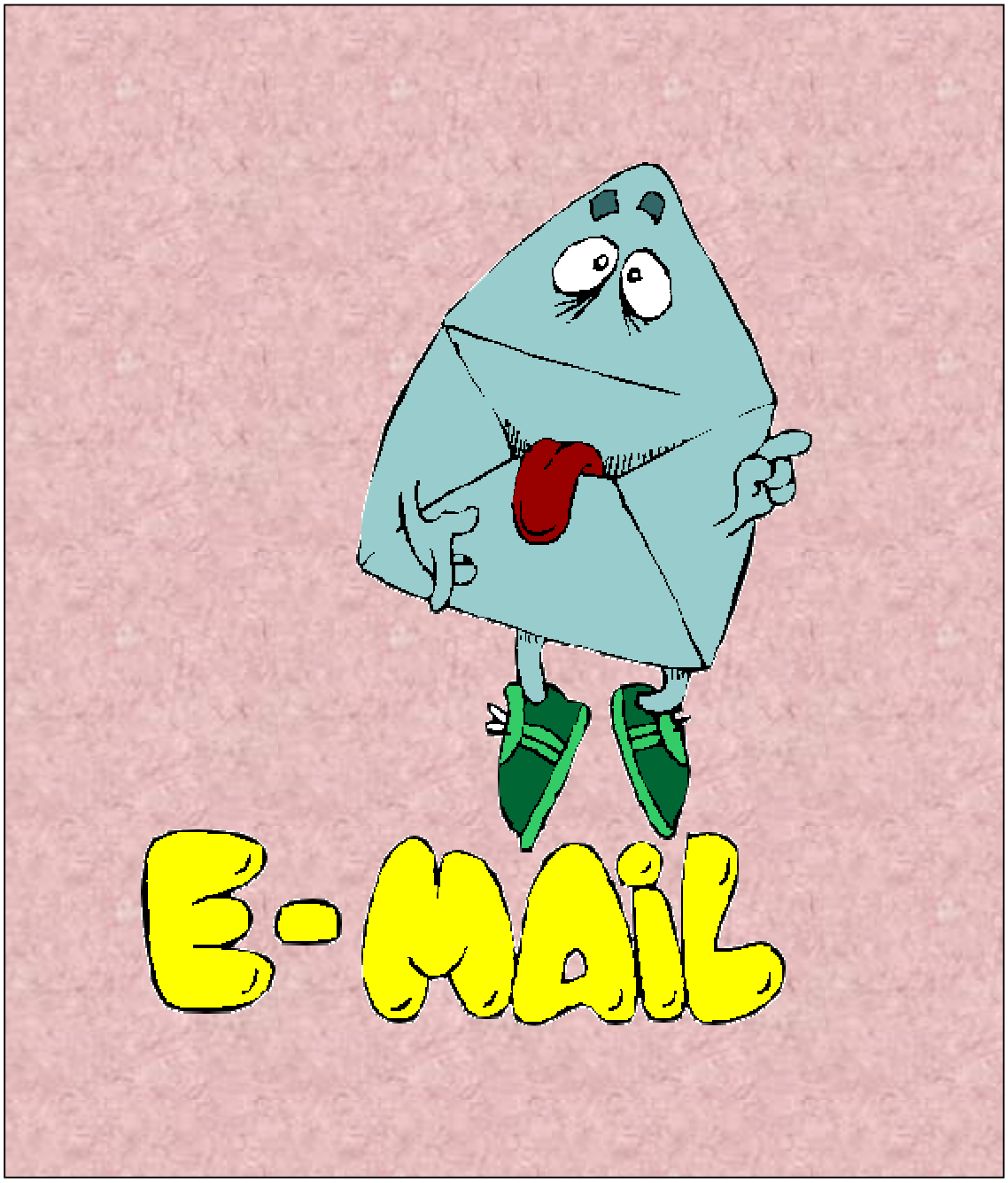
Stationery – Scripting
Teil 2



Inhaltsverzeichnis

Lektion 1: Gradient-Filter

Entpacken Sie die heruntergeladene Zip-Datei, wählen Sie den Ordner Lektion1und öffnen Sie die Datei Lektion1.html mit einem Editor Ihrer Wahl, es genügt schon ein reiner Texteditor.

Sie sehen folgenden Quellcode:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
</head>
<body>
<p id=start>S T A R T</p>
</body>
</html>
```

Wir fügen nun als erstes einen 3 Zeilen lagen VB-Code ein, die generell vorhanden sein sollten, wenn Sie Stationeries mit VB erstellen. Das verhindert bei manchen Browsern eine Fehlermeldung wenn ein OnClick-Ereignis eintritt. Fügen Sie in den Code die roten Zeilen ein:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
</head>
<body>

</body>
</html>
```

Erklärung:

Sollte ein Fehler auftreten, wird die den Fehler verursachende Zeile übersprungen.

Wir wollen nun unseren Quelltext erweitern, indem wir einen Startbildschirm erzeugen, dabei Verwende ich für den Body einen Verlausfilter und eine Formatierung für ** ** in einer Styleanweisung.

Fügen Sie den folgenden, roten und blauen Quellcode ein:
Wichtig: der blau markierte Bereich muss als eine!! Zeile eingegeben werden.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body     { filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff000000,
endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
</style>
</head>
<body>
<p id=start >S t a r t</p>
</body>
</html>
```

Der Paragraph mit dem Text 'START' habe ich nach rechts unten gesetzt, wir werden diesen Text später als Auslöser für ein Script-Routine verwenden.
Der Filter im Body ist ein Gradientfilter, der es erlaubt eine Startfarbe und eine Endfarbe zu definieren, die dann fließend von der Start- zur Endfarbe übergeht.

Syntax Gradientfilter:
progid:DXImageTransform.Microsoft.Gradient(startcolorstr=xx, endcolorstrxx ,gradienttype=x,enabled=x)

Gradiendtype ist die Art des Übergangs. **1=von links nach rechts, 0 ist von oben nach unten.**
Startcolorstr=#00000000 = die Startfarbe, mit der ein Objekt gefüllt werden soll.
Endcolorstr=#00000000 = die Endfarbe, mit der ein Objekt gefüllt werden soll.

Die Farbparameter setzten sich aus 8 Byte zusammen und werden in der Regel hexadezimal eingegeben.**00000000**
Die ersten beiden Werte geben den Alphaanteil an, oder anders ausgedrückt, den Transparenzanteil. 00=voll transparent, FF=keine Transparenz.
Die zweiten zwei Bytes sind der Rotanteil.
Die dritten zwei Bytes sind der Grünanteil
Und die vierten zwei Bytes sind der Blauanteil.

Die Parameter können auch zur Laufzeit in VBScript verändert werden.

Betrachten Sie das Beispiel oben und verändern sie die Werte für die Farbanteile.

Variation:
Geben Sie dem Body einen Hintergrund und verändern Sie die Werte für den Transparenzanteil.
Tip: Der Gradientfilter kann auch für <DIV>, <TABLE>, <P> und Formulare verwendet werden.

Lektion 1a: Popup definieren.

Als Popup bezeichnet man Fenster, die von einer Internetexplorer oder Outlook-Express erzeugt werden können und dabei den ganzen Bildschirm einnehmen. In der Regel bleibt nur die Taskleiste sichtbar. Dadurch, das die störenden Leisten und Menüs des E-Mailprogramms ausgeblendet werden, ergeben sich mehr Möglichkeiten, kreative Bildschirm Motive und Animationen zu gestalten.
Leider gibt es im Internet auch aufdringliche Popup's, in der Hauptsache Werbung, die sich nicht mehr schließen lassen. Deshalb kamen Popup`s in Verruf, was sehr schade ist.
Um das zu verhindern wurde auch im Internetexplorer und dadurch auch in Outlook-Express Sperren gegen Popup's eingebaut. Um Popup's betrachten zu können, müssen daher im Internetexplorer einige Einstellungen verändert werden.
Ich habe Ihnen die zwei wichtigsten Einstellungen für den IE7 in den beiden nachfolgenden Bildern dargestellt.

Genauere Hinweise finden Sie zum Beispiel unter:
http://www.lettermanstationery.com/

Dort finden Sie auch Filter, die Sie möglicherweise noch nicht installiert haben.

Dieses Kapitel soll Ihnen die Grundlagen zur Erstellung von Popup's zeigen.
Wir werden nun Schritt für Schritt ein Popup erzeugen und nach und nach mit Leben füllen.

Erst mal ein paar theoretische Grundlagen.

Erstens: **ein Popup ist in der Regel der Inhalt eines Div's.**

Zweitens: **ein Popup sollte wenn möglich immer durch einen Schalter gestartet werden und nicht sofort beim Start aktiv sein.**

Drittens: **Wichtig!!! Es muss immer!! Ein Schalter vorhanden sein, um das Popup zu schließen.**

Wir wollen in die Praxis einsteigen.

Mit dem Paragraph mit der ID=start werde ich nun den Befehl zum Starten eines Script geben, das unser Popup am Monitor zeigen soll. Zugleich erstelle ich ein DIV, das unser Popup darstellen wird.

Ergänzen und ändern unseren Html-Code nun um, wie unten gezeigt.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body     { filter:progid:DXImageTransform.Microsoft.Gradient
          (startcolorstr=#ff000000,endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
</style>
</head>
<body>
<P id=start onClick=startepopup()>S t a r t</P>
<div id=popup>

</div>
</body>
</html>
```

Erklärung:
`<P id=start onClick=startepopup()>S t a r t</P>`
Der Befehl **onClick =startepopup()** bedeutet: Wenn mit der linken Maustaste auf den Paragraph **id=start** geklickt wird, dann starte eine Scriptroutine die den Namen **startepopup()** hat.
Diese Routine müssen wir erst noch im Scriptteil erstellen. Falls Sie jetzt auf die Vorschau gehen und mit der Maus auf START klicken wird ein Scriptfehler angezeigt.
Momentan den Fehler einfach ignorieren.

Jetzt kommt ein wichtiger Punkt:
denn als nächstes werde ich sofort einen Schalter erzeugen, mit dem ich später mein Popup wieder schließen kann.
Wichtig! Alles was im Popup zu sehen sein soll, muss sich innerhalb des DIV's befinden, auch Stylesheet-Anweisungen.

