



Mit Java-Script

Cookies und Co.

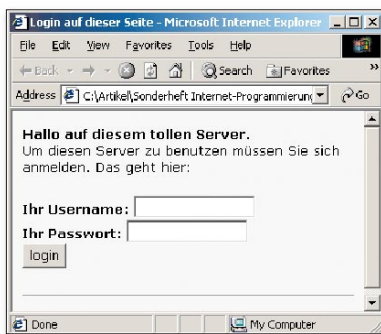
Fast **alle großen Websites setzen Cookies ein**, und in fast allen Browsern sind sie aktiviert. Wie Sie **selber Cookies nutzen** können, zeigt Ihnen der folgende Beitrag.

THOMAS WÖLFER

Dieser Beitrag ist in zwei Teile gegliedert: Im ersten Teil erhalten Sie das notwendige Hintergrundwissen über Cookies und wenden dies in einem einfachen Beispiel an. Im zweiten Teil implementieren Sie eine wiederverwendbare Cookie-Klasse, die Sie immer wieder in eigenen Projekten benutzen können. Den kompletten Quell-Code zu allen Beispielen und zur Cookie-Klasse finden Sie auf der Heft-CD dieses Sonderheftes.

■ Cookies – der Einstieg

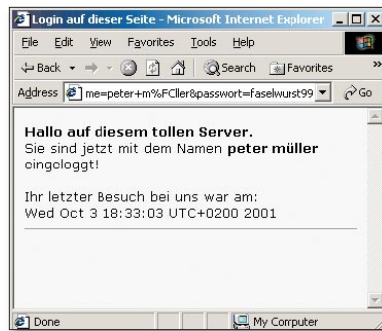
Bei Cookies handelt es sich um einen kleinen Datenblock, der mit einer bestimmten Webseite assoziiert ist. Die Daten als solche liegen in Form von Strings vor und können innerhalb des Web-Browser als Speicherplatz für Zustände (state) verwendet werden. Die Cookies selbst werden auf dem Client



DAS SPEICHERN VON ANMELDEDATEN mit Cookies ist nicht sehr sicher, allerdings ist es sehr praktisch. Der Surfer muss sich seine Account-Daten nicht länger merken.

gespeichert. Dadurch ist es zum Beispiel möglich, Angaben eines Besuchers auf dessen eigenem Rechner zu speichern und bei einem späteren Besuch des Sur-

fers – oder bei einem Wechsel auf eine andere Seite innerhalb der gleichen Domain – wiederzuverwenden. In Cookies können Sie Angaben eines Users spei-



NACH DEM ANMELDEN begrüßen Sie den Surfer nicht nur mit seinem Namen, sondern auch mit dem Datum seines letzten Besuchs.

chern, um diese Angaben in Ihren Scripts wiederzuverwenden.

An anderer Stelle in diesem Heft haben Sie erfahren, wie Sie eine User-Verwaltung für Ihre Webseite in PHP implementieren: Wenn Sie diese einsetzen, müssen sich Ihre User immer bei Ihnen anmelden. Um diesen Anmeldevorgang zu vereinfachen, können Sie zum Beispiel Cookies verwenden, indem Sie die Anmelde-Angaben auf dem Rechner des Surfers speichern und beim nächsten Besuch einfach eine automatische Anmeldung vornehmen. (Dabei ist zu beachten, dass dies keine sehr sichere Methode der Account-Verwaltung ist. Für einfache Websites, im Besonderen, wenn dabei keine finanziellen Transaktionen durchgeführt werden, ist dies aber sicherlich kaum ein Problem.)

■ Die Eigenschaften

Cookies haben verschiedenen Eigenschaften, und zwar Folgende:

- Haltbarkeit

- Sichtbarkeit
- Sicherheit

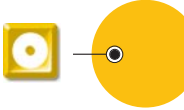
Die Haltbarkeit eines Cookies gibt an, wie lange ein gegebenes Cookie gültig sein soll. Im Normalfall ist ein Cookie nur während einer Browser-Session gültig. Wird der Browser beendet, verschwindet auch das Cookie. Es ist aber möglich, explizit eine längere Haltbarkeit anzugeben, in diesem Fall wird das Cookie als Datei auf dem Client-Rechner gespeichert und steht auch später noch zur Verfügung. Läuft das Haltbarkeitsdatum ab, steht das Cookie nicht länger zur Verfügung.

