

In diesem Kapitel:

- Einführung in die Erstellung dynamischer Menüs
- Das obere Textmenü erstellen
- Das grafische Menü erstellen
- Einen Klickpfad realisieren

Menüs erstellen

Im vorherigen Kapitel haben Sie die Grundlagen der dynamischen Bilderzeugung und Bildmanipulation kennen gelernt und Ihre generellen TypoScript-Kenntnisse vertieft. In diesem Kapitel erfahren Sie, wie Sie dynamische Menüs erstellen bzw. beschreiben können. Dabei werden Sie die Kenntnisse der vorherigen Kapitel anwenden und neue Objekte, Eigenschaften und Funktionen kennen lernen.

Einführung in die Erstellung dynamischer Menüs

TYPO3 liefert von Haus aus eine automatische Grafikerzeugung mit (siehe Kapitel 6, *Grafiken mit TypoScript erstellen*). Dieses Feature kann man gerade bei den grafischen Menüs gut nutzen, da hier nur eine grafische Vorlage erstellt werden muss. Die »Massenproduktion« von Grafiken mit MouseOver-Effekten etc. übernimmt TYPO3 mehr oder weniger automatisch.

Warum aber *mehr oder weniger* automatisch? Da TYPO3 ein sehr mächtiges Content Management System ist, steht Ihnen auch und gerade bei den Menüs eine Vielzahl von Möglichkeiten zur Verfügung. Um diese nutzen zu können, wird zur Beschreibung eines Menüs ebenfalls TypoScript verwendet. Mit TypoScript lassen sich beliebige Menüs erstellen, die das Verhalten und Aussehen haben, das Sie bevorzugen.

Unterschiedliche Arten von Menüs

Es gibt vier grundsätzlich unterschiedliche technische Arten von Menüs: Textmenüs, grafische Menüs, Layer-Menüs und JavaScript-Menüs. Die wohl einfachsten Menüs sind Textmenüs, da hier keine Grafiken erzeugt werden müssen. Die restlichen drei Menüarten sind etwas anspruchsvoller, aber keineswegs schwierig zu erstellen. Mit diesen vier Menüarten lassen sich herkömmliche Menüs, aber auch Klickpfade und Sitemaps erstellen.

Folgende Auflistung von Menüobjekten soll Ihnen einen Überblick darüber geben, welche häufig verwendeten Arten von Menüs in TYPO3 zur Verfügung stehen. Diese Menüobjekte können jedoch nicht direkt aufgerufen werden, sondern nur indirekt über ein übergeordnetes Objekt HMENU (HMENU = hierarchisches Menü), das im folgenden Abschnitt bei der Erstellung des Textmenüs vorgestellt wird.

TMENU

Mit dieser Menüart können textbasierte Menüs erstellt werden. Es ist das einfachste zu erstellende Menü und bietet in Kombination mit Stylesheets (CSS) viele Designmöglichkeiten. Häufig wird TMENU auch zur Erstellung von menüähnlichen Bereichen eingesetzt, wie z.B. für einen Klickpfad oder Site-maps.

GMENU

Das grafische Menü bietet bessere Gestaltungsmöglichkeiten als das textbasierte Menü. Hierbei wird aus jedem Menüelement eine Grafik erstellt, die mittels TypoScript beschrieben wird.

JSMENU

Mit einem JavaScript-Menü lässt sich ein Auswahlfeld erstellen, aus der der Webseiten-Besucher eine Seite auswählen kann, zu der er wechseln möchte. Ein häufiger Anwendungsfall ist hierbei die Erstellung eines (nicht dynamischen) Menüs mit den am häufigsten besuchten Seiten.

TMENU_LAYERS

Mit dieser Menüart kann ein textbasiertes Menü auf Basis von Layern erstellt werden. Hiermit lassen sich aufklappbare Menüs realisieren, die bei einem MouseOver über eine erste Menüebene automatisch Menüelemente einer zweiten Menüebene öffnen.

GMENU_LAYERS

Ähnlich wie TMENU_LAYERS, jedoch auf Grundlage des grafischen Menüs.

Mögliche Zustände von Menüelementen

Einzelne Menüelemente können unterschiedliche Zustände erhalten. Ein Menüelement bzw. Menü-Item ist ein einzelner Eintrag aus der Navigation, wie z.B. ein Textlink *Homepage*. Ein klassischer und direkt nachvollziehbarer Zustand eines Menüelements ist beispielsweise der MouseOver-Effekt, der in TYPO3 als Roll-Over-Effekt bezeichnet wird. Bei einem grafischen Menü kann z.B. angegeben werden, dass eine andere Grafik für den Zustand geladen werden soll, wenn die Maus über ein Menüelement bewegt wird.

TYPO3 bietet aber noch mehr Zustände als nur einen RollOver-Effekt. Dabei sind fast alle diese Zustände in den oben genannten Menüarten vorhanden, eine Ausnahme bildet das JavaScript-Menü, bei dem z.B. kein RollOver-Zustand möglich ist. Folgende Auflistung zeigt, welche Zustände für Menüelemente zur Verfügung stehen:

NO

Mit dem Zustand NO wird der *normale* Zustand eines Menüelements beschrieben. Die Angabe einer Beschreibung für diesen elementaren Zustand ist zwingend notwendig.

RO

RollOver bzw. MouseOver. Änderungen z.B. an der Schrift bzw. an der Grafik bei einem Darüberfahren mit der Maus können hier beschrieben werden. Der RO-Zustand findet insbesondere bei der Menüart *GMENU* bzw. *GMENU_LAYER* Anwendung.

CUR

Der Menüpunkt mit der aktuell geöffneten Seite erhält den Zustand CUR. Hierdurch kann dem Menüpunkt der geöffneten Seite ein anderes Aussehen gegeben werden, z.B. durch Hervorhebung.

ACT

Alle Menüpunkte, die die momentan geöffnete Seite als Unterseite haben, können mit dem Zustand ACT anders beschrieben werden. Auch die momentan geöffnete Seite gehört mit zu diesem Zustand, kann über CUR aber explizit angesprochen werden.

IFSUB

Wenn ein Menüeintrag mindestens eine Unterseite hat, sich also weitere Menüeinträge unterhalb eines Menüeintrags befinden, kann die Darstellung über den Zustand IFSUB anders beschrieben werden.