Außerdem werde ich das Div unsichtbar machen, da es erst beim Öffnen des Popup sichtbar werden soll.
Beachten Sie, das der CSS-Teil des Close-Schalters nun im Popup-Div steht.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body     { filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff000000,
           endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
#popup  { visibility:hidden }
</style>
</head>
<body>
<P id=start onClick=startepopup()>S t a r t</P>
<div id=popup>
<p id=ende onClick="parent.closepopup()">C l o s e</p>
<style type=text/css>
          #ende    { position:absolute; color:white;font-size:24pt;font-style:italic;right:20;bottom:20}
</style>
</div>
</body>
</html>
```

Sie sehen, der Endeknopf wird im! DIV erzeugt, in dem wir das Popup definieren. Das ist wichtig, sonst würde er später nicht funktionieren.
Nun können wir den Knopf wieder mit einer Stylesheet-Anweisung formatieren.
Das dürfen wir nicht! im Headbereich wie vorher, der Knopf steht ja im Popupbereich und muss!! deshalb im Div nach!! den Tags stehen. Am besten nach den HTML-Anweisungen im Popup-DIV.

Erklärung des Codes:

onClick=parent.closepoup()
Klingt jetzt ein wenig kompliziert ist aber eigentlich ganz logisch.
Der Befehl steht im Popup-DIV, das später ein eigenes Fenster darstellt.
Dieses Fenster wurde aufgerufen von unserem Html-Dokument, ist also sozusagen ein Ableger unseres Dokuments. Das Hauptdokument, von dem aus unser Fenster erzeugt wird, wird als **parent,** auf deutsch Eltern, bezeichnet.
Die Scriptroutine, die das Fenster schließt, ist im Hauptdokument, also im 'Elternteil', daher **parent.closepopup()**. Die Routine **closepopup()** befindet sich natürlich in unserem Scriptteil.

Nun ist es an der Zeit, ans Programmieren zu gehen und die Voraussetzungen für das Popup zu erzeugen.
Zuerst folgt dessen Definition, die immer den selben Aufbau hat:

```
<script language=vbscript>
ax=screen.availWidth
ay=screen.availHeight
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.backgroundcolor="blue"
</script>
```

Die einzelnen Befehle im Überblick:

ax=screen.availWidth = speichere in der Variablen **ax** die verfügbare Breite des Bildschirms

ay=screen.availHeight = speichere in der Variablen **ay** die verfügbare Höhe des Bildschirms

set po=window.createPopup() = erzeuge in der Variablen **po** ein neues Popup

set poBody=po.document.body = erzeuge in der Variablen **poBody** einen neuen Body für das Popup das den Namen po hat.

set poall=poBody.all = speichere in der Variablen **poall** alle Tags, die im Popup vorhanden sind. Diese Variable ist für uns enorm wichtig, da wir später alle! Html-Objekte mit diesem Variablennamen vorneweg ansprechen müssen.

PoBody.innerHTML=popup.innerHTML = der Inhalt des neuen Popupfensters ist der Inhalt des Div's mit dem Namen **popup.**

poBody.style.backgroundColor="blue" Der Hintergrund unseres neuen Fensters soll erst mal blau sein. Sie können selbstverständlich auch Hintergrundgrafiken verwenden. Das werden wir allerdings erst später programmtechnisch machen.

Damit haben wir ein fertiges Popup erstellt, das aber noch nicht dargestellt wird.

In der nächsten Routine werden wir das Popup starten.

Sie errinnern sich, die Aufrufroutine steht in Paragraph **<p id=start>**, die Routine muss heißen: **startepopup().**
Fügen wir nun folgenden Code, rot dargestellt, ein:

```
<script language=vbscript>
ax=screen.availWidth
ay=screen.availHeight
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.backgroundcolor="blue"
sub startepopup()
        po.show 0,0,ax,ay
        poall.ende.style.visibility="visible"
end sub

sub closepopup()
        parent.po.hide()
end sub

</script>
```

Erklärung:

po.show(zahl1,zah2,zahl3,zahl4) aktiviert das Popoup:

die vier Zahlen von links nach rechts gaben dabei folgendes Fensterpunkte an:

Zahl1 = Position der linken Seite des neuen Fensters
Zahl2 = Position des neuen Fensters von oben
Zahl3 = Breite des neuen Fensters
Zahl4 = Höhe des neuen Fensters

In unserem Fall erscheint das Popup-Fenster also:
In der linken oberen Ecke des Monitors,
es ist in voller Monitorbreite und –höhe zu sehen.
Die Höhe ist allerdings beschränkt durch die Taskleiste von Windows.

Den Schalter mit der id=ende, setzen wir auf sichtbar damit das Fenster wieder geschlossen werden kann.

Die Routine closepopup() schließt unser Popup-Fenster wieder.

Testen Sie nun das gesamte Dokument im Internetexplorer und verändern sie die Werte in po.show(x,x,x,x) um selbet zu sehen, wie sich das Fenster entsprechend verändert.

Als nächstes möchte ich den Hintergrund des Popup ändern. Dazu möchte ich eine Hintergrundgrafik verwenden. Sie finden Sie im Ordner unter dem Namen: h33.jpg.

Dazu müssen wir die Grafik im HTML-Teil, nicht im Popup-DIV zuerst laden, in den Bereich außerhalb des Monitors verschieben und dann im Script als Hintergrund dem Popup zuordnen.

Wir binden das Bild im Body-Bereich ein, verschieben es an eine nicht sichtbare Stelle und fügen im Code einen Bezug dazu ein.
Diese Art der Hintergrundeinbindung sollte Ihnen bekannt sein aus den Kursteil 1.

Die gesamte Seite im Code sieht nun aus wie gleich nachfolgend gezeigt..
Machen Sie die Änderungen und Ergänzungen, die dort rot dargestellt sind.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body      { filter:progid:DXImageTransform.Microsoft.Gradient
          (startcolorstr=#ff000000,endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
#popup  { visibility:hidden}
#b1               { position:absolute;left:-3000 }     'Das Bild wird nach links außerhalb des sichtbaren Bereichs
                                                       'verschoben
</style>
</head>
<body>
<img src=h33.jpg id=b1>
<P id=start onClick=startepopup()>S t a r t</P>
<div id=popup>
<p id=ende onClick="parent.closepopup()">C l o s e</p>
<style type=text/css>
#ende    { position:absolute;color:white;font-size:24pt;font-style:italic;right:20;bottom:20}
</style>
</div>
<script language=vbscript>
ax=screen.availWidth
ay=screen.availHeight
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.backgroundImage="url(" & b1.src & ")"  'Das DIV bekommt den Hintergrund.
sub startepopup()
po.show 0,0,ax,ay
poall.ende.style.visibility="visible"
end sub
sub closepopup()
parent.po.hide()
end sub
</script>
</body>
</html>
```

Nun haben wir ein Hintergrundbild richtig eingebunden.
Sie sehen, das Popup bekommt einen Hintergrund und mit dem Schalter CLOSE kann ich es wieder schließen.

Als nächstes möchte ich nun ein Bild in der Mitte zentriert darstellen.
Das Bild finden Sie im Ordner unter: **rahmenpi3.jpg**
Da dieses Bild im Popup angezeigt werden soll, muss es auch im Popup-DIV erstellt werden.

Dazu gebe ich folgendes ein, rot dargestellt:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body      { filter:progid:DXImageTransform.Microsoft.Gradient
          (startcolorstr=#ff000000,endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
#popup  { visibility:hidden}
#b1                { position:absolute;left:-3000 }
</style>
</head>
<body>
<img src=h33.jpg id=b1>
<P id=start onClick=startepopup()>S t a r t</P>
<div id=popup>
<p id=ende onClick="parent.closepopup()">C l o s e</p>
<img src=rahmenpi3.jpg id=b2>
<style type=text/css>
#ende    { position:absolute;color:white;font-size:24pt;font-style:italic;right:20;bottom:20}
#b2                { position:absolute }
</style>
</div>
<script language=vbscript>
ax=screen.availWidth
ay=screen.availHeight
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.backgroundImage="url(" & b1.src & ")"
sub startepopup()
po.show 0,0,ax,ay
poall.ende.style.visibility="visible"
poall.b2.style.left=(ax-poall.b2.clientwidth)/2
poall.b2.style.top=(ay-poall.b2.clientheight)/2
end sub
sub closepopup()
parent.po.hide()
end sub
</script>
</body>
</html>
```

Erklärung:
Das Bild bekommt die Bezeichnung: ID=b2. In der Stylesheetanweisung, die ebenfalls im Popup-DIV sein muss, wird es absolut positioniert.
Nun kommen im Script zwei Zeilen die das Bild an die richtige Stelle bringen und jetzt kommt dieses poall, das ich schon angesprochen habe, zum tragen.
Für alle Elemente in popup-DIV muss vorher der Variablenname eingetragen werden den wir ja folgendermaßen definiert hatten:
set poall=poBody.all

poall.b2.style.left=(ax-poall.b2.clientwidth)/2 - wird in der horizontalen Mitte positioniert
poall.b2.style.top=(ay-poall.b2.clientheight)/2 – wird in der vertikalen Mitte zentriert

Als letztes nun noch ein wenig Animation.

Dazu mache ich folgendes:

1. Das Popup-DIV bekommt einen neuen Hintergrund: h9.jpg

2. Ich erstelle ein neues Div im popup-Div mit dem Namen id=neu

3. das neue div bekommt einen eigenen Hintergrund, id=b1

4. das neue div soll in der Größe ebenfalls den ganzen Bildschirm einnehmen.

5. Der Hintergrund des neuen div soll leicht transparent sein (Alphafilter)

6. Der Hintergrund des popup-DIV soll langsam von oben nach unten scrollen

Der gesamte Code sieht nun so aus, wie auf der nächsten Seite dargestellt.
Wie immer wird alles Neue rot dargestellt, blau muss als ganze Zeile eingefügt werden.

Zu Punkt1:
Bild einfügen, -3000px links positionieren.