Die Sichtbarkeit eines Cookies wird durch zwei zusammengesetzte Eigenschaften bestimmt: durch seinen Pfad und seine Domain. Im Normalfall ist ein Cookie nur auf der Seite sichtbar, die es angelegt hat sowie auf allen anderen Seiten, die im Web-Server im gleichen Verzeichnis liegen. Mit dem Pfad eines Co-



BEIM NÄCHSTEN BESUCH des Surfers wird sein Passwort automatisch eingetragen, wenn er sich mit dem gleichen Benutzernamen anmeldet.

kies kann man diese aber verändern. Meist geschieht dies dadurch, dass der Pfad auf „/“ gesetzt wird, sodass jede Seite eines Servers, ganz gleich wo sie abgelegt ist, das Cookie benutzen kann.



Ebenso ist ein Cookie normalerweise nur auf dem Server sichtbar, von dem die Seite stammt, die das Cookie angelegt hat. Bei sehr großen Sites ist es aber

```
// cookie lesen
function GetCookie( name)
{
    var dc = document.cookie;
    var prefix = name + "=";
    var begin = dc.indexOf("; " + prefix);
    if( begin == -1)
    {
        begin = dc.indexOf(prefix);
        if( begin != 0) return null;
    }
    else
    {
        begin += 2;
    }
    var end = document.cookie.indexOf("; ", begin);
    if( end == -1) end = dc.length;
```

DEN JAVASCRIPT-CODE lagern Sie am besten in einer externen JavaScript Datei aus. So kann dieser Code einfacher gecached werden.

manchmal wünschenswert diese Information auch mehreren Servern zur Verfügung zu stellen – zum Beispiel www.nickles.de und auch search.nickles.de. Dafür ist das Domain-Property zu-

```
<form name=login_form action=welcome.htm>
  <b>Ihr Username:</b> <input type=text size=16 name=name onBlur="OnBlurUser();" />
  <b>Ihr Passwort:</b> <input type=text size=16 name=password onBlur="OnBlurUser();" />
  <input type=submit value=login />
</form>
```

DEN HTML-TEIL für die Beispiele bearbeiten Sie am besten mit einem HTML-fähigen Editor.

Der Sicherheits-Aspekt von Cookies bestimmt, wie diese im Netz transportiert werden. Ist das Sicherheits-Flag eines Cookies gesetzt, wird der Cookie-Inhalt nur dann transportiert, wenn der Client mit dem Server über eine HTTPS-Verbindung kommuniziert. Ist das Flag nicht gesetzt, wird das Cookie auch per HTTP transportiert.

■ Cookies und Limits ...

Cookies waren von Anfang an dafür gedacht nur kleine Datenmengen zu speichern, und so sehen die Specs für Cookies auch aus. Ein Web-Browser muss nicht mehr als 300 Cookies speichern können (obwohl alle aktuellen das tun), und mehr als 20 Cookies pro Web-Server sind auch nicht verlangt. Obendrein darf ein Cookie nicht mehr als 4 KByte

Daten enthalten. Besonders wegen der 20-Cookies-pro-Server-Einschränkung ist es sinnvoll, nicht für jede Einstellung ein eigenes Cookie zu verwenden, sondern alle Einstellungen in einem einzelnen Cookie zu verwalten. (Im Beispiel Cookie-Klasse wird genau dies getan.)

■ Cookies: Lesen und schreiben

Sie speichern ein Cookie, indem Sie (in JavaScript) einen Wert in der *cookie*-Eigenschaft des Dokuments ablegen. Weil einige Zeichen innerhalb von Cookies nicht

zulässig sind, ist es dabei sinnvoll, die zu speichernden Werte zu escapen:

```
document.cookie = "NameDes" +
  "Cookies" + escape( VariableDie" +
  "DenWertEnthaelt);
```

Um die Eigenschaften wie den Pfad oder die Haltbarkeit anzugeben, erweitern Sie das Cookie um zusätzliche „Name = Wert“-Paare.