SPC

Mit SPC (engl. space, Platz) lässt sich das Aussehen eines Abstands beschreiben. Seiten können in der Navigation vom Seitentyp *Abstand* angelegt werden, um zwischen zwei Menüelementen einen bestimmten Freiraum zu erreichen. Ist eine Seite von diesem Typ, greift hier der Zustand SPC.

USR

Wenn für eine Seite eine Beschränkung auf Frontend-User festgelegt ist, greift der Zustand USR. Dieser Zustand kann nicht mit anderen Zuständen kombiniert werden und ist somit ein exklusiver Zustand.

IFSUBRO/IFSUBCUR/CURRO/ACTRO

Die einzelnen Zustände können teilweise auch miteinander kombiniert werden. So können Sie z.B. für ACT und RO jeweils eigene Beschreibungen vornehmen. Treffen auf einen Menüeintrag aber beide Zustände zu, hat also in diesem Beispiel die momentan geöffnete Seite Unterseiten, können Sie hier eine gesonderte Beschreibung für das Auftreten beider Zustände vornehmen.

Das obere Textmenü erstellen

Ist der Trailer fertig gestellt, können Sie sich nun dem ersten Menü im oberen Bereich unseres Praxisbeispiels widmen. Dieses Menü soll ein reines Textmenü werden, die einzelnen Menüelemente werden dynamisch der Seitenstruktur entnommen. Bevor allerdings die Beschreibung der Menüelemente losgeht und somit auch Überlegungen stattfinden, ob es sich um ein text- oder um ein grafisches Menü handeln soll, muss zunächst angegeben werden, woher das Menü überhaupt aufgebaut werden soll. Hierzu gibt es ein Objekt HMENU (hierarchisches Menü), in dem diese Eigenschaften gesetzt werden können.

Sprechen Sie, wie in Beispiel 7-1 zu sehen, zunächst mittels TypoScript den Marker `MENU_OBEN` an.

Beispiel 7-1: Den Platzhalter »MENU_OBEN« ansprechen

```
01 page = PAGE
02 page {
    [...]
16     10.marks {
        [...]
29         TRAILER.file {
            [...]
56     }
57
58     # Das Textmenü oben erstellen
59     MENU_OBEN = HMENU
60 }
61 }
```

In Zeile 47 wird auf dem Platzhalter `MENU_OBEN` eine Instanz des `HMENU`-Objekts gebildet. Ein Blick in das Frontend zeigt, dass der Marker bereits angesprochen ist, ein Menü allerdings noch nicht dargestellt wird.

Startpunkt des Menüs angeben

Sie müssen jetzt dem System noch mitteilen, von wo das Menü dargestellt werden soll. Hierfür gibt es die Eigenschaft `.special` des `HMENU`-Objekts. Um diese Eigenschaft nutzen zu können, muss zunächst bekannt sein, von welcher Seite aus

das Menü dargestellt wird. Sie wissen zwar, dass dies die Hilfsseite *Menü oben* ist – für TypoScript wird aber eine eindeutige ID (unique-ID = uid) des Seiten-Datensatzes benötigt.

Wenn Sie in der Baumdarstellung mit der Maus über das Icon der Seite *Menü oben* fahren, wird Ihnen als <Alt>-Tag die entsprechende Seiten-ID angezeigt: Die in Abbildung 7-1 angezeigte Seiten-ID lautet somit *uid=4*. Die ausgelesene Seiten-ID kann nun, wie in Beispiel 7-2 zu sehen, in einer *.special.value*-Eigenschaft angewendet werden.

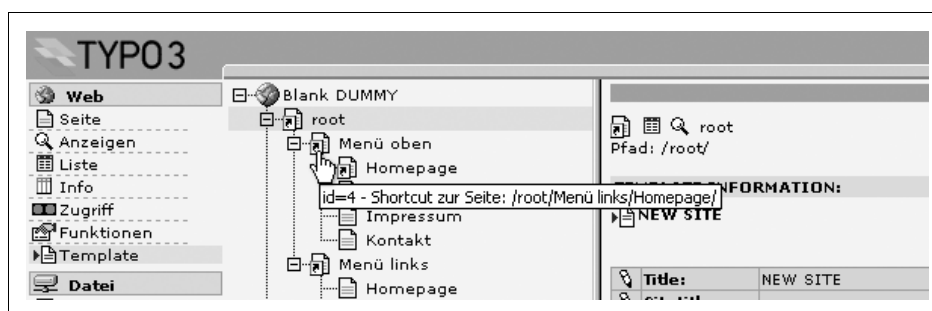


Abbildung 7-1: Seiten-ID auslesen



Die Seiten-ID wird mit großer Wahrscheinlichkeit in Ihrem Projekt vom hier gezeigten Beispiel abweichen. Wenn Sie die Seiten auf eine andere Art und Weise erstellt haben oder zwischendurch eine Seite angelegt und wieder gelöscht haben o.Ä., wird die ID für die Seite *Menü oben* vermutlich anders lauten.

Beispiel 7-2: Angeben, von welcher Hilfsseite aus das Menü dargestellt werden soll

```
[...]  
58 # Das Textmenü oben erstellen  
59 MENU_OBEN = HMENU  
60 MENU_OBEN {  
61     special = directory  
62     special.value = 4  
63 }  
[...]
```

In Zeile 61 geben Sie über die Eigenschaft *.special* an, dass es sich bei dem erzeugten Menü um ein *directory*-Menü handeln soll, also um ein klassisches Menü. Weitere Werte für Menü-Möglichkeiten, die mit *.special* realisiert werden können, erhalten Sie weiter unten in diesem Kapitel und in der TypoScript-Referenz im Anhang.

In Zeile 62 wird für das *special=directory*-Menü über die Eigenschaft *.special.value* ein Wert angegeben, der angibt, von wo aus das Menü dargestellt werden

soll. Hier sehen Sie den Wert 4, da die Hilfsseite *Menü oben* die eindeutige Seite-ID 4 hat.

Die Anzeige im Frontend wird noch kein Resultat liefern – der Marker sollte zwar nicht mehr angezeigt werden, ein Menü selbst wird aber auch noch nicht dargestellt.

