen der enthaltenen Pixelbilder, ihre Farbmodi und die Anzahl der Farbauszüge überprüfen lassen. „Kochrezepte“ für die Umwandlung von offenen Layout-Dokumenten in PDF/X-Dateien gibt es unter www.eci.org.

PDF/X-Spezifikationen

PDF/	ISO	PDF-Version	Datenaustausch	Farbräume
X-1a:2001	15930-1	1.3	vollständig	CMYK, Sonderfarben
X-1a:2003	15930-4	1.4		
X-2:2002	DIS 15930-2*	1.3	unvollständig	CMYK, Sonderfarben, RGB, CIELAB
X-2:2003	15930-5	1.4		
X-3:2002	15930-3	1.3	vollständig	CMYK, Sonderfarben, RGB, CIELAB
X-3:2003	15930-6	1.4		

* Entwurf, nicht als ISO-Norm verabschiedet

JDF

JDF (*Job Definition Format*) ist zunächst ein umfassendes, hersteller-, programm- und plattformunabhängiges Job-Ticket-Format für den gesamten Workflow von der Druckvorstufe (*prepress*) über Druck (*press*) und Druckweiterverarbeitung (*postpress*) bis zur Auslieferung (*delivery*). JDF basiert auf der Meta-Sprache XML (*Extensible Markup Language*, vgl. „Informationen verbreiten“, Abschnitt 10.5.1)

Die JDF-Spezifikation geht aber über die bloße Definition eines Job-Ticket-Formats hinaus: Sie beschreibt zugleich das Konzept zur Vernetzung von Systemkomponenten und zur Automatisierung von Produktionsprozessen. Auch kaufmännische und betriebswirtschaftliche Aufgaben wie Angebotserstellung, Kalkulation und Kostenrechnung können integriert werden.

Zurzeit aktuelle JDF-Spezifikation ist die Version JDF 1.2. Sie wird von der internationalen Organisation CIP4 (*International Cooperation for the Integration of Processes in Prepress, Press and Postpress*, www.cip4.org) weiterentwickelt.

Die JDF-Spezifikation ist zu umfangreich und komplex, um ihren wesentlichen Inhalt auch nur annähernd vollständig in knapper und nachvollziehbarer Form darzustellen. Deshalb soll nur in aller Kürze auf die Grundprinzipien der JDF-Jobbeschreibung und des JDF-Workflows eingegangen werden. Die JDF-Jobbeschreibung besteht aus baumartig (hierarchisch) angeordneten Knoten (*Nodes*).

- Produktknoten beschreiben End- oder Teilprodukte wie zum Beispiel Buch, Buchblock, Buchdecke.
- Prozessgruppenknoten fassen Einzelprozesse zu Gruppen zusammen.
- Prozessknoten beschreiben Einzelprozesse wie zum Beispiel Ausschießen, Druckplattenbebilderung, Druck oder Falzen.

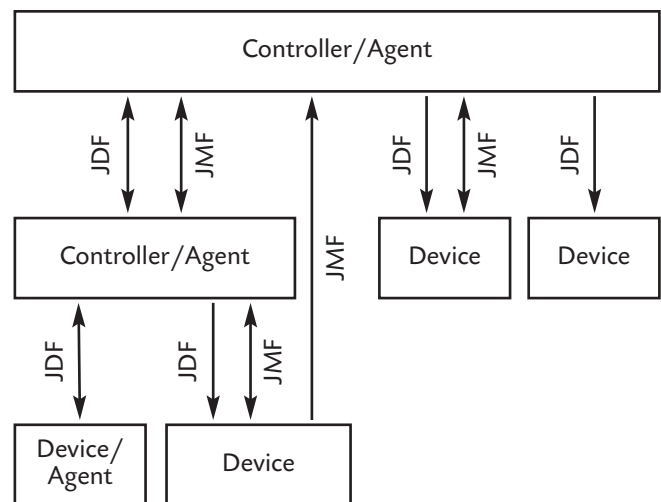
Die Verbindungen von aufeinander folgenden Prozessen werden durch Ressourcen (*Resources*) repräsentiert. Ressource ist der Oberbegriff für alle Outputs von Prozessen, die – soweit es sich nicht um Endprodukte handelt – als Inputs in andere Prozesse einfließen. Jeder Prozess konsumiert und erzeugt Ressourcen; das können sowohl materielle Produkte (zum Beispiel Druckplatte, Druckbogen) als auch Daten (zum Beispiel überfülltes Dokument, Belichter-Bitmap) oder Prozessparameter (zum Beispiel Ausschießschema, Farbzoneneinstellung) sein.