Um ein Cookie zu lesen, verwenden Sie einfach den String, den Sie erhalten wenn Sie `document.cookie`

auswerten. Den gelieferten String müssen Sie in „Name=Wert“-Paare aufteilen und schon haben Sie den gewünschten Teil.

■ Ein einfaches Beispiel

Als erstes Beispiel soll die User-Verwaltung aus einem anderen Beitrag dieses Sonderheftes dienen. Dort muss sich ein Anwender mit seinem Namen und seinem Passwort anmelden. Die Richtigkeit dieser Angaben wird auf dem Server überprüft. Um dem Anwender das Anmelden in Zukunft zu vereinfachen, könnten Sie Folgendes tun: Nachdem der Surfer seinen Namen eingegeben hat und das entspre-

chende Feld verlässt, suchen Sie ein Cookie, das als Inhalt den eingegebenen Benutzernamen verwendet. Liegt dieses Cookie vor, suchen Sie nach einem Cookie, das das Passwort enthält und wird dieses gefunden, füllen Sie das Passwort-Feld automatisch auf. Wird das Passwort-Feld verlassen, speichern Sie einfach sowohl den User-Namen als auch das Passwort und der User muss nie wieder sein Passwort eingeben. (Es wäre natürlich sinnvoller diese beiden Werte erst dann zu speichern, wenn der Server die Authentifizierung bereits durchgeführt hat. Im Beispiel wurde das aus Sicherheitsgründen nicht getan.)

Sie benötigen vier Funktionen: eine zum Lesen eines Cookies, eine zum Speichern eines Cookies sowie die beiden Hilfsfunktionen, die beim Verlassen der Formularfelder aufgerufen werden. Das Formular implementieren sie wie folgt:

```
<form name=login_form
  >action=welcome.htm>
  Ihr Username: <input type=text
  >name=name onBlur="OnBlur
  >User();" />
  Ihr Passwort: <input type=text
  >name=password onBlur="OnBlur
  >Password();" />
  <input type=submit value=login>
  </form>
```

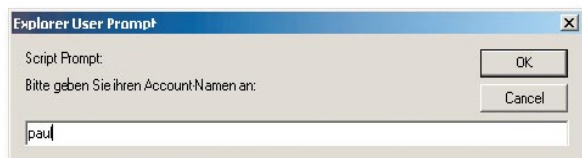
Im Head der HTML-Datei inkludieren Sie außerdem eine JavaScript-Datei, die die benötigten Funktionen enthält:

```
<html><head><script language=
  >JavaScript
  >src=login.js</script></head>
```

`OnBlurUser()` wird dann aufgerufen, wenn das Feld *Benutzernamen* verlassen wird:

```
function OnBlurUser()
{
    lastName = GetCookie
    >("username");
    if( lastName == document.all
    >("login_form").name.value)
    {
        password = GetCookie
        >("password");
        document.all("login_form")
        >.password.value = password;
    }
}
```

Die Funktion liest dann den Wert eines Cookies aus, das den Namen *username* trägt. Entspricht dieser Wert dem User-Namen, der im Feld *name* eingegeben



IM BEISPIEL für die JavaScript-Klasse muss sich der Surfer beim ersten Besuch der Seite mit einem Namen anmelden.

wurde, liest die Funktion den Wert des Cookies mit dem Namen *passwort* aus und trägt diesen im Passwort-Feld ein.

OnBlurPassword() wird aufgerufen, wenn das Passwort-Feld verlassen wird. Diese Funktion speichert einfach den Wert des Usernamen-Feldes und den des Passwort-Feldes in Cookies.