Das Textmenü darstellen

Um das Menü nun im Frontend sichtbar zu machen, muss TYPO3 zumindest mitgeteilt werden, wie der normale Zustand der ersten Menüebene aussehen soll. Die einzelnen Navigationsebenen werden wieder mit Nummern beschrieben, dieses Mal jedoch nicht in Zehnerschritten zur Kennzeichnung einer Reihenfolge bzw. zur Angabe einer grafischen Ebene, sondern in Einerschritten zur Beschreibung der Navigationsebene.

Zur Erzeugung des oberen Textmenüs wird auch nur eine Ebene 1 benötigt, da für diese Navigation keine Unterseiten vorgesehen sind. Diese erste Ebene soll ein Textmenü sein, das in TYPO3 durch das Objekt TMENU erstellt wird. Die Beschreibung des Textmenüs erfolgt, wie in Beispiel 7-3 zu sehen, unter Angabe des Kennzeichners NO für den normalen Zustand.

Beispiel 7-3: Das Textmenü erzeugen

```
[...]  
58 # Das Textmenü oben erstellen  
59 MENU_OBEN = HMENU  
60 MENU_OBEN {  
61     special = directory  
62     special.value = 4  
63     1 = TMENU  
64     1.NO = 1  
65 }  
[...]
```

In Zeile 63 wird angegeben, dass auf der ersten Menüebene ein Textmenü (TMENU) angewendet werden soll. Somit stehen auf der ersten Ebene alle Eigenschaften von TMENU zur Verfügung. In Zeile 64 wird für die erste Menüebene der normale Zustand aktiviert.

Zustände müssen im Regelfall explizit mittels [Zustand] = 1 aktiviert werden. Eine Ausnahme hierzu bildet der Zustand NO, der nicht ausdrücklich aktiviert werden muss, machen Sie es aber dennoch, es wird Ihnen die Arbeit an späterer Stelle vereinfachen. Zur Durchführung des folgenden Beispiels muss der Zustand NO allerdings aktiviert werden, da die Ausführung ansonsten auf Grund einer fehlenden Beschreibung für den normalen Zustand NO verhindert werden würde.

Ein Blick in das Frontend sollte nun das in Abbildung 7-2 gezeigte Menü erscheinen lassen.



Abbildung 7-2: Das Menü wird für den angegebenen Platzhalter angezeigt

Stylesheets verwenden

Das Stylesheet greift bereits durch die Art und Weise, wie die Designvorlage angelegt wurde. Wenn Sie dennoch nachträglich eine CSS-Definition für die Menüelemente angeben möchten, können Sie in das `<A>`-Tag eingreifen und einen zusätzlichen Parameter aufnehmen. Da die Links von TYPO3 automatisch erstellt werden, stellt das TMENU-Objekt die Eigenschaft `.ATagParams` zur Verfügung, mit der Sie das `<A>`-Tag durch zusätzliche Angaben erweitern können:

```
page.10.MENU_OBEN.1.NO.ATagParams = class="links_weiss"
```

Menüeinträge voneinander trennen

Die Menüeinträge *kleben* zurzeit noch aneinander. Mit der Eigenschaft `.linkWrap` des TMENU-Objekts kann jeder einzelne Menüeintrag mit HTML-Code umhüllt werden, um hiermit ein Leerzeichen vor jedem Menüeintrag mit Hilfe des HTML-Codes ` ` zu erzwingen. Das bereits bekannte Wrap-Symbol `|`, das in Kapitel 5, *TypoScript in der Praxis*, vorgestellt wurde, bestimmt auch hier wieder die Position, an der ein Inhalt einfließen soll. In folgendem Beispiel wird an der Stelle des Wrap- bzw. Pipe-Symbols jeweils der Link eines Menüeintrags gesetzt.

Beispiel 7-4: Ein Leerzeichen zwischen jeweils zwei Menüelemente setzen

```
[...]  
58 # Das Textmenü oben erstellen  
59 MENU_OBEN = HMENU  
60 MENU_OBEN {  
61     special = directory  
62     special.value = 4  
63     1 = TMENU  
64     1.NO = 1  
65     1.NO.linkWrap = &nbsp;|  
66 }  
[...]
```

Striche zwischen den einzelnen Menüeinträgen einfügen

Das Ergebnis wird Sie vielleicht schon zufrieden stellen, die Vorgabe des Grafikers, die Sie in Abbildung 4-1 im Kapitel 4, *Das Praxisbeispiel vorbereiten*, sehen können, ist allerdings eine andere: Zwischen jedem Menüeintrag soll ein Strich erscheinen.

Um dieses zu erreichen, kann der Strich | mit in die `linkWrap`-Eigenschaft aufgenommen werden. Jedoch ist dieser Strich, im Folgenden als Pipe-Symbol bezeichnet, bereits für den Wrap belegt und kennzeichnet die Position, die *umhüllt* werden soll. Es bleibt nichts anderes übrig, als den ASCII-Code dieses Pipe-Symbols zu verwenden: `|`. Im Folgenden wird nun die Eigenschaft `.linkWrap` um diesen zusätzlichen Strich erweitert:

```
65      1.NO.linkWrap = &nbsp;|&nbsp;&#124;&nbsp;
```

Beim `linkWrap` wird nun zunächst ein Leerzeichen ausgegeben, dann folgt der Menüeintrag inklusive `<A>`-Tag, erneut ein Leerzeichen und zum Schluss das Pipe-Symbol. Dieser `linkWrap` wird für jeden einzelnen Menüeintrag wiederholt, was dazu führt, dass am Ende des Menüs, wie in Abbildung 7-3 zu sehen, immer genau ein Pipe-Symbol zu viel angezeigt wird.