```
<img src=h9.jpg id=b3>
#b3                 { position:absolute;left:-3000 }
```

Zu Punkt 2:

```
<div id=neu></div>
```

Zu Punkt 3:

```
#b1                 { position:absolute;left:-3000 }
<img src=h33.jpg id=b1>
poall.neu.style.backgroundImage="url(" & b1.src & ")"
```

Zu Punkt 4:

```
poall.neu.style.width=ax
poall.neu.style.height=ay
```

Zu Punkt 5:

```
#neu     { position:absolute;top:0;left:0;filter:progid:DXImageTransform.Microsoft.Alpha(opacity=90,style=0)}
```

Zu Punkt 6:

```
lauf()
sub lauf()
x=x+1
poBody.style.backgroundpositiony=x
settimeout "lauf()",32
end sub
```

Dadurch, das der Hintergrund leicht transparent ist, sieht es fast so aus als würde Wassertropfen nach unten perlen.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Lektion 1   Grundlagen Popup   2005 by Werner Jakob</title>
<script language=vbscript>
on Error resume next
</script>
<style type=text/css>
body      { filter:progid:DXImageTransform.Microsoft.Gradient
          (startcolorstr=#ff000000,endcolorstr=#ffff0000,gradienttype=1)}
#start    { position:absolute;color:yellow;font-size:25;bottom:20;right:20}
#popup  { visibility:hidden}
#b1                 { position:absolute;left:-3000 }
#b3                 { position:absolute;left:-3000 }
</style>
</head>
<body>
<img src=h33.jpg id=b1>
<img src=h9.jpg id=b3>
<P id=start onClick=startepopup()>S t a r t</P>
<div id=popup>
<div id=neu></div>
<p id=ende onClick="parent.closepopup()">C l o s e</p>
<img src=rahmenpi3.jpg id=b2>
<style type=text/css>
#ende    { position:absolute;color:white;font-size:24pt;font-style:italic;right:20;bottom:20}
#b2                 { position:absolute }
#neu      { position:absolute;top:0;left:0;filter:progid:DXImageTransform.Microsoft.Alpha(opacity=90,style=0)}
</style>
</div>
<script language=vbscript>
x=0
ax=screen.availWidth
ay=screen.availHeight
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.backgroundImage="url(" & b3.src & ")"
sub startepopup()
po.show 0,0,ax,ay
poall.ende.style.visibility="visible"
poall.b2.style.left=(ax-poall.b2.clientwidth)/2
poall.b2.style.top=(ay-poall.b2.clientheight)/2
poall.neu.style.width=ax
poall.neu.style.height=ay
poall.neu.style.backgroundImage="url(" & b1.src & ")"
lauf()
end sub
sub lauf()
x=x+1
poBody.style.backgroundpositiony=x
settimeout "lauf()",32
end sub
sub closepopup()
parent.po.hide()
end sub
</script>
</body>
</html>
```

Lektion 2: Das Popup gleichmäßig von links oben einblenden

In dieser Lektion möchte ich Ihnen zeigen, wie man durch kleine Berechnungen Einblendeffekte zur Darstellung eines Popup's erzeugen kann.
Sie sollten nach jeder Änderung des Codes eine Seitenvorschau durchführen, um genau zu sehen, was die einzelnen Befehle bewirken.

Folgende Änderungen möchte ich hinzufügen:

1. Um das Popup einen Rand legen.
2. Das Seitenverhältnis von Breite zur Höhe berechnen.
3. Wenn die Maus auf das Wort Ende kommt, soll sich der Mauszeiger in eine Hand wandeln.
4. Das Popup vorbereiten, den Endeknopf und das Bild in der Mitte erst mal ausblenden, erst wenn das Popup ganz am Bildschirm dargestellt ist, die Elemente einblenden.
5. Das Popup von links oben einblenden.

Beginnen wir mit Schritt 1.

Öffnen Sie aus dem Ordner Lektion2 die Datei Lektion2.html.

Fügen Sie nun folgende Zeile, die rot dargestellt ist, in den Code ein.

```
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.border="#D4A976 4 groove"
PoBody.style.backgroundImage="url(" & b3.src & ")"
```

Damit wird ein hellbrauner Rand um das Popup gezeichnet.

Nun ein paar Überlegungen. Der Bildschirm hat eine gewisse Breite und eine gewisse Höhe. Um den Faktor zu bestimmen, müssen wir einfach die Breite durch die Höhe teilen.
Das mache ich hier:

```
x=0
ax=screen.availWidth
ay=screen.availHeight
adiv=ax/ay
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
```

Dieses Maß brauche ich später, um ein gleichmäßiges Öffnen des Popup zu gewährleisten. Es gibt das Verhältnis von Bildschirmbreite zur Bildschirmhöhe an.
Nehmen wir an, Sie haben eine Auflösung von 1024x768 eingestellt, dann wäre das Verhältnis 1024 : 768 = 1.333333333
Das heißt: wenn sich die Höhe des Popup um 2px ändert, dann muss sich die Breite um 2*1.333333 Pixel ändern, usw.

Nun noch ein paar Vorbereitungen.
Da das Popup langsam geöffnet werden soll, von der linken oberen Ecke nach rechts und nach unten, ist es sinnvoll, den Inhalt des Popup erst mal zu verstecken und auch das Hauptbild, also die Dame im Rahmen soll erst später zu sehen sein.
Das mache ich mit den nächsten Anweisungen.

```
<div id=popup>
<div id=neu></div>
<p id=ende onClick="parent.closepopup()">C l o s e</p>
<img src=rahmenpi3.jpg id=b2>
<style type=text/css>
#ende   {       position:absolute;color:white;font-size:24pt;
                font-style:italic;right:20;bottom:20;cursor:pointer;visibility:hidden}
#b2     {       position:absolute;visibility:hidden }
#neu    {       position:absolute;top:0;left:0;
                filter:progid:DXImageTransform.Microsoft.Alpha(opacity=90,style=0) }
</style>
```

Die Anweisung: cursor:pointer wandelt den Mauszeiger um, sobald die Maus auf das formatierte Objekt wandert.

So, nun ein paar Überlegungen zum Einblenden des Popup.
Sehen wir uns den Code noch mal an:

Po.show 0,0,ax,ay

So sieht die Zeile bisher aus.
Nochmal die Bedeutung der vier Zahlen.

erste Zahl	=	Position der linken, oberen Ecke des Popup zum linken Seitenrand
zweite Zahl	=	Position der linken, oberen Ecke des Popup zum oberen Seitenrand
dritte Zahl	=	Breite des Popup
vierte Zahl	=	Höhe des Popup

Ersetzen sie nun mal die Zeile im Code: po.show 0,0,ax,ax durch po.show 100,100,100,100

Sie stellen fest, dass das Popup nur noch 100x100 Pixel groß ist, außerdem beginnt es 100 Pixel von oben und 100 Pixel von rechts.

Mit einer Schleife möchte ich nun das Popup in der linken, oberen Ecke des Bildschirms beginnen lassen und anschließend langsam, gleichmäßig größer werden lassen.
Wenn das Popup ganz geöffnet ist, soll das Bild dargestellt werden.

Da der Hintergrund schon beim Öffnen animiert sein soll, werde ich das als erstes aufrufen.
Das geschieht in der **Startpopup()** Routine.
Erst nachdem das geschehen ist, verzweige ich zur Routine in der **poshow 0,0**.
Zwei Variable werden hochgezählt, die die Breite und die Höhe des Popup bestimmen.
Ist die richtige Breite erreicht, dann wird das Popup noch mal sauber in der Größe angepasst und das Bild sowie der Ende-Schalter dargestellt.

```
<script language=vbscript>
dim px,py   ' Vaiablen für die Breite und Höhe des Popup
x=0
ax=screen.availWidth
ay=screen.availHeight
adiv=ax/ay        'Variable die das Verhältnis von Breite zur Höhe beinhaltet
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.border="#D4A976 4 groove"
PoBody.style.backgroundImage="url(" & b3.src & ")"

'Erster Teil zum Öffnen des Popup. Hier wird die Größe und der Hintergrund des zweiten Div's festgelegt
sub startepopup()
poall.neu.style.width=ax
poall.neu.style.height=ay
poall.neu.style.backgroundImage="url(" & b1.src & ")"
lauf()                    'animiere den Hintergrund
startepopup2()            'Öffne das Popup
end sub

'Die Schleife, die das Popup langsam öffnet
sub startepopup2()
px=px+2*adiv    'neue  Breite
py=py+2         ' neue Höhe
if px>ax then  'Wenn Breite erreicht dann stelle Popup nochmal sauber dar,zentriere das Bild
                                    ' mache das Bild und den Ende-Schalter sichtbar
        po.show 0,0,ax,ay
        poall.b2.style.left=(ax-poall.b2.clientwidth)/2 'Das Bild in die Mitte des Fensters stellen, horizontal
        poall.b2.style.top=(ay-poall.b2.clientheight)/2 'Das Bild in die Mitte des Fensters stellen, vertikal
        poall.b2.style.visibility="visible" 'Bild sichtbar machen
        poall.ende.style.visibility="visible"  ' Endeknopf sichtbar machen
        exit sub
end if
po.show 0,0,px,py  'Popup mit den Schleifenmaßen px und py zeigen
settimeout "startepopup2()",32 'berechne alle 32`tausendstel die Position und Größe neu
end sub

'Bewegung des Hintergrundes
sub lauf()
x=x+1
poBody.style.backgroundpositiony=x
settimeout "lauf()",32
end sub

'Popup schließen
sub closepopup()
parent.po.hide()
end sub
</script>
```

Lektion 3: Popup von der Mitte aus öffnen.

In dieser Lektion wollen wir das Öffnen des Popup etwas umgestalten.
Das Popup soll von der Mitte beginnend zuerst mit einer Breite von 2 Pixel die Höhe erreichen, anschließend von der Mitte aus die Breite gleichmäßig erweitern.

Dazu erstelle ich 2 unterschiedliche Routinen, die erste für die Höhe, die zweite für die Breite.

Dazu wieder zuerst die theoretische Betrachtung:

Hier sehen Sie wiederl die Zeile zum Anzeigen des Popup.

po.show links, oben, breite, höhe

Nehmen wir folgende Variablen an:
ax = Breite des Monitors
ay = Höhe des Monitors
px = Zählveriable für die Breite
py = Zählvariable für die Höhe

Um den genauen Mittelpunkt zum Starten zu erreichen, wenn die Breite des Popup 2 px betragen soll, muss die Zeile zum Öffnen so aussehen:

po.show ax/2 , ay/2 , 2 , 0

Nun muss eine Schleife erzeugt werden, die erstens den Anfangspunkt nach oben verschiebt und zweitens die Höhe um die doppelte Anzahl von px vergrößert, um ein gleichmäßiges Vergrößern des Popup gewährleistet. Wenn die richtige Höhe erreicht ist, springen wir zur nächsten Routine, die die Breite bearbeitet.