Software zur Steuerung und Überwachung des gesamten Workflows wird in der Terminologie der JDF-Spezifikation als *Management Information System (MIS)* bezeichnet. MIS sind also – um einen anderen Begriff zu verwenden – Workflow-Management-Programme.

Zum JDF-Workflow gehören vier logische Komponenten:

- Agenten erzeugen und modifizieren JDF.
- Controller empfangen JDF, wählen Geräte für bestimmte Aufgaben aus und reichen JDF an sie weiter. Controller können zugleich als Agenten fungieren, also auch JDF erzeugen oder modifizieren.
- Geräte (*Devices*) empfangen JDF, interpretieren es und führen die Anweisungen entweder selbst aus oder steuern Maschinen. Auch Geräte können zugleich Agenten sein.
- Maschinen (*Machines*) sind nicht JDF-fähige Hard- oder Software, die von JDF-Geräten mit maschineneigenen Anweisungen gesteuert werden.

Controller und Geräte kommunizieren untereinander über das *Job Messaging Format (JMF)*. Mittels JMF meldet das Gerät zum Beispiel seine Bereitschaft oder Beschäftigung an den Controller; bei der Einrichtung eines neuen Geräts fragt der Controller ab, welche Prozesse das Gerät ausführen kann. JMF basiert, ebenso wie JDF, auf der Meta-Sprache XML (vgl. Abschnitt 10.5.1). Die Grafik am Fuß dieser Spalte zeigt einige Varianten der Zusammenarbeit von Agenten, Controllern und Geräten.



Beispiele für Interaktionen von Agenten, Controllern und Devices mittels JDF und JMF

Zu 10.4 JavaScript und DOM

10.4.5 Variablen und Variablentypen

Mit Variablen sind Speicherbereiche bezeichnet, in denen Inhalte hinterlegt werden. Der Inhalt einer Variablen lässt sich jederzeit ändern und wird auch als Wert der Variablen bezeichnet. Die Speicherbereiche lassen sich durch zuvor definierte Variablennamen ansprechen.

Variablen können unterschiedliche Variablentypen enthalten: Zahl (*number*), Zeichenkette (*string*), Wahrheitswert (*boolean*), Funktion (*function*), Objekt (*object*) – oder sie sind undefiniert. Es gibt globale und lokale Variablen, die sich durch ihre Gültigkeitsbereiche unterscheiden. Globale Variablen haben ihre Gültigkeiten über alle Skript-Bereiche innerhalb der gesamten Datei, während lokale Variablen nur innerhalb jeweils einer Funktion gültig sind. Globale Variablen werden in der Regel am Anfang des Skript-Bereichs deklariert, lokale mit dem Schlüsselwort `var` innerhalb von Funktionen. Werden Variablen ohne das Schlüsselwort `var` innerhalb von Funktionen deklariert, sind sie global. Bei der Deklaration von globalen Variablen außerhalb von Funktionen ist das Schlüsselwort `var` optional. Beispiele:

- `var Tier="Der Hund bellt";`
Mit `var` wird eine Variable mit dem Variablennamen `Tier` definiert, deren Wert aus der Zeichenkette "Der Hund bellt" besteht. Zeichenketten stehen immer in Anführungszeichen. Die Variable `Tier` ist vom Typ *string*.
- `x=10;`
Hier wird der globalen Variablen `x` der Zahlenwert 10 zugewiesen. Die Variable `x` ist von Typ *number*.
- `var LichtLeuchtet=true;`
Wahrheitswerte haben nur zwei Zustände: `true` (wahr, zutreffend) oder `false` (falsch, unzutreffend). Die Variable `LichtLeuchtet` ist von Typ *boolean*.
- `var Grafik=new Image();`
Mit dem Schlüsselwort `new` wird eine Instanz des vordefinierten `Image`-Objekts erzeugt. Die Variable `Grafik` ist vom Typ *object*.