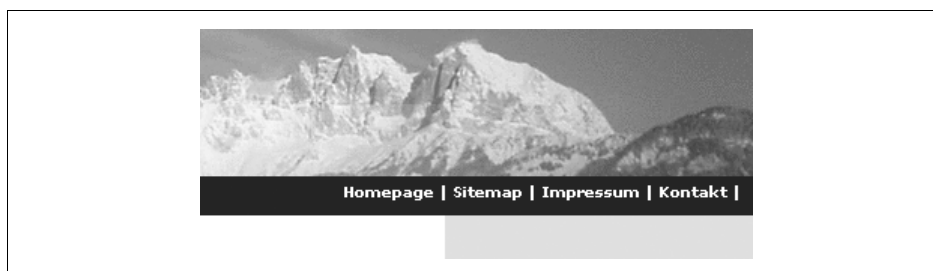


Abbildung 7-3: Ein Pipe-Symbol am Ende des Menüs ist überflüssig

optionSplit für Textmenüs mit Pipe-Symbol

Eine Lösung dieses Problems ist nur mittels einer so genannten `optionSplit`-Möglichkeit gegeben. Damit können diverse Eigenschaften und sogar Objekte, die in einer Schleife ausgeführt werden, aufgeteilt werden. Die Möglichkeiten der Aufteilung sind zwar beschränkt, für den praktischen Einsatz in der Regel jedoch ausreichend.

Bei der Gestaltung eines Textmenüs sollen beispielsweise folgende Punkte berücksichtigt werden: Der erste und der letzte Menüeintrag sollen sich von denen in der Mitte unterscheiden, dabei soll die Unterscheidung durch einen anderen Wrap vorgenommen werden. Das erste und das letzte Element sollen eine rote Textfarbe erhalten, die Menüelemente in der Mitte eine grüne Textfarbe.

Ein solches Vorhaben wird mit `OptionSplits` realisiert. Diese teilen eine Wertzuweisung durch die Eingabe der Zeichenkombination `|*` in Anfang, Mitte und Ende

auf. Die folgenden Beispiele dienen dem besseren Verständnis, sind aber nicht direkt ausführbar.

```
textfarbe = [Erstes Element] |*| [mitte] |*| [Letztes Element]
textfarbe = red |*| green |*| red
```

Die Unterteilung ist noch weiter möglich: Jeder Teilbereich in sich kann wieder aus einem ersten, zweiten etc. Element bestehen. Diese Unterteilung wird durch das Symbol || (zwei aufeinanderfolgende Pipe-Symbole) erreicht.

```
textfarbe = [Erstes Element]||[Zweites Element] |*| [mitte] |*| [Letztes Element]
textfarbe = red || blue |*| green |*| red
```

Dieser Code würde das erste Element mit einer roten Textfarbe, das zweite Element mit einer blauen, alle mittleren Elemente mit einer grünen und das letzte Element wieder mit einer roten Textfarbe erscheinen lassen.

Für die Realisierung des Praxisbeispiels müssen somit folgende Überlegungen angestellt werden:

- Für den ersten Menüeintrag gilt als linkWrap: ` | ||`
- Für alle Einträge in der Mitte gilt ebenfalls: ` | ||`
- Für das letzte Element werden das Pipe-Symbol sowie das letzte Leerzeichen nicht mehr benötigt: ` |`

Zusammengesetzt ergibt sich hieraus die folgende Kombination: am Anfang und in der Mitte jeweils ` | ||`, am Ende nur ein ` |`. Daraus folgt der (durchaus umfangreiche) linkWrap:

```
linkWrap = &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124;
```

In Beispiel 7-5 wird dieser linkWrap in das TypoScript-Template eingebunden. Ein Betrachten des Ergebnisses im Frontend sollte das in Abbildung 7-4 gezeigte Resultat liefern.

Beispiel 7-5: Pipe-Symbole im Menü mit optionSplit darstellen

```
[...]
58 # Das Textmenü oben erstellen
59 MENU_OBEN = HMENU
60 MENU_OBEN {
61     special = directory
62     special.value = 4
63     1 = TMENU
64     1.NO = 1
65     1.NO.linkWrap = &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124; |*| &nbsp;|&nbsp;|&#124;
66 }
[...]
```



Abbildung 7-4: Die Pipe-Symbole werden jetzt nur noch zwischen den Menüelementen angezeigt

Das grafische Menü erstellen

Auf der linken Seite soll nun ein grafisches Menü erscheinen, das umzusetzende Endergebnis können Sie der Abbildung 7-5 entnehmen. Die Menüeinträge sollen jeweils durch eine weiße Linie voneinander getrennt werden. Bei einem MouseOver wie auch bei aktiven Seiten soll sich zudem die Hintergrundfarbe ändern.



Abbildung 7-5: Das umzusetzende grafische Menü

Zunächst wird der Marker `MENU_LINKS` mit dem Objekt `HMENU` angesprochen. Ähnlich wie bei `TMENU` müssen Sie wieder der `.special`-Eigenschaft den Wert `directory` zuweisen, um anzugeben, dass ein *normales* Menü ausgehend von einer bestimmten Seite erstellt werden soll. Für die Eigenschaft `.special.value` müssen Sie die Seiten-ID der Seite *Menü links* angeben. In Beispiel 7-6 wird hierfür die Seiten-ID 3 angenommen, in Ihrem `TYPO3`-Projekt werden Sie voraussichtlich eine andere Seiten-ID für die Hilfsseite *Menü links* sehen. Bei dem Betrachten der Seite im Frontend werden Sie noch kein Menü erkennen können – der Marker selbst sollte aber auch nicht mehr dargestellt werden.

Beispiel 7-6: Den Platzhalter MENU_LINKS ansprechen

```
01 page = PAGE
02 page {
    [...]
16     10.marks {
        [...]
58         # Das Textmenü oben erstellen
59         MENU_OBEN = HMENU
60         MENU_OBEN {
61             special = directory
62             special.value = 4
63             1 = TMENU
64             1.NO = 1
65             1.NO.linkWrap = &nbsp;|&nbsp;|&nbsp;|*|&nbsp;|&nbsp;|&nbsp;|*|&nbsp;|
66         }
67
68         # Das grafische Menü erstellen
69         MENU_LINKS = HMENU
70         MENU_LINKS {
71             special = directory
72             special.value = 3
73         }
74     }
75 }
```

Grafische Menüeinträge erzeugen lassen

Das Objekt GMENU arbeitet ähnlich wie TMENU, hat die Eigenschaften zur Beschreibung von Zuständen allerdings vom GIFBUILDER-Objekt geerbt, das Sie bereits im vorherigen Kapitel kennen gelernt haben.