Die Scheife sieht dann so aus:

```
sub startepopup2()
py=py+2  'Die Höhe des Popup wird pro Schleifendurchgang um 2px erhöht.
if py*2>ay then              'Falls die Höhe größer ist als die Höhe der eingestellten
                             Bildschirmauflösung
        startepopup3()       'dann springe zur Berechnung der Breite
        exit sub             'verlasse diese Routine
end if
po.show ax/2,ay/2-py,2,py*2  ' Zeige das Popup bei jedem Schleifendurchgang
settimeout "startepopup2()",32   'Wiederhole die Schleife alle 32`tausenstel Sekunden.
end sub
```

ax/2 = linker Startpunkt ist halbe Bildschirmbreite
ay/2 –py = oberer Startpunkt, wird bei jeden Schleifendurchgang um 2 px vermindert
2 = Breite des Popup
py*2 = Höhe des Popup, doppelter Schleifenwert.

Ersetzen Sie nun die alte Routine startepopup2() mit dem oben gezeigten Code.

In der zweiten Routine bearbeiten wir nun die Breite.

Unser Popup hat die optimale Höhe erreicht, wir müssen nun die Breite gleichmäßig erweitern und den linken Anfangspunkt nach links wandern lassen.

Die Routine sieht so aus:

```
sub startepopup3()
px=px +2      'Erhöhe die Breite des Popup um 2px bei jedem Durchgang
if px*2>ax then   'Falls px*2 größer als die Bildschitmbreite
        po.show 0,0,ax,ay  'stelle das Popup genau dar.
        poall.b2.style.left=(ax-poall.b2.clientwidth)/2  'Zentriere das Bild
        poall.b2.style.top=(ay-poall.b2.clientheight)/2
        poall.b2.style.visibility="visible"   'und stelle es dar
        poall.ende.style.visibility="visible"  'mach den Ende-Knopf sichtbar
        exit sub
end if
po.show ax/2-px,0,px*2,ay 'Zeige nach jedem Schleifendurchgang das Popup
settimeout "startepopup3()",32
end sub
```

Erklärung:
po.show ax/2-px , 0 , px*2 , ay

ax/2 ist die Hälfte der Bildschirmbreite und der linke Ausgangspunkt.

Durch ax/2-px ziehen wir bei jedem Schleifendurchgang zwei Pixel ab, was den Anfangspunkt immer weiter nach 0 zu bewegt, also immer weiter nach links.

Die Breite ändern wir mit px*2, also der doppelten Anzahl von Pixel wie den linken Punkt, so das die Breite gleichmäßig erweitert wird.

Ist die gewünschte Breite erreicht, dann stellen wir das Popup wieder sauber dar, zentrieren das Bild und machen das Bild sowie den Endeknopf sichtbar.

Fügen Sie nun die oben genannte Routine in den Code ein, am besten nach startepopup2().

Damit sind wir vorläufig am Ende der Popuplektionen angekommen.

Später werden Sie noch erfahren, wie sie Multiple Popup's öffnen können.

Lektion 4a: Bild zerlegen mit DIV's

In diesem Kapitel möchte ich Ihnen ein paar nicht so bekannte Tricks zeigen.
Als erstes betrachten wir, wie ein Bild in zerlegt werden kann, mit Hilfe von Div's oder Tabellenfeldern. Im Beispiel verwende ich DIV's.

Grundidee:
Ich habe ein Bild, das ich in drei Teile zerlegen möchte.
Anschließend sollen die 3 Einzelbilder animiert zusammengehen und das Gesamtbild bilden.
Dazu verwende ich das Bild im Ordner Lektion4a, es hat den Dateinamen bild1.jpg.

Beginnen wir mit dem Grundaufbau.
Laden Sie dazu die Datei Lektion4.html in Ihren Editor oder Ihr bevorzugtes HTML-Erstellungsprogramm.

Der Code sieht nun so aus:

```
<html>
<head>
<title>Lektion 4a  -  zerlegtes Bild</title>
</head>
<style type="text/css">
body     {          filter:progid:DXImageTransform.Microsoft.Gradient
                    (startcolorstr=#ff960000,endcolorstr=#ffFF9650,gradienttype=0);
                    border:6 #B7770D groove}
</style>
<body scroll=no>
<img id=bld1 src="Bild1.jpg">
<div id=teil1></div>
<div id=teil2></div>
<div id=teil3></div>
</body>
</html>
```

Der Body-Teil hat einen Verlaufsfilter, außerdem wird das Scrollen verhindert.
Es gibt ein Bild mit dem Dateinamen Bild1.jpg das Bild hat die ID=bld1.
Es sind noch da: 3 DIV's mit den ID's teil1 - teil3.
Sehen Sie sich den Code in der Vorschau bzw. im Internet Explorer an.

Dieses Bild möchte ich nun in 3 Teile zerlegen.
Dazu verwende ich die drei Div's.
Aber zuerst werde ich die DIV's mal provisorisch positionieren und das Bild selbst aus dem sichtbaren Bereich verschieben.

Fügen Sie folgende rot markierten Stylesheetanweisungen hinzu:

```
<style type="text/css">
body    {       filter:progid:DXImageTransform.Microsoft.Gradient
                (startcolorstr=#ff960000,endcolorstr=#ffFF9650,gradienttype=0);
                border:6 #B7770D groove}
#bld1   {       position:absolute;left:-3000 }
#teil1  {       position:absolute;left:50;border:white 1 groove }
#teil2  {       position:absolute;border:white 1 groove }
#teil3  {       position:absolute;right:50;border:white 1 groove }
</style>
```

Den DIV's habe ich eine Umrandung gegeben, Farbe weiß, 1 Pixel breit, Aussehen, groove.
Wenn Sie nun wieder ihr Dokument in der Vorschau betrachten, werden Sie nichts außer 3 kleinen Linien sehen auf unserem Hintergrund sehen. Das Bild haben wir verschoben, die DIV's haben keinen Inhalt. Es ist nur der Rahmen jedes einzelnen DIV's in Größe der Schrifthöhe sichtbar.

Um die DIV's sichtbar zu machen, müssen wir Ihnen einen Inhalt geben und eine Größe festsetzen.

Nun, wir wollen unser Bild in diese 3 DIV's zerteilen. Also muss jeder einzelne DIV 1/3 der Bildbreite werden.
Die Höhe der Div's soll der Höhe unseres Bildes entsprechen.
Also erstelle ich ein Script, das mir diese Daten liefert.

Fügen Sie in ihr Dokument ein:

```
<script language=vbscript>
teil1.style.height=bld1.clientHeight
teil1.style.width=bld1.clientWidth/3
</script>
```

Die Höhe des ersten DIV mit der ID=reil1 soll sein: **bld1.clientheight**, das ist die Höhe meines Bildes.
Die Breite soll sein, Bildbreite:3, entspricht **bld1.clientwidth/3.**

Betrachten Sie wieder alles in der Vorschau, Sie sehen DIV id=teil1 in der gewünschten Höhe und Breite.

Als nächstes lade ich in dieses Div mein Bild als Hintergrund.

Fügen Sie ein:

```
<script language=vbscript>
teil1.style.height=bld1.clientHeight
teil1.style.width=bld1.clientWidth/3
teil1.style.backgroundImage="url(" & bld1.src &")"
</script
```

Jetzt wird im ersten DIV das erste Drittel meines Bildes dargestellt.
Ich habe das Bild als Hintergrund meines DIV's , das geschieht immer mit Beginn des Bildes links oben, da das Bild größer ist als mein DIV, wird das Bild entsprechend abgeschnitten.

Als nächstes bearbeite ich das zweite DIV.
Es soll in der Mitte dargestellt werden, und das zweite Drittel meines Bildes aufnehmen.

```
<script language=vbscript>
teil1.style.height=bld1.clientHeight
teil1.style.width=bld1.clientWidth/3
teil1.style.backgroundImage="url(" & bld1.src &")"
teil2.style.height=bld1.clientHeight
teil2.style.width=bld1.clientwidth/3
teil2.style.left=(document.body.clientWidth-teil2.clientwidth)/2
</script>
```

Erklärung:
Der sichtbare Bereich meines Dokuments in der Vorschau, oder im IE, oder in OE wird mit;
document.body.clientwidth bestimmt.
Der Div wird durch die Berechnung in die Mitte gesetzt.
Betrachten Sie das Ergebnis wieder in der Vorschau.

Doch wie bekomme ich nun das zweite Drittel meines Bildes in dieses DIV?
Nun, sie erinnern sich doch noch an die erste Lektion mit dem animierten Hintergrund.
Dort haben wir den Hintergrund verschoben mit: objekt.style.backgroundpositionX=x.
Diese Prinzip wenden wir nun hier an.
Ich gebe dem DIV als Hintergrund mein Bild: id=bld1 und verschiebe es anschließend um ein Drittel des Bildes nach links.

Fügen Sie folgende Zeile hinzu:

```
<script language=vbscript>
teil1.style.height=bld1.clientHeight
teil1.style.width=bld1.clientWidth/3
teil1.style.backgroundImage="url(" & bld1.src &")"
teil2.style.height=bld1.clientHeight
teil2.style.width=bld1.clientwidth/3
teil2.style.left=(document.body.clientWidth-teil2.clientWidth)/2
teil2.style.backgroundImage="url(" & bld1.src & ")"
teil2.style.backgroundPositionX=-bld1.clientWidth/3
</script>
```

Wenn Sie jetzt Ihr Dokument wieder in der Vorschau betrachten, sehen Sie das Ergebnis!

Versuchen Sie nun selbst, ohne den Code unterhalb anzusehen, den letzten Teil des Bildes in DIV id=tei3 zu bekommen.

Der Code:

```
<script language=vbscript>
teil1.style.height=bld1.clientHeight
teil1.style.width=bld1.clientWidth/3
teil1.style.backgroundImage="url(" & bld1.src &")"
teil2.style.height=bld1.clientHeight
teil2.style.width=bld1.clientwidth/3
teil2.style.left=(document.body.clientWidth-teil2.clientWidth)/2
teil2.style.backgroundImage="url(" & bld1.src & ")"
teil2.style.backgroundPositionX=-bld1.clientWidth/3
teil3.style.height=bld1.clientHeight
teil3.style.width=bld1.clientWidth/3
teil3.style.backgroundImage="url(" & bld1.src &")"
teil3.style.backgroundPositionX=-bld1.clientWidth/3*2
</script>
```

Doch sehen wir uns nun mal den Code etwas genauer an....
Ich weiß nicht, wie es Ihnen geht, aber ich bin kein Freund von langen Code eintippen.
Ich sehe im Code viele Zeilen, die immer wieder vorkommen.
Da muss sich doch was ändern lassen.
Ich sehe zum Beispiel, das document.body.clientWidth, document.body.clientHeight, bld1.clientwidth und bld1.clientheight immer wieder vorkommen.
Was ist, wenn ich diese in einer Variablen berechne und dann nur noch den Variablenname eingeben muss?