Die Variablentypen werden in JavaScript nicht ganz so streng gehandhabt, wie es in anderen Programmiersprachen üblich ist. Variablen vom Typ *number* lassen sich zum Beispiel direkt im Browser anzeigen, ohne sie vorher in Zeichenketten zu konvertieren.

Eingaben aus Formulareinträgen werden immer wie Zeichenketten (*strings*) behandelt. Um Rechenoperationen mit Zahlenwerte aus Formulareinträgen durchzuführen, müssen sie erst in den Variablentyp *number* konvertiert werden. Für die Typumwandlung gibt es vordefinierte, objektunabhängige Funktionen.

Mit `x=12.4;` (*number*) lassen sich sofort Rechenoperationen durchführen, mit `x="12.4";` (*string*) dagegen nicht. Die objektunabhängige Funktion `parseFloat()` wandelt die Zeichenkette in eine Zahl um und gibt diese als numerischen Wert zurück. Berücksichtigt werden auch Gleitkommazahlen mit Dezimalpunkt. Beispiel: `var Zahl=parseFloat("12.4");`

10.4.6 Operatoren

Operatoren werden für Berechnungen, Vergleiche, Verknüpfungen und Wertzuweisungen benötigt. Eine Operation kombiniert in der Regel zwei Operanden und ermittelt den Ergebniswert. Operatoren gibt es zum Beispiel für:

- Wertzuweisungen (`=`)
Beispiele: `z=137;` `Titel="Informationen verbreiten";`
- Berechnungen: Addition (`+`), Subtraktion (`-`), Multiplikation (`*`), Division (`/`), Modulo-Division (`%`)
Beispiele: `x=a+b/c;` `z=12*(a+b);`
Für Additionen und Subtraktionen gibt es verkürzte Notationen:
`i++;` steht für `i=i+1;`
`k--;` steht für `k=k-1;`
`x+=5;` steht für `x=x+5;`
`z-=2;` steht für `z=z-2;`
Bei der Modulo-Division werden zwei Werte dividiert; das Ergebnis ist der Restwert der Division. Beispiele:
`17%5=2` (17 geteilt durch 5 gleich 3 Rest 2)
`10%2=0` (10 geteilt durch 2 gleich 5 Rest 0)
- Vergleiche: gleich (`==`), ungleich (`!=`), kleiner (`<`), größer (`>`), kleiner oder gleich (`<=`), größer oder gleich (`>=`)
Beispiele: `A<B;` `Z=="Hamburg";`
Achtung: Um Werte auf Gleichheit zu prüfen, werden zwei aufeinander folgende Gleichheitszeichen notiert; bei nur einem Gleichheitszeichen erfolgt eine Wertzuweisung!
- Logische Verknüpfungen: UND (`&&`), ODER (`||`), NICHT (`!`)
Beispiel: `Fluessig=(Wassertemp>0)&&(Wassertemp<100);`
- Zeichenkettenverknüpfungen (`+`)
Beispiel: `var Name="Pablo"+" "+"Picasso";`
Die drei Zeichenketten "Pablo", " " (Leerzeichen) und "Picasso" werden zu einer Zeichenkette verknüpft. Die Zeichenkettenverknüpfung liefert als Ergebnis die Zeichenkette "Pablo Picasso".
- Typenbestimmung: Der Typ einer Variablen lässt sich mit dem Operator `typeof` ermitteln.
Beispiel: `alert(typeof PLZHamburg)`

10.4.8 Schleifen

Mit Schleifen werden Reihen von Anweisungen wiederholt ausgeführt. Die Anzahl der Durchläufe unterliegt einer Bedingung. Jede Schleife besteht aus einem Schleifenrumpf mit den darin enthaltenen Anweisungen. JavaScript kennt `for`-Schleifen und `while`-Schleifen sowie Unterarten davon. Die `for`-Schleife lässt sich durch die `while`-Schleife ersetzen und umgekehrt, es besteht nur ein anderes syntaktisches Konzept. Programmierer(innen) können sich also die Syntax aussuchen (*syntactical sugar*). In Schleifen muss es mindestens eine Möglichkeit geben, sie zu beenden. Sonst entstehen Endlosschleifen, die nur durch Beendigung des Tasks enden.