Jeder Menüeintrag wird nun mit einer blauen Hintergrundfarbe versehen. Die erste Menüebene soll eine grafische Ebene sein. Für den normalen Zustand wird eine Grafik erzeugt, die eine Breite von 178 Pixeln und eine Höhe von 24 Pixeln hat. Die Hintergrundfarbe eines Menüelements soll im normalen Zustand blau sein. Zur Realisierung wird das TypoScript-Template um entsprechende Angaben erweitert, zu sehen in Beispiel 7-7.

Beispiel 7-7: Breite, Höhe und Hintergrundfarbe von Elementen der ersten Menüebene

```
[...]
68 # Das grafische Menü erstellen
69 MENU_LINKS = HMENU
70 MENU_LINKS {
71     special = directory
72     special.value = 3
73     1 = GMENU
74     1.NO = 1
75     1.NO {
76         XY = 178, 24
```

Beispiel 7-7: Breite, Höhe und Hintergrundfarbe von Elementen der ersten Menüebene (Fortsetzung)

```
77         backColor = #364497
78     }
79 }
[...]
```

Beim Betrachten des Ergebnisses im Frontend können Sie erkennen, dass sich bereits etwas getan hat. Zwar sind die einzelnen Menüpunkte nicht direkt sichtbar, wenn Sie mit der Maus allerdings über die Fläche fahren, an der sich der Platzhalter für das linke Menü befand, so werden Sie einen <Alt>-Tag mit dem Titel einer Seite aus dem Menü erhalten. Um es kurz zu machen: Die Menüeinträge sind bereits vorhanden, allerdings nicht direkt sichtbar. Die Hintergrundfarbe der Menüelemente unterscheidet sich noch nicht von der Hintergrundfarbe der Tabelle, außerdem befindet sich noch kein Text auf den einzelnen Menüelementen.

Im folgenden Beispiel soll nun eine grafische Textebene auf den blauen Hintergrund gerendert werden. Hierzu wird das im vorherigen Kapitel Erlernte auch beim grafischen Menü angewendet. In Beispiel 7-8 wird Ebene 10 als grafische Textebene angelegt. Auf dieser grafischen Textebene wird dynamisch der Seitentitel der jeweiligen Seite gerendert.

Beispiel 7-8: Text auf die einzelnen Menüpunkte rendern

```
[...]
68 # Das grafische Menü erstellen
69 MENU_LINKS = HMENU
70 MENU_LINKS {
71     special = directory
72     special.value = 3
73     1 = GMENU
74     1.NO = 1
75     1.NO {
76         XY = 178, 24
77         backColor = #364497
78     }
79     # Text auf den Menüpunkt rendern
80     10 = TEXT
81     10.text.field = title
82     10.fontColor = #FFFFFF
83     10.fontFile = fileadmin/fonts/verdana.ttf
84     10.fontSize = 12
85     10.niceText = 1
86     10.offset = 14, 16
87 }
88 }
[...]
```

Beim Betrachten des Ergebnisses im Frontend werden die Menüeinträge bereits richtig angezeigt, wie in Abbildung 7-6 zu sehen. Der Menüpunkt *Informationen über das Snowboardgebiet* wird im Moment noch nicht vollständig angezeigt, da der verwendete Text für eine grafische Zeile zu lang ist. In einem späteren Abschnitt dieses Kapitels werden Sie erfahren, wie Sie einen mehrzeiligen grafischen Text realisieren können.

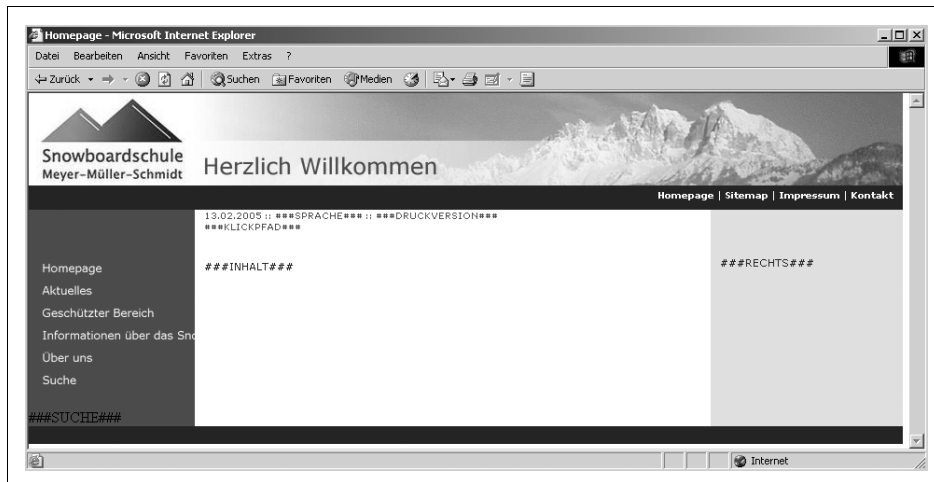


Abbildung 7-6: Die Menüpunkte auf der linken Seite werden mit grafischem Text dargestellt

Eine weiße Linie erzeugen

In der vom Grafiker gelieferten Vorlage werden die Menüelemente durch eine weiße Linie voneinander getrennt. Zur Realisierung gibt es gleich zwei Möglichkeiten:

- Mit einem Wrap: Nach jedem Menüelement wird eine weiße Linie mittels HTML-Code *eingewrappt*. In TypoScript könnte dieses zum Beispiel so lauten:

```
wrap = |<br>
```
- Die Linie wird direkt mit in die Grafik aufgenommen.

Anhand der zweiten Möglichkeit soll nun die weiße Linie mit in die Grafik eingearbeitet werden. Hierzu kann beispielsweise auf Ebene 20 eine vorhandene weiße Linie mit Hilfe des IMAGE-Objekts hineingeladen werden. Eine solche Linie ist aber nicht vorhanden, daher wird Sie in Beispiel 7-9 mit dem GIFBUILDER-Objekt dynamisch erstellt.