Sagen wir doch:

```
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth
by=bld1.clientHeight
```

und formen den ganzen Code entsprechen um.

```
<script language=vbscript>
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth
by=bld1.clientHeight
teil1.style.height=by
teil1.style.width=bx/3
teil1.style.backgroundImage="url(" & bld1.src &")"
teil2.style.height=by
teil2.style.width=bx/3
teil2.style.left=(ax-bx/3)/2
teil2.style.backgroundImage="url(" & bld1.src & ")"
teil2.style.backgroundPositionX=-bx/3
teil3.style.height=by
teil3.style.width=bx/3
teil3.style.backgroundImage="url(" & bld1.src &")"
teil3.style.backgroundPositionX=-bx/3*2
</script>
```

Das erspart schon mal etwas Schreinarbeit.
Aber mir ist das immer noch zu viel.
Es kommen einige Zeilen öfters vor, da kann man bestimmt auch noch was tun.

Dafür gibt es Schleifen!
Eine Schleife hat den Aufbau:

```
for Variable = Beginn  to Ende.
        Anweisung
        Anweisung
        usw.........
next
```

Nehmen wir an, die der Variablenname ist i und Beginn soll 0 sein, Ende soll 2 sein, dann würde die Schleife so aussehen:

```
for i = 0 to 2
        Anweisung 1
        Anweisung 2
        usw.....
next
```

In den Lektionen 1-10 haben sie ein Array kennen gelernt.
In einem Array kann man z.B. Objekte mit Ihren ID-Namen abspeichern und sie später wieder auslesen und ansprechen.

Jetzt erstelle ich ein Array, das meine 3 DIV's beinhaltet.

```
bi=array(teil1,teil2,teil3)
```

Nun kann ich die DIV`s ansprechen über.

DIV mit id=teil1 : bi(0)
DIV mit id=teil2 : bi(1)
DIV mir id=teil2 : bi(2)

Damit kann ich mir einige Schreibarbeit ersparen.

Sehen wir uns auf der nächsten Seite nun den neuen Code an.

```
<script language=vbscript>
bi=array(teil1,teil2,teil3)
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth/3
by=bld1.clientHeight
for i=0 to 2
        bi(i).style.height=by
        bi(i).style.width=bx
        bi(i).style.backgroundImage="url(" & bld1.src &")"
        bi(i).style.backgroundposition=-bx*i
next
teil2.style.left=(ax-bx)/2
</script>
```

Sehen wir uns die Schleife mal an:

1. Durchgang, gegeben: **i=0**

bi(i).style.height=by
bi(i) → bi(0) → teil1 = by (by=Höhe des Bildes)
Das heißt: im ersten Durchgand wird die Höhe des DIV mit ID=teil1 so hoch wie das Bild

bi(i).style width=bx
bi(i) → bi(0) → teil1 = bx (bx=breite des Bildes/3

bi(i).style.backgroundImage="url(" & bld1.src &")"
bi(i) → bi(0) → teil1 = URL von Bild.jpg

bi(i).style.backgroundposition=-bx*i
bi(i) → bi(0) → teil1 = backgroundpositionX

Das Ganze wiederholt sich noch zweimal, wobei i jedes Mal um 1 erhöht wird.
Bei i=1 folgt:
bi(i) → bi(1) → teil2

Und beim nächsten Schleifendurchgang: i=2
Bi(i) → bi(2) → teil3

Testen Sie das Dokument bis hierher.

Im nächsten Schritt wollen wir das die beiden äußeren DIV's zur Mitte wandern, bis sich wieder das ganze Bild zusammenfügt.
Dazu ein paar Überlegungen:

Wir wissen das DIV1 einen gewissen Abstand zu DIV2 hat.
Wir wissen außerdem, das DIV3 den selben Abstand zu DIV2 hat.
Wir können auch die Position der linken Kante von DIV2 bestimmen.
Jetzt brauchen wir also eine zeitgesteuerte Schleife, die die Position von DIV1 nach rechts und von DIV3 nach links verschiebt, bis DIV1 an DIV2 anstößt.

Bildung der Formeln:
Der Abstand A im Bild ist 50px. Siehe Stylesheet, left=50
Die Bildbreite B ist bx. Siehe bx=bld1.clientwidth/3
Der Abstand zwischen den Bildern, C, = linkerrand von DIV2 - A - B
Dieser Abstand C muss überwunden werden, um das DIV1 an DIV2 anzuschließen.
Um den selben Weg muss DIV3 nach links verschoben werden.

Wie mache ich das nun programmtechnisch?
Hier der Code:

```
schiebe=0
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then exit sub
teil1.style.left=50+schiebe
teil3.style.right=50+schiebe
settimeout "lauf()",32
end sub
```

Ich definiere eine Variable, nenne sie 'schiebe' und gebe ihr vor der Schleife den Wert 0.
Bei jedem Schleifendurchgang erhöhe ich die Variable, hier im Beispiel um 1.
Falls der Wert von Schiebe+50+breite des DIV größer ist als die linke Position von DIV 2, dann soll die Schleife verlassen werden und DIV1 1px nach rechts und DIV3 1px nach links verschoben werden.

Hier nun das gesamte Scriptteil bis hierher.
In Ihr Dokument einsetzen, dann wieder Browservorschau oder direkt im Browser betrachten.
Ich habe im Code noch eine Zeitsperre eingebaut, dass das Verschieben erst nach drei Sekunden beginnt.

```
<script language=vbscript>
bi=array(teil1,teil2,teil3)
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth/3
by=bld1.clientHeight
schiebe=0   'Variable zum Verschieben von DIV1  und DIV3
for i=0 to 2
        bi(i).style.height=by
        bi(i).style.width=bx
        bi(i).style.backgroundImage="url(" & bld1.src &")"
        bi(i).style.backgroundposition=-bx*i
        bi(i).style.top=(ay-by)/2
next
teil2.style.left=(ax-bx)/2
settimeout "lauf()",3000   'Nach dem Laden 3 Sekunden gewartet bis zum Start von lauf()
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then
        exit sub
end if
teil1.style.left=50+schiebe
teil3.style.right=50+schiebe
s=settimeout("lauf()",32)
end sub
</script>
```

Jetzt möchte noch einen Zusatzeffekt einbringen.
Nach einer kleinen Pause sollen die Bildteile zu scrollen anfangen.
Dazu erstellen wir eine weitere Routine, hier nenne ich sie lauf2().
Die Routine soll angesprungen werden, 2 Sekunden nachdem die drei Bildteile zusammengefügt sind.
Dazu fügen wir in die Routine lauf1() folgende Zeilen ein:

```
settimeout "lauf()",3000
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then
        schiebe=0   'Die Variable setze  auf 0, damit sie in der Routine lauf2() bei 0 beginnt.
        settimeout "lauf2()",2000
exit sub
end if
teil1.style.left=50+schiebe
teil3.style.right=50+schiebe
s=settimeout("lauf()",32)
end sub
```

Nun zur Routine lauf2()

Wir wissen, wie die Positionen der Hintergründe der DIV'S versetzt sind.
In DIV1 gar nicht, der Hintergrund steht an Position **0**.
In DIV2 um eine DIV-Breite, als um **–bx.**
In DIV3 um 2 DIV-Breiten, also **–bx*2.**

In einer Variablen, die ich bei jedem Schleifendurchgang um 1 erhöhe, werden nun die einzelnen Hintergründe der DIV's gescrollt.
Das mache ich wieder mir einer **for - next** Schleife.

Dieses Wissen setze ich nun in der Routine lauf2() ein.

```
sub lauf2()
schiebe=schiebe+1   'Die Variable, um deren Wert die Hintergründe der DIV's verschoben wird.
for i=0 to 2
        bi(i).style.backgroundpositionx=schiebe-i*bx
next
settimeout "lauf2()",32
end sub
```

Damit sind wir fast am Ende.

Nun mache ich nur noch zwei Schönheitseinträge.

Der erste ist ein Sound,
der zweite eine Routine, die mir die Seite neu startet, wenn eine Größenänderund am Bildschirm vorgenommen wird.

Im Htmlteil, am Besten direkt nach dem Body-TAG:

```
<bgsound src=l1.wav>
```

Im Scriptteil, als eigene Routine:

```
sub window_onResize()
window.location.reload
end sub
```

Auf der nächsten Seite noch mal der gesamte Code des Dokuments.

```
<html>
<head>
<title>Lektion 4a  -  zerlegtes Bild</title>
</head>
<style type="text/css">
body     {
          filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,endcolorstr=#ffFF965
0,gradienttype=0);
                    border:6 #B7770D groove}
#bld1    {          position:absolute;left:-3000 }
#teil1   {          position:absolute;left:50;border:white 1 groove}
#teil2   {          position:absolute;border:white 1 groove }
#teil3   {          position:absolute;right:50;border:white 1 groove }
</style>
<body scroll=no>
<bgsound src=l1.wav>
<img id=bld1 src="Bild1.jpg">
<div id=teil1></div>
<div id=teil2></div>
<div id=teil3></div>
</body>
<script language=vbscript>
bi=array(teil1,teil2,teil3)
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth/3
by=bld1.clientHeight
schiebe=0
for i=0 to 2
          bi(i).style.height=by
          bi(i).style.width=bx
          bi(i).style.backgroundImage="url(" & bld1.src &")"
          bi(i).style.backgroundposition=-bx*i
          bi(i).style.top=(ay-by)/2
next
teil2.style.left=(ax-bx)/2
settimeout "lauf()",3000
sub window_onResize()
window.location.reload
end sub
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then
          schiebe=0
          settimeout "lauf2()",2000
          exit sub
end if
teil1.style.left=50+schiebe
teil3.style.right=50+schiebe
s=settimeout("lauf()",32)
end sub
sub lauf2()
schiebe=schiebe+1
for i=0 to 2
          bi(i).style.backgroundpositionx=schiebe-i*bx
next
settimeout "lauf2()",32
end sub
</script>
</html>
```

Lektion 4b: geteiltes Bild in DIV's als Popup

In dieser Lektion möchte ich Ihnen Verschiedenes zeigen.

1. Wie wir unser vorhergehendes Dokument in ein Popup umstellen können
2. Wie wir Schaltflächen mit Filtern hinterlegen können.
3. Wie eine Hintergrundmusik zu einem Bestimmten Zeitpunkt ein- und ausgeschaltet werden kann.