Außerdem setzt diese Funktion die Le-



BEI FOLGENDEN BESUCHEN wird der Benutzer automatisch angemeldet. Zudem wird ihm mitgeteilt, wie oft er sich bereits auf dem Server befunden hat.

benszeit des Cookies auf ein Jahr. Jedesmal wenn der Surfer sich neu anmeldet, verlängert sich die Haltbarkeit des Cookies um 365 Tage. Das läuft wie folgt ab:

```
function OnBlurPassword()
{
    var now = new Date();
    var nextYear = new Date();
    nextYear.setTime( next
    ↳Year.getTime() + 365 * 24
    ↳* 60 * 60 * 1000);
```

Zunächst ermitteln Sie das aktuelle Datum und errechnen daraus das Datum in 365 Tagen. Mit diesem Datumswert speichern Sie sowohl den User-Namen als auch das Passwort und geben diesen beiden Cookies die entsprechende Haltbarkeit:

```
SetCookie( "username", document.
↳all("login_form").name.
↳value, nextYear);

SetCookie( "passwort", document.
↳all("login_form").passwort.
↳value, nextYear);
```

Schließlich speichern Sie noch das aktuelle Datum, das Sie beim Begrüßungstext verwenden können:

```
SetCookie( "lastDate", now, next
↳Year);
```

Nun stellt sich aber die Frage, wie die *SetCookie()*- und *GetCookie()*-Funktionen funktionieren. Das geht folgendermaßen:

SetCookie() ist verhältnismäßig einfach. Die Funktion erhält fünf Parameter, von denen die meisten optional sind:

```
function SetCookie( name, value,
↳expires, path, domain, secure)
```

name bestimmt den Namen des Cookies und *value* gibt den zu diesem Namen gehörenden Wert an. *expires* gibt das Ablaufdatum des Cookies, *path* *domain* und *secure* bestimmen dessen weitere oben beschriebenen Eigenschaften.

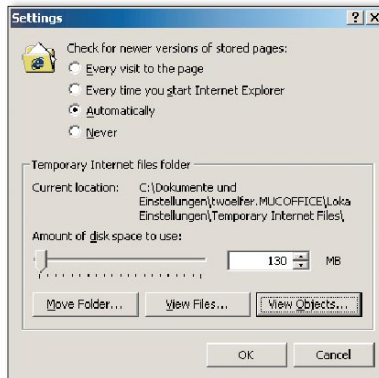
Mit diesen Angaben setzen Sie den vollständigen Cookie-String zusammen:

```
var curCookie = name + "=" +
↳escape(value) +
↳((expires) ? "; expires="
↳+ expires.toGMTString() :
↳"" ) +
↳((path) ? "; path=" + path
↳: "" ) +
↳((domain) ? "; domain=" +
↳domain : "" ) +
↳((secure) ? "; secure" :
↳"" );
```

Schließlich speichern Sie das Cookie einfach durch eine Zuweisung:

```
document.cookie = curCookie;
```

GetCookie() ist komplizierter, weil diese Funktion aus dem kompletten String den Wert herausuchen muss, der zum als



BEIM INTERNET-EXPLORER landen die Cookies in den "Temporären Internet-Dateien".

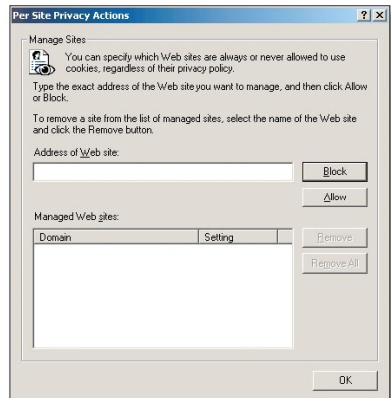
Parameter angegebenen Namen passt:

```
function GetCookie( name)
{
    var dc = document.cookie;
    var prefix = name + "=";
    var begin = dc.indexOf("; " +
↳prefix);
    if( begin == -1)
    {
        begin = dc.indexOf(prefix);
        if( begin != 0) return
↳null;
    }
    else
    {
        begin += 2;
    }
    var end = document.cookie.
↳indexOf("; ", begin);
    if( end == -1) end =
↳dc.length;
    return unescape( dc.sub
↳string( begin +
↳prefix.length, end));
```

Ein kurzes Anwendungs-Beispiel für diese Funktionen finden Sie auf der Heft-CD.