Syntax der `for`-Schleife:

```
for (Startwert der Laufvariablen; Bedingung;
Änderung der Laufvariablen) {
    Anweisung1;
    Anweisung2;
    ...
}
```

Das Ergebnis der Bedingung entscheidet, ob ein Schleifendurchlauf stattfindet (`true`) oder nicht (`false`). Ist das Ergebnis der Bedingung `true`, so werden die Anweisungen innerhalb der geschweiften Klammern ausgeführt.

Für das Zählen der Schleifendurchgänge ist die Laufvariable zuständig. Sie wird am Anfang auf einen Startwert festgesetzt. Nach jedem Schleifendurchlauf wird der Wert der Laufvariablen so geändert, dass die Bedingung entweder weiterhin erfüllt ist (`true`) oder nicht (`false`).

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Schleifen</title>
</head>
<body>
<script type="text/javascript">
for (var i=1; i<=10; i++) {
    document.write("Das Quadrat von " + i + " ist " + i*i + "<br>");
}
</script>
</body>
</html>
```

Der Startwert der Laufvariablen `i` wird auf 1 gesetzt. Mit `i<=10`; wird die Bedingung überprüft und mit `i++` die Laufvariable `i` nach jedem einzelnen Schleifendurchgang um 1 erhöht (`i++` ist die kürzere Notation für `i=i+1`). Die Schleife wird zehnmal durchlaufen, da bei `i=11` die Bedingung `i<=10` nicht mehr erfüllt ist (`false`).

Mit jedem Schleifendurchgang wird durch die Anweisung (Methode) `document.write()`; eine Zeichenkette ausgegeben, in der das Quadrat der Laufvariablen berechnet wird. Anzeige im Browser:

```
Das Quadrat von 1 ist 1
Das Quadrat von 2 ist 4
Das Quadrat von 3 ist 9
Das Quadrat von 4 ist 16
Das Quadrat von 5 ist 25
Das Quadrat von 6 ist 36
Das Quadrat von 7 ist 49
Das Quadrat von 8 ist 64
Das Quadrat von 9 ist 81
Das Quadrat von 10 ist 100
```

Das Beispiel kann alternativ mit der `while`-Schleife realisiert werden. Syntax der `while`-Schleife:

```
Startwert der Laufvariablen;
while (Bedingung) {
    Anweisung1;
    Anweisung2;
    ...
    Änderung der Laufvariablen;
}
```

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Schleifen</title>
</head>
```

```
<body>
<script type="text/javascript">
var i=1;
while (i<=10) {
    document.write("Das Quadrat von " + i + " ist " + i*i + "<br>");
    i++;
}
</script>
</body>
</html>
```

Zuerst wird die globale Variable `i` auf den Wert 1 gesetzt. Am Anfang der `while`-Schleife wird die Bedingung `i<=10` überprüft. Solange das Ergebnis der Bedingung `true` ist, wird mit jedem Schleifendurchlauf die Zeichenkette mit dem Quadrat von `i` im Dokument ausgegeben und mit `i++`; die Variable `i` um 1 erhöht.

10.4.9 Bedingte Ausführungen

JavaScript kennt Wenn-Dann-Bedingungen (`if/else`), Entweder-Oder-Abfragen und Fallunterscheidungen (`switch`). Die Ausführung von Anweisungen ist immer von einer Bedingung abhängig. Ist die Bedingung erfüllt (`true`), so werden die Anweisungen ausgeführt, ist sie nicht erfüllt (`false`), können alternative Anweisungen ausgeführt werden.