Bitte laden Sie aus dem Ordner Lektion 4b, die Datei Lektion 4b.html.
Sie entspricht der fertigen Lektion 4a.

Nun definieren wir die Aufgabenliste.

1. Ich möchte einen Schalter zum Starten des Popup.
2. Ich möchte einen Schalter zum beenden des Popup.
3. Im Startbildschirm möchte ich eine Textbox
4. Beim, Schließen des Popup möchte ich den Text in der Textbox ändern.
5. Die Hintergrundmusik, l2.wav, soll erst beim Öffnen des Popup gespielt werden.
6. Die Hintergrundmusik soll stoppen beim beenden des Popup.

Betrachten wir den HTML-Teil unseres bisherigen Dokuments..

```
<body scroll=no>
<bgsound src=l2.wav>
<img id=bld1 src="Bild1.jpg">
<div id=teil1></div>
<div id=teil2></div>
<div id=teil3></div>
</body>
```

Ich arbeite nun nach und nach die einzelnen Teile ein.
Als erstes den Startknopf für unser Popup und die Hintergrundmusik, diese soll jetzt aber noch nicht zu hören sein.

```
<body scroll=no>
<bgsound id=musik volume=-10000 src=l2.wav" loop=10>
<button id=st onClick=start()>Start</button>
<img id=bld1 src="Bild1.jpg">
<div id=teil1></div>
<div id=teil2></div>
<div id=teil3></div>
</body>
```

Mit volume=-1000 blende ich die Musik aus.
Der Button zum Starten des späteren Popup erhält die ID=st.

Nun baue ich die Textbox ein.

```
<body scroll=no>
<bgsound id=musik volume=-10000 src=l2.wav" loop=10>
<img id=bld1 src="Bild1.jpg">
<button id=st onClick=start()>Start</button>
<DIV id=tx >
<P>Einleitender Text</P>
<P>    Hier kann später ein beliebiger Text  stehen.</P>
</DIV>
<div id=teil1></div>
<div id=teil2></div>
<div id=teil3></div>
</body>
```

Die Textbox bekommt die ID=tx.

Als nächstes schaffe ich die Voraussetzungen für mein Popup.
Dazu verlege ich alle Teile des Dokuments, die später im Popup zu sehen sein sollen, in ein DIV das ich id=popup nenne.

```
<body scroll=no>
<bgsound id=musik volume=-10000 src="l2.wav" loop=10>
<img id=bld1 src="Bild1.jpg">
<button id=st onClick=start()>Start</button>
<DIV id=tx >
<P>Einleitender Text</P>
<P>    Hier kann später ein beliebiger Text
stehen.</P>
</DIV>
<div id=popup>
        <div id=teil1></div>
        <div id=teil2></div>
        <div id=teil3></div>
        <button id=ende onClick=parent.byebye()>Bye Bye</button>
</div>
</body>
```

Zugleich habe ich noch einen Schließen-Button hinzugefügt.

Die Bestandteile meines HTML-Dokument habe ich soweit fertig,
jetzt müssen sie noch formatiert werden.
Das mache ich natürlich mit CSS.
Die HTML-Objekte, die nicht in Popub-DIV stehen, wie gewöhnlich vor dem <body>,
die Objekte im Popup-DIV, wie Sie aus den vorherigen Lektionen wissen,
müssen im Popup-DIV stehen.

Hier nun die Formatierungen:

```
<html>
<head>
<title>Lektion 4b  -  zerlegtes Bild in Popup</title>
</head>
<script language=vbscript>
on Error resume next
</script>
<style type="text/css">
body    {       filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,
                endcolorstr=#ffFF9650,gradienttype=0);
                border:6 #B7770D groove}
#st     {       position:absolute;
                color:white;
                filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,
                endcolorstr=#ffFF9650,gradienttype=0);
                border:white 1 inset;
                right:10;
                bottom:10 }
#popup {       position:absolute;
                visibility:hidden }
#tx     {       position:absolute;
                left:50;
                top:50;
                color:white;
                font-size:16pt;
                width:400;
                border:white 2 inset;
                padding:20px;
                text-align:center }
</style>
<body scroll=no>
<bgsound id=musik volume=-10000 src="l2.wav" loop=10>
<img id=bld1 src="Bild1.jpg">
<button id=st onClick=start()>Start</button>
<DIV id=tx >
<P>Einleitender Text</P>
<P>     Hier kann später ein beliebiger Text
stehen.</P>
</DIV>
<div id=popup>
        <div id=teil1></div>
        <div id=teil2></div>
        <div id=teil3></div>
        <button id=ende onClick=parent.byebye()>Bye Bye</button>
<STYLE type=text/css>
#bld1   {       position:absolute;left:-3000 }
#teil1  {       position:absolute;left:50;border:white 1 groove}
#teil2  {       position:absolute;border:white 1 groove }
#teil3  {       position:absolute;right:50;border:white 1 groove }
#ende   {       position:absolute;
                color:white;
                filter:progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,
                endcolorstr=#ffFF9650,gradienttype=0);
                border:white 1 inset;
                right:10;
                bottom:10 }
</STYLE>
</div>
</body>
```

Erklärung der Objekte außerhalb des Popup-DIV

Fangen wir mit dem Startknopf id=st an.
Die Position ist absolut, Schriftfarbe weiß, Button-Farbe ist ein Gradientfilter.
Er ist 10Pixel vom rechten- und 10Pixel vom unteren Fensterrand enfernt.

Das Popup ist ebenfalls absolute positioniert, beim Start ist es unsichtbar.

Die Textbox ist absolut positioniert, von linken Rand wie auch von oben 50 Pixel entfernt.
Die Breite ist 400 Pixel. Die Schriftfarbe ist weiß, Schriftgröße ist 16pt.
Die Box soll eine Umrandung bekommen, weiß, 2pixel breit vom Typ inset.
Vom Rand der Box zum Text soll ein Abstand von 20Pixel sein. padding=20px.
Zuletzt soll der Text in der Box noch zentriert dargestellt werden.

Nun betrachten wir die Objekte innerhalb des Popup-Div's.

Das Bild, das als Hintergrund der 3 DIV's, teil1 – teil3, dienen soll, wird absolut positioniert und erst mal nach außerhalb des sichtbaren Bereichs verschoben.

Die drei DIV's, teil1 – teil3, haben die selbe Formatierung wie in Lektion4a.

Es bleibt noch der Ende-Button.

Im Prinzip hat er die selben Formatierungen wie schon der Start-Knopf.

Es folgt nun die Programmierung des Scripts.

Unsere Verschieberoutinen können wir so lassen wie sie stehen,
nur einige Parameter müssen geändert werden, und natürlich das Popup muss definiert werden.

Auf der nächsten Seite sehen Sie den Code, wie er in Lektion4 stand.

```
<script language=vbscript>
bi=array(teil1,teil2,teil3)
ax=document.body.clientWidth
ay=document.body.clientheight
bx=bld1.clientwidth/3
by=bld1.clientHeight
schiebe=0
for i=0 to 2
        bi(i).style.height=by
        bi(i).style.width=bx
        bi(i).style.backgroundImage="url(" & bld1.src &")"
        bi(i).style.backgroundposition=-bx*i
        bi(i).style.top=(ay-by)/2
next
teil2.style.left=(ax-bx)/2
settimeout "lauf()",3000
sub window_onResize()
window.location.reload
end sub
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then
        schiebe=0
        settimeout "lauf2()",2000
        exit sub
end if
teil1.style.left=50+schiebe
teil3.style.right=50+schiebe
s=settimeout("lauf()",32)
end sub
sub lauf2()
schiebe=schiebe+1
for i=0 to 2
        bi(i).style.backgroundpositionx=schiebe-i*bx
next
settimeout "lauf2()",32
end sub
</script>
```

Als erstes ändern wir die Zuordnung der Variablen.

```
bi=array(teil1,teil2,teil3)
```

Diesen Teil werden wir verschieben, nach der Definition des Popup.

Als nächstes fragen wir die neue Fenstergröße ab. Dazu ändere ich die Variablenzuordnung von **ay** und **ay.**

aus

```
ay=document.body.clientheight
bx=bld1.clientwidth/3
```

wird

```
ax=screen.availWidth
ay=screen.availHeight
```

Die nächsten 3 Anweisungen können wir direkt so stehen lassen

```
bx=bld1.clientwidth/3
by=bld1.clientHeight
schiebe=0
```

Nun folgt die Definition des Popup:

```
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.filter="progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,
endcolorstr=#ffFF9650,gradienttype=0)"
```

Den body des Popup hinterlege ich mit einem Gradientfilter.

Jetzt erst definiere ich mein Array mit den drei DIV's.
Sie sind Bestandteil des POPUP und müssen jetzt über die Variable, die in **Set xxxx=poBody.**all definiert wurde, angesprochen werden. In unseren Fall ist das die Variable **poall**.
Also definiere ich mein Array:

```
bi=array(poall.teil1,poall.teil2,poall.teil3)
```

Wenn im Dokument der Button 'START' gedrückt wird, benötige ich die Routine: start(), wie wir es in der HTML-Anweisung, **<button id=st onClick=start()>Start</button>,** angegeben hatten.
Hier also die Routine:

```
sub start()
musik.src=musik.src
musik.volume=0
po.show 0,0,ax,ay
init()
end sub
```

Hier kommt nun unsere Musik ins Spiel, sie soll ja starten, sobald das Popup gestartet wird.
Die Musik wird ja gestartet, beim Laden das Dokuments, über den **<bgsound>.**
Dort haben wir aber die Lautstärke auf **-10000** gestellt, es ist also nichts zu hören.
Würden wir nun die Lautstärke einfach hoch setzen, dann wäre das Musikstück schon eine Weile gelaufen und wir würden es nicht von Anfang an hören.
Deshalb ein kleiner Trick:
Unserem **<bgsound>** hatten wir ja die **id=musik** gegeben. Deshalb:
musik.src = musik.src.
Das neue Musikstück ist das, das in *<bgsound>* geladen wurde und es beginnt wieder von Vorne. Danach die Lautstärke hoch**, musik.volume=0,** volle Lautstärke, und fertig.

Als nächstes öffne ich das Popup am Monitor, wie in den vorherigen Lektionen gezeigt, über **po.show links, oben, breite, höhe**.