Beispiel 7-9: Eine weiße Linie in die Grafik aufnehmen

```
[...]
79 # Text auf den Menüpunkt rendern
80 10 = TEXT
81 10.text.field = title
```

Beispiel 7-9: Eine weiße Linie in die Grafik aufnehmen (Fortsetzung)

```
82  10.fontColor = #FFFFFF
83  10.fontFile = fileadmin/fonts/verdana.ttf
84  10.fontSize = 12
85  10.niceText = 1
86  10.offset = 14, 16
87
88  # Eine weisse Linie erzeugen
89  20 = IMAGE
90  20.file = GIFBUILDER
91  20.file {
92      XY = 178,1
93      backColor = #FFFFFF
94  }
95  20.offset = 0, 23
[...]
```

In Zeile 89 wird eine weitere Ebene 20 als Instanz des IMAGE-Objekts angelegt. Die hier erzeugte Grafik liegt somit oberhalb der Ebene 10 (Textebene). Zeile 90 gibt an, dass die Grafik dynamisch mit GIFBUILDER erzeugt werden soll mit einer Breite von 178 Pixeln und einer Höhe von einem Pixel. Die Hintergrundfarbe soll Weiß sein. Ebene 20 mit der erzeugten Linie wird in Zeile 95 um 0 Pixel nach rechts und 29 Pixel nach unten verschoben und befindet sich nun am unteren Rand der gesamten Grafik.

Eine zusätzliche Linie mittels wrap aufnehmen

Beim Betrachten des Ergebnisses im Frontend werden Sie bemerken, dass die Linien zwar am unteren Rand angezeigt werden, die weiße Linie oberhalb des ersten Menüpunkts jedoch fehlt, wie in der Anforderung in Abbildung 7-5 zu sehen. Sie haben nun die Möglichkeit, diese obere weiße Linie statisch mit in die Designvorlage aufzunehmen oder das gesamte Menü mit einer solchen weißen Linie zu *umhüllen*.

Letztere Variante wird in Beispiel 7-10 vorgestellt. Hierzu wird der Platzhalter MENU_LINKS, der eine Instanz des HMENU-Objekts ist, um eine wrap-Funktion erweitert, in der die weiße Linie statisch aufgenommen wird. Eine Grafik mit der Bezeichnung *linie_weiss.gif* haben Sie in Kapitel 4, *Das Praxisbeispiel vorbereiten*, bereits im Ordner *fileadmin/images/* abgelegt. Bei einem Betrachten des Ergebnisses im Frontend sollten Sie das in Abbildung 7-7 gezeigte Resultat erhalten.

Beispiel 7-10: Eine zusätzliche weiße Linie mittels wrap-Funktion aufnehmen

```
[...]
68  # Das grafische Menü erstellen
69  MENU_LINKS = HMENU
70  MENU_LINKS {
71      wrap = <br>|
```

Beispiel 7-10: Eine zusätzliche weiße Linie mittels wrap-Funktion aufnehmen (Fortsetzung)

```
72     special = directory
73     special.value = 3
74     1 = GMENU
75     1.NO = 1
76     1.NO {
77         [...]
78     }
79 }
```

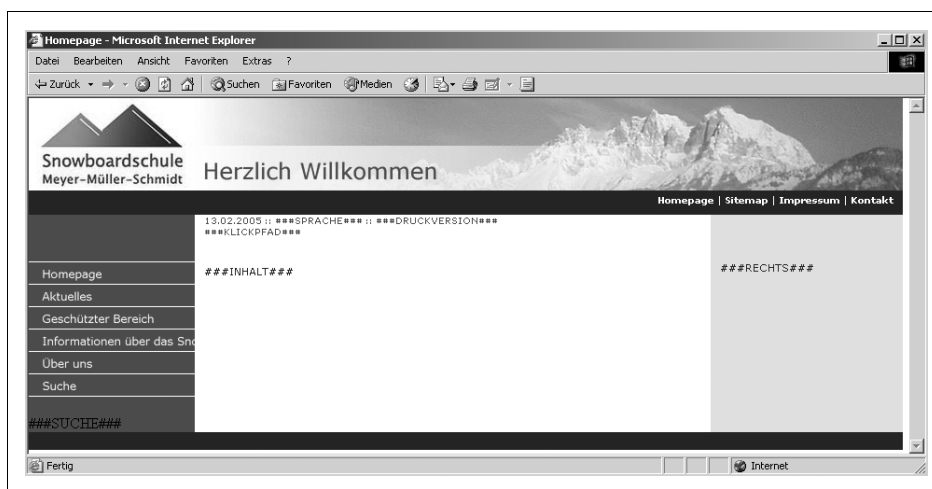


Abbildung 7-7: Das grafische Menü mit weißen Trennlinien

Unterschiedliche Menüzustände integrieren

Ebenfalls vom Grafiker vorgegeben ist das Layout eines Menüpunkts beim Eintreten eines RollOver-Zustands. Die grafische Beschreibung der unterschiedlichen Menüzustände beim RollOver weicht im Regelfall nur minimal vom normalen Zustand ab. Häufig zeigen sich nur dezente Veränderungen wie z.B. eine andere Hintergrundfarbe.

Um nicht den RO-Zustand über 20 Zeilen TypoScript-Code neu beschreiben zu müssen, kann die Beschreibung des normalen Zustands in den RO-Zustand kopiert und anschließend mittels Überschreibung von Eigenschaften abgeändert werden. Das Abändern eines Werts wie z.B. der Hintergrundfarbe ist notwendig, da ansonsten kein Effekt zu Stande käme.

In Beispiel 7-11 wird in Zeile 100 mittels relativer Kopie der Zustand NO in den Zustand RO kopiert. Bei diesem Kopiervorgang werden sämtliche Eigenschaften mit kopiert und stehen als Kopie in RO zur Verfügung. Da in Zeile 75 der Zustand NO bereits aktiviert wurde, wird durch den Kopiervorgang nun der Zustand RO

ebenfalls aktiviert. Erst durch das Überschreiben der Eigenschaft `.backColor` in Zeile 101 wird ein RO-Zustand optisch sichtbar.