■ Die eigene Cookie-Klasse

Nachdem Sie in JavaScript eigene Objekte (Klassen) definieren können, ist eines klar: Bei Cookies handelt es sich um



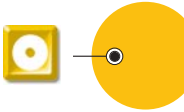
BEI AKTUELLEN BROWSERN können Sie für jede Webseite getrennt einstellen, ob Cookies angenommen werden sollen oder nicht.

einen ganz eindeutigen Kandidaten für die Implementierung einer eigenen Klasse, die Sie dann später immer wieder verwenden können. Genau diese Klasse implementieren Sie im restlichen Teil dieses Beitrags.

Die Cookie-Klasse wird über einen Konstruktor sowie zwei Methoden verfügen. Bei diesen Methoden handelt es sich um eine *Save()*- und eine *Load()*-Methode. Eine dritte interessante Methode wäre die *Delete()*-Methode, mit der Sie ein gegebenes Cookie vom Rechner löschen können. Bei der Implementierung dieser Methode können Sie das Gelernte zum ersten Mal einsetzen.

Außerdem wird die Cookie-Klasse deutlich sinnvoller funktionieren als die beiden bisher vorgestellten Funktionen, denn die Cookie-Klasse wird tatsächlich, wie weiter oben erwähnt, beliebig viele „Name=Wert“-Paare in einem einzelnen Cookie auf dem Rechner verwalten können.

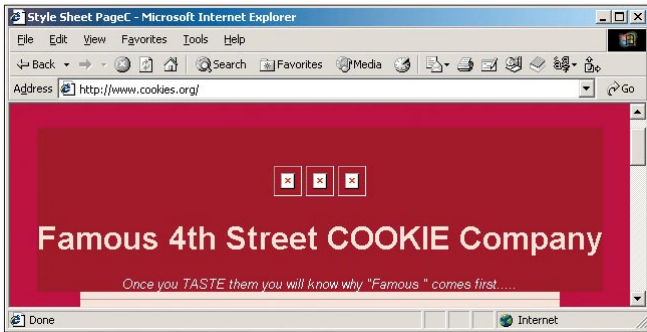
Bei JavaScript existieren keine echten Member-Variablen wie in C++. Alle Eigenschaften sind einfach nur eine Eigenschaft. Um trotzdem „normale“ Eigenschaften von den „internen“ Eigenschaften des Cookies unterscheiden zu können, verwenden Sie bei der Cookie-Klasse eine Notation wie Sie dies vielleicht auch schon bei C++ Klas-



sen gewohnt sind: „Interne“ Member haben einen Namen, der mit dem `m_` Prefix beginnt. Aber Achtung: Es handelt sich hier wirklich nur um eine Notation. Sie müssen also aufpassen, dass Sie später keine Eigenschaften für die Cookie-Klasse definieren, die mit einem solchen Namen beginnen!

Kommen wir zunächst zum Konstruktor. Dieser speichert alle ge-

Mit der ersten Zeile erzeugen Sie ein neues Cookie-Objekt. Dabei werden die „internen“ Eigenschaften des Objektes (also die mit den `m_*` Bezeichnungen) gesetzt. Das ist zum Beispiel der Name des Cookies für den Web-Browser: Dieser wird das Cookie später unter der Bezeichnung *MeinName* ablegen und das Cookie für 48 Stunden gültig halten.