In der nächsten Zeile steht der Aufruf **init().**
Diese Subroutine werde ich gleich einfügen, sie enthält die schon bekannte Schleife zum Aufbau der 3 DIV's

```
sub init()
for i=0 to 2
        bi(i).style.height=by
        bi(i).style.width=bx
        bi(i).style.backgroundImage="url(" & bld1.src &")"
        bi(i).style.backgroundposition=-bx*i
        bi(i).style.top=(ay-by)/2
next
poall.teil2.style.left=(ax-bx)/2
settimeout "lauf()",3000
end sub
```

Hier steht nichts Neues, ausser das ich den Programmblock in eine eigen Routine gelegt habe, hier eben init().

Ich zeige Ihnen nun den ganzen Programmcode und erkläre anschließend noch ein paar Zeilen.

```
<script language=vbscript>
ax=screen.availWidth
ay=screen.availHeight
bx=bld1.clientwidth/3
by=bld1.clientHeight
schiebe=0
set po=window.createPopup()
set poBody=po.document.body
set poall=poBody.all
PoBody.innerHTML=popup.innerHTML
PoBody.style.filter="progid:DXImageTransform.Microsoft.Gradient(startcolorstr=#ff960000,
                        endcolorstr=#ffFF9650,gradienttype=0)"
bi=array(poall.teil1,poall.teil2,poall.teil3)
sub start()
musik.src=musik.src
musik.volume=0
po.show 0,0,ax,ay
init()
end sub
sub init()
for i=0 to 2
        bi(i).style.height=by
        bi(i).style.width=bx
        bi(i).style.backgroundImage="url(" & bld1.src &")"
        bi(i).style.backgroundposition=-bx*i
        bi(i).style.top=(ay-by)/2
next
poall.teil2.style.left=(ax-bx)/2
settimeout "lauf()",3000
end sub
sub lauf()
schiebe=schiebe+1
if schiebe+50+bx >(ax-bx)/2 then
        schiebe=0
        settimeout "lauf2()",2000
        exit sub
end if
poall.teil1.style.left=50+schiebe
poall.teil3.style.right=50+schiebe
s=settimeout("lauf()",32)
end sub
sub lauf2()
schiebe=schiebe+1
for i=0 to 2
        bi(i).style.backgroundpositionx=schiebe-i*bx
next
settimeout "lauf2()",32
end sub
sub byebye()
parent.po.hide()
musik.volume=-10000
tx.innerHTML="Nun kann hier ein<br>Abschiedstext stehen."
end sub
</script>
```

Das Wichtigste:
Sie sehen, überall dort wo wir ohne popup direkt die ID verwendet haben,
z.B. **teil1.style.left**, müssen wir nun ein **poall.style.left** vorsetzen.

Eine Routine kam noch hinzu:

```
sub byebye()
parent.po.hide()
musik.volume=-10000
tx.innerHTML="Nun kann hier ein<br>Abschiedstext stehen."
end sub
```

Erklärung:
parent.po.hide() schließt unser Popup.

Musik.volume=-10000 schaltet den <bgsound> wieder auf leise.

tx.innerHTML lässt uns einen neuen Inhalt für den DIV id=tx erstellen.
Dabei können alle HTML-Anweisungen benutzt werde. Die kompletten Anweisungen müssen in Hochkommas stehen.

So das war's auch schon.
In der nächsten Lektion zeige ich Ihnen ein paar Tricks mit DIV's, in denen Text steht.

Lektion 5: Textboxen und deren Formatierungen.

In dieser Lektion zeige ich Ihnen einige Möglichkeiten zur Darstellung einer Textbox.
Einmal mit Bildlaufleisten, dabei werden wir die Bildlaufleisten auch verkleinert darstellen.
Dann schauen wir wie der Text automatisch bis zum Ende Scrollen kann.
Zusätzlich werden wir noch einige Verschönerungen einbauen.
Neben dem Scriptteil wird in dieser Lektion das Hauptaugenmerk auf CSS gerichtet sein.
Doch Schluss mit vielen Worten, lassen wir Taten sprechen.

Laden Sie die Datei: Lektion5.html aus dem Order Lektion 5.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Neue Seite 1</title>
<style type=text/css>

</style>
</head>
<body background="h18.jpg" scroll=no>
<p id=titel>Spielereien mit Textboxen</p>
<div id=thinter></div>
<div id=le></div>
<div id=tx>
        <p id=inh><font color=yellow>
        Erinnerungen<br><br></font>
        Erinnerungen sind wie Wassertropfen<br>
        ein Augenblick kaum eine Ewigkeit.<br><br>
        Ich schwelge in meinen Erinnerungen,<br>
        träume mich zurück in schöne Zeiten,<br>finde viele Kostbarkeiten.<br><br>
        Und Dankbarkeit, dass ich das alles erleben durfte,<br>
        schenkt mir Momente einer stillen Freude.<br><br>
        Man trägt das vergangene Schöne <br>nicht wie einen Stachel,<br>
        sondern wie ein kostbares Geschenk in sich.<br><br>
        Und blickt hoffnungsvoll in eine schöne Zukunft,<br>
        die schon bald Erinnerungen birgt.<br><br>
        <font color="#ff5500">Autor Unbekannt</font></p>
</div>
</body>
<script language=vbscript>

</script>
</html>
```

Erklärung:
In diesem Dokument haben wir eine Hintergrunggrafik, **h18.jpg**, das Scrollen des Hintergrunds ist gesperrt.
Es folgt ein Paragraph mit der **ID=Titel.**
Es folgen zwei DIV's, **ID=hinter** und **ID=le**.
Danach ein DIV mit **ID=tx,** der unseren Text enthält.

Sehen Sie sich das Dokument nun in der Vorschau an.

Ich möchte nun beginnen, diese Objekte über CSS zu formatieren.

Fangen wir mit dem Paragraph **ID=titel** an.

```
<style type=text/css>
#titel    {       position:absolute;
                  color:white;
                  font-size:28pt;
                  font-variant:small-caps;
                  font-style:italic;
                  top:20px;
                  filter:shadow(color=black,direction=135) }
</style>
```

Erklärung:
Die Position des Paragraph ist absolut.
Die Schriftfarbe soll weiß sein.
Die Schriftgröße ist 28pt.
Die Schrift wird als Kapitälchen dargestellt und ist Kursiv. **font-variant** und **font-style**.
Der Text ist 20Pixel vom oberen Fensterrand entfernt.
Es wird ein Schattenfilter mit der Schattenfarbe schwarz und dem Schattenwiinkel von 135° verwendet.

Fügen Sie die roten Zeilen Ihrem Dokument hinzu und betrachten Sie das Ergebnis wieder in der Vorschau.

Als nächstes werden wir den Text und das DIV ID=tx formatieren.

```
#tx       {       position:absolute;
                  border:white 4 groove;
                  padding:10px;
                  top:100;
                  width:400;
                  height:300;
                  color:white;
                  font-size:18pt;
                  text-align:center;
                  background-color:#8C8A5A }
```

Erklärung:
Die Position absolut, eine weiße Umrandung vom Typ Groove. Der Text hat eine Abstand von 10 Pixel zum Rand der Box.
Die Textbox ist 100Pixel von Oben entfernt, hat eine Breite von 400 Pixel und eine Höhe von 300 Pixel. Die Schriftfarbe ist weiß, Schriftgröße 18pt, ^wird zentriert dargestellt.
Zum Schluss wird noch eine Hintergrundfarbe für die Box festgelegt.

Wenn Sie sich nun das Dokument wieder in der Vorschau oder im Browser betrachten, werden ein paar Sachen negativ auffallen.

1. Wir haben eine Höhe von 300 Pixel angegeben, doch die Box ist höher.
2. Je nach Bildschirmauflösung werden Sie den Text nicht ganz lesen können, da wir im **<body>** die Scrollleiste ausgeschalten haben, können wir auch nicht runterscrollen.

Das alles hängt zusammen mit der Scrollfähigkeit unserer Textbox, **DIV ID=tx**.
Um das Problem zu lösen, fügen Sie im CSS-Teil für **#tx** folgende Zeile hinzu:

overflow:auto

Nun hat die Textbox die gewünschte Höhe.
Auf das overflow komme ich gleich noch mal zu sprechen, erst einmal möchte ich den Titel wie auch die Textbox zentrieren.
Das geschieht wie immer bei absolut positionierten Objekte im Scriptteil.

```
<script language=vbscript>
sub window_onLoad()
        tx.style.left=(document.body.clientWidth-tx.clientWidth)/2
        titel.style.left=(document.body.clientwidth-titel.clientWidth)/2
end sub
</script>
```

Damit sind diese beiden Objekte zentriert.

Nun wollen wir uns aber das overflow genauer betrachten.
Es gibt diese Möglichkeiten:

overflow:visible	Ist die Grundeinstellung. Dabei bestimmt der Inhalt die Größe des Objekts. Es werden keine Scrollleisten angezeigt.
Overflow:hidden	Die Größe des Objekts wird eingehalten. Überstehende Inhalte werden abgeschnitten. Keine Scrollleisten.
Overflow:scroll	Die Größe des Objekts wird eingehalten, Sowohl vertikale als auch horizontale Scrollleisten werden eingeblendet.
Overflow:auto	Die Größe des Objekts wird eingehalten. Beim Überschreiten werden je nach Bedarf eine horizontale, vertikale oder beide Scrollleisten eingeblendet.

Testen Sie alle Möglichkeiten und sehen Sie sich die Ergebnisse im Browser an.

Zum genauen Positionieren gibt es noch ein paar Überlegungen.
Wenn Sie genau hinsehen und unsere Textbox mit dem Titel vergleichen, werden Sie sehen, dass die Textbox nicht 100% in der Mitte ist.

Stellen Sie sicher das Sie **overflow:auto** eingegeben haben.
Geben sie zum nun Testen in der Routine **window_onLoad()** als letzte Zeile folgendes ein:

msgbox tx.clientwidth

Sie werden sehen, es kommt als Breite nur 375px als Ergebnis.
Wo sind die restlichen 25px?
Nun, da ist erst mal unser border, den wir mit 4px angegeben haben.
Der Rand ist sowohl links wie auch rechts, also insgesamt 8px.
Fehlen immer noch 17px. Diese werden durch die Scrollleiste verbraucht.
Das heißt, wenn wir unsere Textbox genau zentrieren wollen, müssen diese beide Werte in die Berechnung eingezogen werden.

Ergänzen Sie also folgende Zeile:
tx.style.left=(document.body.clientWidth-tx.clientWidth-25)/2

Nachdem Sie alles getestet haben, können Sie die Zeile msgbox=tx.clientwidth wieder löschen.

Ich möchte nun zur Verschönerung einen 200px breiten Border auf der linken Seite hinzufügen, den wir später animieren werden.

Dazu seht im HTML-Teil schon ein DIV bereit, er hat die ID=le.

Geben Sie nun für diesen DIV die CSS-Formatierungen ein.