Beispiel 7-11: Einen RollOver-Zustand aktivieren und beschreiben

```
[...]  
68     # Das grafische Menü erstellen  
69     MENU_LINKS = HMENU  
70     MENU_LINKS {  
       [...]  
74     1 = GMENU  
75     1.NO = 1  
76     1.NO {  
       [...]  
97     }  
98  
99     # Einen RollOver-Zustand beschreiben  
100    1.RO < .1.NO  
101    1.RO.backColor = #061467  
102    }  
[...]
```

Eine zweite Navigationsebene hinzufügen

In der Vorgabe des Praxisbeispiels ist für das linke Menü auch eine zweite Navigationsebene vorgesehen. Wenn es zu einer Seite Unterseiten gibt, sollen diese als Untermenü eine andere Darstellung erhalten. Die Hintergrundfarbe für diese zweite Navigationsebene soll heller sein und der Text in einem Dunkelblau erscheinen. Außerdem soll der grafische Text der zweiten Ebene einen Punkt kleiner sein.

Unterseiten anlegen

Doch bevor Sie die zweite Navigationsebene mittels TypoScript beschreiben, ist es sinnvoll, zunächst im Seitenbaum Unterseiten anzulegen. Der jetzt vorgestellte Weg ist eine Alternative zu dem in Kapitel 4, *Das Praxisbeispiel vorbereiten*, angewendeten und eignet sich insbesondere dann, wenn Sie mehrere Seiten gleichzeitig anlegen möchten.

Wählen Sie im Backend, wie in Abbildung 7-8 zu sehen, aus dem linken Menü das Backend-Modul *Funktionen* aus ❶. Für das Praxisbeispiel sollen für die Seite *Informationen über das Snowboardgebiet* Unterseiten angelegt werden. Wählen Sie daher aus dem Seitenbaum diese Seite aus ❷, indem Sie auf den Textlink der Seite klicken. Auf der rechten Seite erscheint nun eine Maske ❸, in der Sie bis zu neun Unterseiten auf einen Schlag anlegen können. Geben Sie hier die drei gewünschten Unterseiten *Geschichte*, *Mitarbeiter* und *Anfahrtsbeschreibung* an ❹. Bestätigen Sie anschließend das Anlegen der Seiten mit dem Button *Seiten anlegen* ❺.



Achten Sie darauf, dass Sie die Unterseiten auch auf der richtigen Seite anlegen. Das Anlegen von mehreren Seiten geht über die Funktion *Erzeuge mehrere Seiten* schnell, das Löschen von Seiten hingegen muss manuell über den Seitenbaum erfolgen.

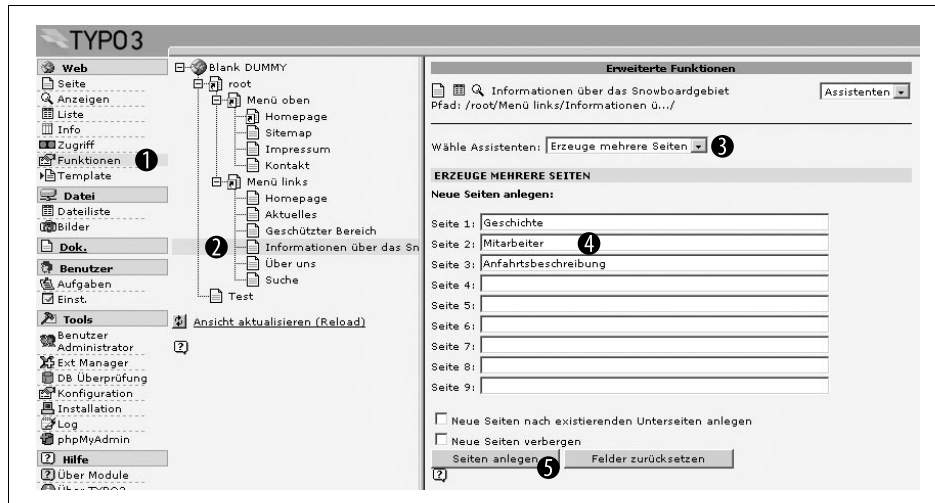


Abbildung 7-8: Mehrere Seiten mit einem Assistenten anlegen

Die zweite Navigationsebene beschreiben

In den vorherigen Beispielen haben Sie die erste Menüebene mittels `.1 = GMENU`, `.1.NO = 1` etc. beschrieben. Die `.1` steht hierbei für die erste Navigationsebene, es liegt also auf der Hand, dass ein `.2` die zweite Navigationsebene beschreibt.

In folgendem Beispiel wird in Zeile 104 auf die zweite Ebene die Beschreibung der ersten Ebene kopiert. Die Nummern `1, 2, ...` geben hierbei die Navigations- bzw. Menüebenen an. Lediglich die TypoScript-Beschreibung der ersten Menüebene wird in die zweite Menüebene kopiert und nicht der Inhalt der Menüeinträge. Somit ist die Beschreibung der zweiten Menüebene identisch zur ersten.

Beispiel 7-12: Die zweite Ebene als Kopie der ersten Ebene anlegen

```
[...]  
68 # Das grafische Menü erstellen  
69 MENU_LINKS = HMENU  
70 MENU_LINKS {  
    [...]  
76     1.NO {  
        [...]  
97     }  
98 }
```

Beispiel 7-12: Die zweite Ebene als Kopie der ersten Ebene anlegen (Fortsetzung)

```
99      # Einen RollOver-Zustand beschreiben
100     1.RO < .1.NO
101     1.R0.backColor = #061467
102
103     # Die zweite Ebene als Kopie der ersten Ebene beschreiben
104     2 < .1
105   }
[...]
```

Ein Blick in das Frontend zeigt, dass bisher noch keine zweite Menüebene auftaucht, die zum Beispiel erscheinen müsste, wenn die Seite *Informationen über das Snowboardgebiet* aufgerufen wird, obwohl im TypoScript-Template so weit alles in Ordnung scheint. Sichtbar wird die zweite Ebene erst, wenn Sie die im Folgenden beschriebene Eigenschaft `.entryLevel` gesetzt haben.

entryLevel für weitere Menüebenen setzen

Um bei der Verwendung der `.special`-Eigenschaft des HMENU-Objekts weitere Ebenen darstellen zu können, ist es erforderlich, die Eigenschaft `.entryLevel` zu verwenden. `.entryLevel` gibt an, auf welcher Ebene sich der Ausgangspunkt, in unserem Fall also die Seite *Menü links*, innerhalb des TYPO3-Seitenbaums befindet, und ist eine Eigenschaft des HMENU-Objekts. Der Ausgangspunkt zur Beschreibung des Menüs auf der linken Seite wurde über die Eigenschaft `.special.value` angegeben. Diese Eigenschaft hat als Wert die eindeutige Seiten-ID der Hilfsseite *Menü links* zugewiesen bekommen.