DIE COOKIE-COMPANY www.cookie.org backt Kekse – Informationen zum Programmieren finden Sie dort nicht.

wünschten Eigenschaften eines Cookies: Dies sind *nicht* die „NAME=Wert“-Paare – die kommen später –, sondern nur die Eigenschaften die den Web-Browser selbst interessieren. Die „optionalen“ Angaben sind dabei genau das: optionale Parameter:

```
function XCookie( documentName,
    ↪ cookieName, hoursLifeTime,
    ↪ path, domain, fSecure )
{
    this.m_document = document
    ↪ Name;
    this.m_name = cookieName;
    if( hoursLifeTime ) this.m_
    ↪ expirationTime = new Date((
    ↪ new Date()).getTime() +
    ↪ hoursLifeTime*3600000);
    else this.m_expirationTime =
    ↪ null;
    if( path ) this.m_path = path;
    else this.m_path = null;
    if( domain ) this.m_domain =
    ↪ domain;
    else this.m_domain = null;
    if( fSecure ) this.m_fSecure =
    ↪ true;
    else this.m_fSecure = false;
}
```

Die Methode zum Speichern eines Cookies baut aus allen Eigenschaften des Cookies den Cookie-String zusammen. Um den Unterschied zwischen den internen Eigenschaften und den „NAME=Wert“-Eigenschaften zu verdeutlichen, geben wir hier zunächst ein einfaches Anwendungsbeispiel. Gewünscht ist, dass Sie später folgenden Ausdruck verwenden können:

```
var myCookie = new XCookie
    ↪ (document, "MeinName", 48);
myCookie.login_name = "Peter
    ↪ Mueller"; myCookie.save();
```

Mit der zweiten Zeile definieren Sie ein „Name =Wert“- Paar: Dem Namen „login_name“ wird der Wert „Peter Mueller“ zugeordnet. Damit können Sie später die „externe“ Cookie-Eigenschaft `login_name` verwenden, um dessen Wert auszulesen.

wenden, um dessen Wert auszulesen.

Die *Save()* Methode muss also einen Cookie-String zusammensetzen, der sich sowohl aus den „internen“ wie auch aus den „externen“ Eigenschaften zusammensetzt. Leider kann man wie gesagt dazwischen nicht unterscheiden, daher überprüft die Methode einfach den Namen der Eigenschaft. Beginnt dieser mit einem „m“ (besser wäre: `m_`), wird die Eigenschaft als intern behandelt:

```
function XCookie_save()
{
    var value = "";
    for( var prop in this )
    {
        if( ( prop.charAt(0) ==
            ↪ 'm' ) ||
            (( typeof( this[prop] ) ==
            ↪ 'function' ) ) )
        {
            continue;
        }
        if( value != "" ) value +=
            ↪ '&';
        value += prop + ':' +
            ↪ escape(this[prop]);
    }
}
```

Zunächst iterieren Sie also einfach über alle Eigenschaften des Cookies. Handelt es sich bei der Eigenschaft um eine, deren Namen mit einem „m“ beginnt, oder ist die Eigenschaft gar eine Funktion, ignorieren Sie diese. Im anderen Fall hängen Sie den Namen der Eigenschaft und dessen Wert an eine String-Variable an.

Danach wird es einfach. Sie legen eine neue Variable an und füllen diese mit dem passenden String auf. Dazu legen Sie zunächst fest, dass der dem „internen“ Cookie-Namen zugeordnete Wert der zuvor zusammengesetzte String ist. Danach fügen Sie anhand der internen Eigenschaften die anderen Cookie-Werte ein:

```
var cookie = this.m_name + '=' +
    ↪ value;
if( this.m_expirationTime )
    ↪ cookie += '; expires=' +
    ↪ this.m_expirationTime.toGMT
    ↪ String();
if( this.m_path ) cookie += ';
    ↪ path=' + this.m_path;
if( this.m_domain ) cookie +=
    ↪ '; domain=' + this.m_domain;
```

COOKIES UND DER SERVER – HINTERGRUNDWISSEN

Cookies sind nicht nur für den Client-seitigen Betrieb von Webseiten interessant. Die Informationen von Cookies sind auch auf dem Server auslesbar, und ebenso ist der Server in der Lage, Cookies auf dem Client zu setzen. Damit können Sie auf Ihrem Server auf Cookies reagieren, die auf dem momentan verbundenen Client vorliegen.