Die Wenn-Dann-Bedingung unterscheidet zwischen zwei Fällen. Wenn (`if`) die Bedingung erfüllt ist, werden die Dann-Anweisungen ausgeführt; ist sie nicht erfüllt (`else`, andernfalls), werden die alternativen Anweisungen ausgeführt. Der `else`-Zweig ist optional, es geht also auch ohne alternative Anweisungen. Syntax der `if/else`-Anweisung:

```
if (Bedingung) {
    Dann-Anweisungen
} else {
    Alternative Anweisungen
}
```

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN">
<html>
<head>
<title>if-else-Anweisung</title>
<script type="text/javascript">
function test() {
    var Eingabe=window.prompt("Wieviel ergibt das Quadrat
von 12?", "??");
    if (Eingabe=="144") {
        alert("Richtig, Super!");
    } else {
        alert("Leider falsch!");
    }
}
</script>
</head>
<body onLoad="test();">
<a href="javascript:test();">Wiederholen</a>
</body>
</html>
```

Nach dem Laden der Datei wird mit dem Event-Handler `onLoad` die Funktion `test()` aufgerufen. Mit der Methode `prompt()` des Objekts `window` wird ein Dialogfenster mit einem Eingabefeld angezeigt. Nach erfolgter Eingabe und Betätigen des OK-Buttons wird der Inhalt des Eingabefelds der Variablen `Eingabe` zugewiesen. Die Variable ist vom Typ Zeichenkette (*string*). In der `if`-Bedingung wird der Inhalt der Variablen `Eingabe` mit der richtigen Antwort "144" verglichen (Vergleichsoperator `==`, Prüfung auf Gleichheit). Ist das Ergebnis der Bedingung wahr (`true`), so erscheint das Meldfenster (`alert()`) mit der Antwort "Richtig, Super!". Bei allen anderen Eingaben wird der `else`-Zweig mit der Meldung "Leider falsch!" ausgeführt; im Browserfenster erscheint ein Verweis, um die Funktion erneut aufzurufen.

Alternativ lässt sich das Beispiel auch mit einer Entweder-Oder-Abfrage und einem Formular realisieren. Syntax der Entweder-Oder-Abfrage:

(Bedingung) ? true-Wert : false-Wert;

Die Entweder-Oder-Abfrage beginnt mit der in runden Klammern stehenden Bedingung, gefolgt von einem Fragezeichen. Danach wird der Wert für den Fall angegeben, dass die Bedingung erfüllt ist, gefolgt von einem Doppelpunkt. Dahinter steht der Wert für den alternativen Fall. Da es sich nicht um Anweisungen, sondern um Werte handelt, wird die Abfrage meist einer Variable zugewiesen, die dann den Wert der Abfrage enthält: `Variable = (Bedingung) ? true-Wert : false-Wert;` Mit der Entweder-Oder-Abfrage sieht das Beispiel so aus:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Entweder-Oder-Abfrage</title>
<script type="text/javascript">
function test() {
    var Ergebnis=(document.Formular.Eingabe.value=="144") ?
    "Richtig, Super!" : "Leider falsch!";
    document.Formular.Eingabe.value=Ergebnis;
}
</script>
</head>
<body>
<p>Wieviel ergibt das Quadrat von 12?</p>
<form name="Formular" action="javascript:test();">
<input type="text" name="Eingabe">
<input type="submit" value="OK">
</form>
</body>
</html>
```

Nach Eingabe und Absenden des Formulars wird die Funktion `test()` aufgerufen. In der Bedingung wird der Formularinhalt mit der Zeichenkette "144" verglichen. Ist die Bedingung wahr, so wird die Zeichenkette "Richtig, Super!" der Variablen `Ergebnis` zugewiesen, andernfalls die Zeichenkette "Leider falsch!". Danach erfolgt die Ausgabe der Variablen `Ergebnis` im Eingabefeld des Formulars.

Die Fallunterscheidung mit `switch` bietet sich an, um mehr als zwei Fälle zu unterscheiden. Dies lässt sich zwar auch mit ineinander verschachtelten `if/else`-Bedingungen erreichen, wird aber sehr schnell unübersichtlich.