```
#le             {       position:absolute;
                        left:0;
                        top:0;
                        width:200;
                        border-right:#8C8A5A 4 groove;
                        filter:alpha(opacity=80,style=0)   }
```

Erklärung:
Die Position wieder absolut.
Vom linken Rand 0px entfernt, von oben 0pc entfernt,
Die Breite soll 200px betragen.
Die rechte Seite soll einen Rand bekommen, in der Farbe: #8C8A5A
Zugleich bekommt er noch einen Filter, damit er transparent wird.

In der Vorschau werden Sie noch nicht viel sehen, da der DIV noch keine Höhe und keinen Inhalt hat.

Als Inhalt möchte ich eine Hintergrund-Kachelung.
Dazu müssen wir zuerst das Bild für den Hintergrund im HTML-Teil einfügen und dann programmtechnisch dem DIV zuweisen.

Fügen Sie im HTML-Teil nach <body> folgendes ein:

```
<img src=h19.jpg id=hg>
```

Und als Format im CSS-Teil:

```
#hg     {       position:absolute;
                left:-3000  }
```

Im Scriptteil fügen wir in der Routine windows_onLoad() folgendes hinzu:

```
le.style.height=document.body.clientwidth
le.style.backgroundImage="url(" & hg.src & ")"
```

Die gewohnte Syntax, die wir jetzt schon oft benutzt haben.

Wenn Sie das Dokument nun im Browser betrachten, stimmt unsere Ausrichtung des Titels und der Textbox nicht mehr.

Dokument in der Browservorschau bis jetzt:

Wir müssen also unsere Ausrichtung wieder anpassen.
Es muss die DIV-Breite und! Der Rand des Div's berücksichtigt werden.

Wenn wir das Ergebnis aus le.clientWidth auslesen, erhalten wir 196px.
Der Rand ist 4px, also insgesamt 200.

Also müssen wir zwei Zeilen anpassen:

```
tx.style.left=(200+document.body.clientWidth-tx.clientWidth-25)/2
titel.style.left=(200+document.body.clientwidth-titel.clientWidth)/2
```

Jetzt stimmt die Ausrichtung wieder.

Sie sehen jetzt, dass ein wirklich genaues Arbeiten gar nicht so einfach ist.

Nun möchte ich Ihnen weitere Möglichkeiten zur Gestaltung der Textbox zeigen.

Fangen wir an mir der Scrollleiste.

Ergänzen Sie im CSS-Teil bei #tx mit folgenden Zeilen:

```
#tx     {       position:absolute;
                border:white 4 groove;
                padding:10px;
                top:100;
                width:400px;
                height:300px;
                color:white;
                font-size:18pt;
                text-align:center;
                background-color:#8C8A5A;
                overflow:auto;
                scrollbar-base-color:#ffffff;
                scrollbar-3d-light-color:#8C8A5A;
                scrollbar-arrow-color:#ffffff;
                scrollbar-darkshadow-color:#8C8A5A;
                scrollbar-face-color:#8C8A5A;
                scrollbar-highlight-color:#8C8A5A;
                scrollbar-shadow-color:#000000;
                scrollbar-track-color:# #635D3E;
                filter:progid:DXImageTransform.Microsoft.alpha(opacity=90,style=0) }
```

Die Scrollbar färben wir ein und die gesamte Textbox machen wir etwas transparent.

Als nächstes ein Trick, wie Sie die Scrollbar scheinbar verkleinern.

Fügen sie imCSS-Teil bei #tx, nach der letzten Zeile folgendes ein:

Zoom=50%

In der Vorschau sehen Sie, dass die Box wie auch die Schrift kleiner wurden.
Jetzt müssen wir ein paar Parameter ändern:

Im CSS-Teil bei #tx:

Aus	**width:400**	wird	**width:800**
Aus	**height:300**	wird	**height:600**
Aus	**font-size:18pt**	wird	**font-size:36pt**

Sie sehen, die Scrollleiste wird scheinbar kleiner.
Aber auch unsere Textbox hat sich verschoben.

Das müssen wir wieder korrigieren.

Aus:
tx.style.left=(200+document.body.clientWidth-tx.clientWidth-25)/2
wird
tx.style.left=(200+document.body.clientWidth-tx.clientWidth/2-12)/2

Als nächstes setzen wir wieder den Zustand vor zoom=50% her.

Löschen Sie die Zeile und ändern Sie die CSS-Anweisungen wie auch die Scriptanweisung wieder in den Zustand wie vorher.

Ich möchte Ihnen nun zeigen, wie Sie scheinbar ganz ohne Scrollleiste auskommen.

Ersetzen Sie die Zeilen im CSS-Teil:

Alt:

```
scrollbar-base-color:#ffffff;
scrollbar-3d-light-color:#8C8A5A;
scrollbar-arrow-color:#ffffff;
scrollbar-darkshadow-color:#8C8A5A;
scrollbar-face-color:#8C8A5A;
scrollbar-highlight-color:#8C8A5A;
scrollbar-shadow-color:#000000;
scrollbar-track-color:#635D3E;
```

Neu:

```
scrollbar-base-color:#8C8A5A;
scrollbar-3d-light-color:#8C8A5A;
scrollbar-arrow-color:#8C8A5A;
scrollbar-darkshadow-color:#8C8A5A;
scrollbar-face-color:#8C8A5A;
scrollbar-highlight-color:#8C8A5A;
scrollbar-shadow-color:#8C8A5A;
scrollbar-track-color:#8C8A5A;
```

Dadurch das die Scrollleiste mit allen Teilelementen die selbe Farbe bekommt wie der Hintergrund der Textbox, verschwindet sie scheinbar.
Hat die keinen eigenen Hintergrund, dann können Sie die Leistenelemente z.B. auf schwarz setzen und den Chromafilter darüber legen.

Das die Scrollleiste noch funktioniert, können Sie folgendermaßen testen:
Klicken Sie im Browser oder in der Vorschau mit der Maus in die Textbox und bewegen Sie das Mausrad. Dadurch können Sie den Text verschieben.

Nun wollen wir die Sache noch etwas automatisieren.
Nachdem unser Dokument geladen ist, möchte ich den Inhalt unser Textbox automatisch nach oben scrollen, bis wir am Ende des Textes angekommen sind.

Dazu ein paar Grundlagen.
Es gibt 3 Befehle, die uns nützlich sein können.

1. Objekt.scrollTop	Damit kann ich den Text scrollen, außerdem kann der Wert ausgelesen werden, um festzustellen, wie weit mein Text schon gescrollt ist.
2. Objekt.scrollHeight	Damit kann die gesamte Höhe des Inhalts unserer Textbox ausgelesen werden.
3. Objekt.clientheight+Border	damit bestimme ich die Höhe des sichtbaren Teil.

Da wir die Höhe festgelegt habe, ist die Höhe 300px.

Wir erzeugen also eine Scriptroutine, die uns nach den Laden das Scrollen automatisch ausführt.

Als erstes definiere ich eine Variable, ich nenne sie mal x, die uns den Scrollvorgang hochzählt. Fügen Sie zwischen **<script language.....>** und **sub windows_onLoad()** folgendes ein:

```
x=0
settimeout "lauf1()",3000
```

Danach erstellen wir eine Routine, ich nenne sie lauf1().

```
sub lauf1()
        x=x+1
        if x>tx.scrollHeight-300 then exit sub
        tx.scrollTop=x
        settimeout "lauf1()",64
end sub
```

Wenn alles richtig ist, sollte jetzt der Text nach drei Sekunden so lange nach oben scrollen, bis alles zu lesen war.

Nun kommen eigentlich nur noch ein paar Verschönerungen.
ASls erstes lege ich eine weiteres Div über unsere Textbox. Das DIV ist im HTML-Teil schon definiert, es hat die ID=thinter. Wir müssen das DIV noch formatieren.

Fügen Sie dazu im CSS-Teil folgendes ein:

```
#thinter        {       position:absolute;
                        border:white 4 groove;
                        top:90;
                        width:420;
                        height:320;
                        overflow:hidden  }
```

Da unser DIV um 20px breiter und um 20px höher ist, muss ich die Position um 10px höher legen als bei unserer Textbox, wenn diese genau in der Mitte sein soll.

Als Hintergrund möchte ich das Bild: h20.jpg verwenden.

Fügen Sie im HTML-Teil folgendes hinzu:

```
<img src=h20.jpg id=hg2 >
```

und im CSS-Teil:

```
#hg2   {        position:absolute;
                left:-3000 }
```

Jetzt müssen wir den Hintergrund noch einfügen und das DIV positionieren.

Das machen wir in der Routine **sub window_onLoad()**

Fügen Sie dort folgende Zeilen ein:

```
thinter.style.left=(200+document.body.clientwidth-thinter.clientWidth-8)/2
thinter.style.backgroundImage="url(" & hg2.src & ")"
```

Versuchen Sie nun selbst, die Positionierung zu verstehen.

Diesen Hintergrund möchte ich nun ebenfalls Scrollen.
Dazu definiere ich wieder ein Variable, diesmal nenne ich sie y.
Danach eine Routine, die ich lauf2() nenne.

Fügen Sie also ein:

```
y=0
lauf2()

sub lauf2()
        y=y+1
        thinter.style.backgroundPositionX=y
        settimeout "lauf2()",32.
end sub
```

Nun haben wir einen animierten Rand um unser Textfenster.

Als letztes wollen wir noch unseren linken Rand animieren und einen Sound einbinden.
Fügen Sie nach <body....> ein:

```
<bgsound src="l3.wav" loop=4 volume=0>
```

Um den Rand zu animieren:

```
z=0
lauf3()

sub lauf3()
        z=z+1
        le.style.backgroundPositionY=z
        settimeout "lauf()",32
end sub
```

Jetzt fehlt uns nur noch eine Routine: bei Größenänderung des Dokuments soll dieses neu aufgebaut werden. Fügen Sie folgende Routine hinzu:

```
sub window_onResize()
location.reload
end sub
```

Das wars auch schon. In der nächsten Lektion möchte ich Ihnen den Matrixfilter vorstellen.