Im Wesentlichen funktioniert das so: Wenn ein Client eine Seite vom Server anfordert, und auf dem Client existieren Cookies, die dieser Seite zugeordnet sind, werden alle diese Cookies zusammen mit der Seitenanforderung an den Server gesendet. Der Server seinerseits kann beim Liefern einer Seite Cookies im Seitenheader mit auf den Client schicken: Empfängt der Browser diese Cookies, speichert er sie der Seite entsprechend zugeordnet ab.

Es kann dabei nur der Server, der für eine bestimmte Seite zuständig ist, auch die Cookies dieser Seite auslesen. Dabei müssen Sie sich allerdings bewusst machen, dass eine Seite aus mehreren "Seiten-Elementen" zusammengesetzt sein kann. Eine Seite kann zum Beispiel eine Werbung enthalten, die von einem ganz anderen Server als die Seite selbst stammt. In diesem Fall hat auch der "Werbe"-Server Zugriff auf Cookies, aber eben nur auf diejenigen, die der Werbung zugeordnet sind, nicht aber auf die, die der Seite selbst zugeordnet sind.

Cookie-Support ist sowohl in PHP als auch in ASP eingebaut. In beiden Fällen finden Sie die passenden technischen Dokumentationen am einfachsten in der Online-Hilfe, indem Sie nach dem Stichwort „Cookie“ suchen.



```

if( this.m_fSecure) cookie +=
  ↳'; secure';
this.m_document.cookie =
  ↳cookie;
}

```

Die *Load()*-Methode lädt nun das Cookie und dekodiert alle „externen“ Eigenschaften des Cookies aus dem String. In diesem Zuge wird für jedes „Name=Wert“-Paar eine neue Cookie-Eigenschaft mit dem richtigen Wert erzeugt:

```

function XCookie_load()
{
    var all = this.m_
  ↳document.cookie;
    if( all == "" ) return
  ↳false;
    var begin = all.indexOf
  ↳( this.m_name + '=' );
    if( begin == -1 ) return
  ↳false;
    begin +=
  ↳this.m_name.length + 1;
    var end = all.indexOf( ',',
  ↳begin );
    if( end == -1 ) end =
  ↳all.length;
    var value = all.
  ↳substring(begin,end);
    var arr =
  ↳value.split('&');
    for( var index=0;
  ↳index<arr.length;
  ↳index++)
    {
        arr[ index] = arr
  ↳[ index].split(':');
    }
    for( var index=0;
  ↳index<arr.length;
  ↳index++)
    {
        this[ arr[index][0]] =
  ↳unescape( arr
  ↳[index][1]);
    }
    return true;
}

```

Schließlich müssen Sie noch einen Prototypen für die Cookie-Klasse definieren, damit Sie die Klasse später verwenden können. Dabei geht es im Wesentlichen darum, festzulegen, unter

welchem Namen die *Load()*- und die *Save()*-Funktion später aufgerufen werden können.

```

new XCookie;
XCookie.prototype.
  ↳save = XCookie_
  ↳save;
XCookie.prototype.
  ↳load =
  ↳XCookie_load;

```

Abschließend finden Sie ein komplettes Beispiel für die Cookie-Klasse. In der HTML-Seite müssen Sie zunächst die JavaScript-Datei, die die Klassen-Definition enthält inkludieren. (Übrigens: An anderer Stelle in diesem Sonderheft zeigen wir Ihnen, wie Sie sicherstellen, dass Ihre Besucher diese Datei nicht immer wieder erneut herunterladen. Sie sollten auf jeden Fall sicherstellen, dass derartige Dateien richtig gecached werden.):

```

<script language="JavaScript1.1"
  ↳src="cookie_class.js"></script>

```

Dann definieren Sie eine einfache Funktion, die beim *OnLoad()* der Seite aufgerufen wird:

```

function OnLoad()
{
    var data = new XCookie( docu
  ↳ment, "ein_netter_name", 48);
}

```

Zunächst erzeugen Sie ein neues Cookie-Objekt. Damit die Cookie-Klasse das zugehörige Dokument kennt, müssen Sie dieses übergeben. Ferner geben Sie einen Namen und eine Lebenszeit an.