Syntax der Fallunterscheidung:

```
switch (Variable oder Ausdruck) {
    case Wert1:
        Anweisung1; Anweisung2; ...; break;
    case Wert2:
        Anweisung1; Anweisung2; ...; break;
    ...
    ...
    default:
        Anweisung1; Anweisung2; ...; break;
}
```

Die Fallunterscheidung beginnt mit dem Befehl `switch` (Schalter), gefolgt von einer in runde Klammern eingeschlossenen Variablen oder einem Ausdruck, von dessen Wert die Auswertung abhängig ist. Innerhalb der geschweiften Klammern findet die eigentliche Fallunterscheidung statt. Mit `case` (Fall) und der Angabe eines zu prüfenden Wertes wird ermittelt, ob die Werte gleich sind. Bei Gleichheit werden die nachfolgenden Anweisungen ausgeführt. Bei Ungleichheit wird der nächste Fall überprüft. Mit `break` (abbrechen) am Ende des Falls wird die Fallunterscheidung beendet; ohne die Anweisung `break` werden auch alle weiteren Fälle ausgeführt. Mit `default:` lassen sich Anweisungen ausführen, für die keine Übereinstimmung in einer `case`-Anweisung gefunden wurde, wenn also keiner der Fälle zutrifft.

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Fallunterscheidung mit switch</title>
<script type="text/javascript">
function test() {
    switch (document.Formular.Eingabe.selectedIndex) {
        case 0:
            Ergebnis="Zwei Reifen";
            break;
        case 1:
            Ergebnis="Drei Reifen";
            break;
        case 2:
            Ergebnis="Vier Reifen";
            break;
        case 3:
            Ergebnis="Mehr als vier Reifen";
            break;
        case 5:
            Ergebnis="Keine Reifen";
            break;
        default:
            Ergebnis="Bitte waehlen Sie!";
            break;
    }
    document.Formular.Ausgabe.value=Ergebnis;
}
</script>
</head>
<body>
<form name="Formular">
<select name="Eingabe" size="7" onChange="test();">
<option>Fahrrad</option>
```

```

<option>Dreirad</option>
<option>PKW</option>
<option>LKW</option>
<option selected>-----</option>
<option>Schiff</option>
<option>-----</option>
</select>
<p><input type="text" name="Ausgabe"></p>
</form>
</body>
</html>

```

Mit der Wahl eines Auswahlfelds (`onChange`) wird die Funktion `test` aufgerufen. In Abhängigkeit von der Indexnummer der Auswahl wird bei der eigentlichen Fallunterscheidung der entsprechende Text der Variablen `Ergebnis` und danach dem Formularfeld `Ausgabe` zugewiesen und darin angezeigt. Achtung: Der Index beginnt mit der Zahl 0.

Hinweise und Korrekturen

Zu 3.7 Color-Management

Dreidimensionale Darstellungen der Farbumfänge von RGB- und CMYK-Farbräumen finden sich unter www.iccview.de.

Zu 4.4.5 Gruppe V: Serifenbetonte Linear-Antiqua

Die Gliederung der Gruppe V in drei Untergruppen (Egyptienne, Clarendon und Italiane) ist allgemein üblich und anerkannt, aber nicht Bestandteil der Norm DIN 16518.

Zur Schreibweise des Namens Aloys (Alois) Senefelder (Abschnitte 4.4.5, 5.4.1, 7.1 und 7.3.2)

Die historisch korrekte, ursprüngliche Schreibweise des Namens lautet *Aloys Senefelder*. In Deutschland hat sich seit etwa 1950 die Schreibweise *Alois* durchgesetzt (vgl. zum Beispiel: Berufliches Schulzentrum Alois Senefelder, München), während es in zahlreichen anderen Ländern bei der ursprünglichen Schreibung *Aloys Senefelder* geblieben ist.

Zu 6.8.7 Laser

Die Wellenlänge des CO₂-Lasers liegt im Bereich von rund 9500 nm bis 10 600 nm (im Buch fehlt jeweils eine Null). Auch diese Wellenlängen gehören zum infraroten Bereich, der von rund 780 nm bis 10 000 000 nm (1 cm) reicht.

Zu 6.9.1.1 Kopie

Auch UVA ist gesundheitlich nicht völlig unbedenklich. Es erzeugt zwar keine Verbrennungen (Sonnenbrand, Erythem), beschleunigt aber die Hautalterung in Abhängigkeit von Dauer und Intensität der Bestrahlung. Vor allem bei sehr hohen Bestrahlungsstärken können die Augen durch UVA akut und langfristig geschädigt werden (Entzündung von Horn- und Bindehaut bzw. grauer Star).