Die optionalen Parameter lassen Sie weg.

Mit diesem Cookie werden Sie nun den Benutzernamen eines Benutzers speichern, und ebenso die bisherige Anzahl der Besuche dieses Surfers bei Ihnen. Dazu versuchen Sie zunächst das Cookie zu laden, und falls dies ging, überprüfen Sie außerdem, ob die Eigenschaft *name* bereits gesetzt ist:

```

fPrompt = false;
if( ! data.load())
  ↳fPrompt =

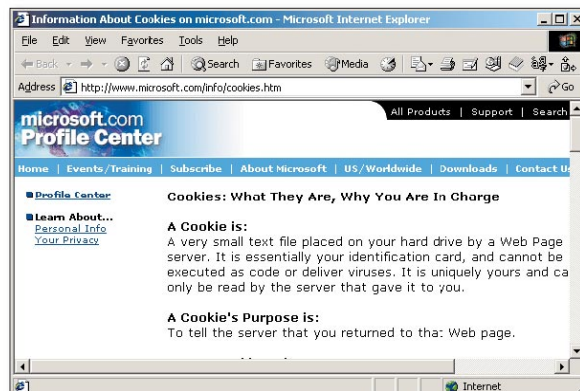
```

```

  ↳true;
else
{
    if( data.name == "null")
  ↳fPrompt = true;
    if( ! data.name) fPrompt =
  ↳true;
}

```

Ging eines davon nicht, fordern Sie den Surfer auf, seinen Namen einzugeben.



VIELE WEBSEITEN-BETREIBER geben sehr genau an, welche Informationen sie in Cookies ablegen und welche nicht. Man braucht nur noch genug Vertrauen, diesen Angaben zu glauben.

Den eingegebenen Namen speichern Sie in der Eigenschaft *name*.

```

if( fPrompt)
{
    data.name = prompt("Bitte geben
  ↳Sie ihren -Namen an:",
  ↳" Namen eingeben");
}

```

Im nächsten Schritt merken Sie sich noch die Anzahl der bisherigen Besuche. Dazu verwenden Sie einfach eine weitere Eigenschaft, die Sie zum Beispiel „hits“ nennen können. Ist diese noch nicht vorhanden, so erzeugen Sie sie und setzen sie auf den Wert „0“. Dann inkrementieren Sie diesen Wert:


```

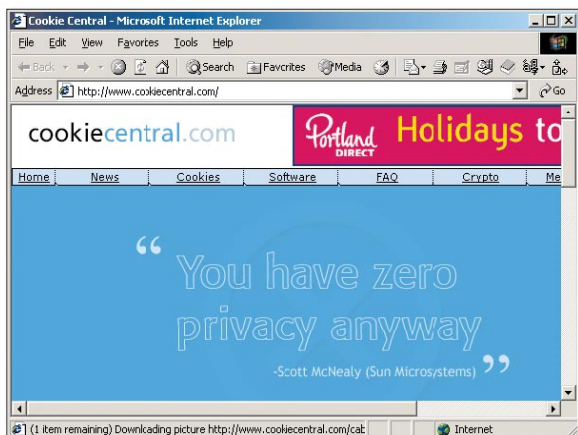
if( data.hits == null) data.hits =
  ↳0;
data.hits++;

```

Schließlich speichern Sie das Cookie. Beim nächsten Besuch des Surfers muss sich dieser nicht mehr namentlich melden – der Hit-Counter wird aber dennoch hochgezählt.

```
data.save();
```

In diesem Beitrag haben Sie erfahren, wie Sie Cookies in einfachen Fällen verwenden um Angaben des Benutzers zu speichern. Außerdem haben Sie eine allgemeingültige Cookie-Klasse für die Verwendung mit JavaScript programmiert, mit der Sie in Zukunft praktisch alle Cookie-Belange abhandeln können. Viel Spaß beim Kekse essen!  UR



BEI COOKIECENTRAL.COM finden Sie eine Menge an Informationen über Cookies – so auch eine ausführliche FAQ.