Die Seite *root* befindet sich auf Ebene 0, die Seiten *Menü links* und *Menü oben* auf Ebene 1. Folgende Abbildung soll dieses veranschaulichen.

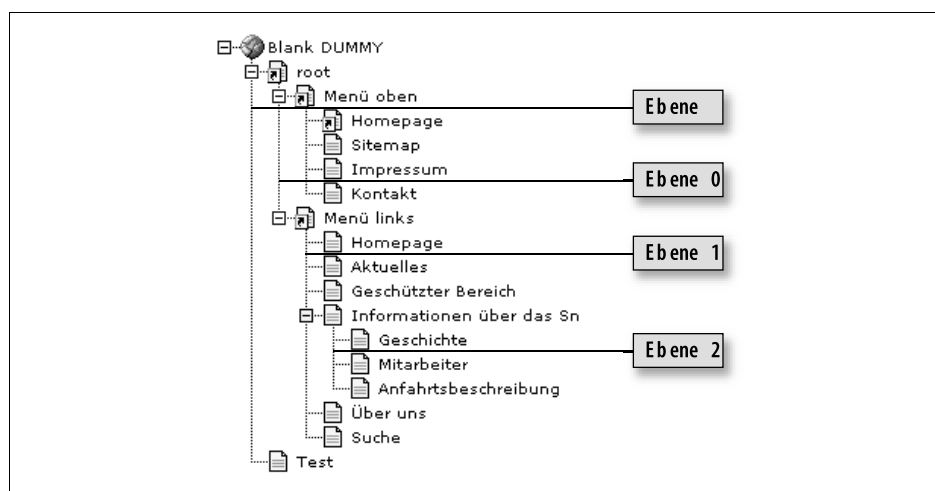


Abbildung 7-9: Die Ebenen für `entryLevel` im Seitenbaum

In TypoScript haben Sie durch `.special.value = 3` angegeben, dass das Menü von der Seite mit der eindeutigen ID 3 (*Menü links*) aufgebaut werden soll. Wie in Abbildung 7-9 zu sehen, befindet sich diese Seite *Menü links* in der Baumdarstellung auf Ebene 1. Diese Ebene muss bei der HMENU-Eigenschaft `.entryLevel` nun angegeben werden, wie in Beispiel 7-13 in Zeile 74 zu sehen.

Beispiel 7-13: entryLevel setzen, damit die zweite Menüebene erscheint

```
[...]  
68 # Das grafische Menü erstellen  
69 MENU_LINKS = HMENU  
70 MENU_LINKS {  
71     wrap = <br>  
72     special = directory  
73     special.value = 3  
74     entryLevel = 1  
75     1 = GMENU  
76     1.NO = 1  
77     1.NO {  
           [...]  
98     }  
           [...]  
106 }  
[...]
```

Ein erneutes Betrachten der Seite im Frontend sollte Sie zufrieden stimmen, für die Seite *Informationen über das Snowboardgebiet* wird nun die zweite Menüebene dargestellt. Diese zweite Menüebene soll nun aber einige andere Eigenschaften bekommen. Wie bereits oben erwähnt, soll die Hintergrundfarbe heller dargestellt werden und der Text in einem Dunkelblau sowie einen Punkt kleiner erscheinen. Nehmen Sie hierzu die in Beispiel 7-14 beschriebenen Erweiterungen vor, mit denen Sie die kopierten Eigenschaften den Vorgaben anpassen. Als Ergebnis sollten Sie im Frontend das in Abbildung 7-10 gezeigte Resultat erhalten.

Beispiel 7-14: Die zweite Menüebene beschreiben

```
[...]  
104 # Die zweite Ebene als Kopie der ersten Ebene beschreiben  
105 2 < .1  
106 2.NO {  
107     backColor = #D3E2F9  
108     10.fontColor = #061467  
109     10.fontSize = 11  
110 }  
111 2.R0 < .2.NO  
112 2.R0.backColor = #C1D5F4  
[...]
```

In Zeile 105 wird die Eigenschaften der ersten Ebene in die zweite kopiert. Die zweite Ebene ist damit, sofern die Eigenschaft `.entryLevel` richtig gesetzt ist, funktionsfähig. In den folgenden drei Zeilen werden die kopierten Eigenschaften mit neuen Werten versehen, hier die Hintergrundfarbe der Grafik in Zeile 107, die Schriftfarbe in Zeile 108 sowie die Schriftgröße in Zeile 109.

In Zeile 111 werden die Eigenschaften des normalen Zustands der zweiten Menüebene in den RollOver-Zustand für die zweite Ebene kopiert. Da der RollOver-Zustand jedoch schon existiert (wurde in Zeile 105 mit kopiert), findet durch das Kopieren nur ein Überschreiben der Eigenschaften statt. In Zeile 112 wird dann die Hintergrundfarbe für den RO-Zustand auf einen leichten Blauton gesetzt.



Abbildung 7-10: Das Menü mit einer abgeänderten zweiten Navigationsebene

Zeilenumbruch innerhalb des grafischen Texts

Ist der Text eines Menüpunkts zu lang, wie z.B. der Seitentitel *Informationen über das Snowboardgebiet*, wird dieser zum jetzigen Zeitpunkt zwangsweise abgeschnitten bzw. fließt aus der Grafik heraus und ist somit nicht mehr sichtbar. Es muss also ein Weg gefunden werden, um einen Zeilenumbruch zu erzwingen und die Grafik entsprechend zu vergrößern.

Leider kann TYPO3 bei grafischem Text nicht von allein erkennen, dass ein Zeilenumbruch erfolgen soll. Der Redakteur muss explizit angeben, an welcher Stelle eine zweite oder auch dritte Zeile beginnen soll. Dies erfolgt über ein spezielles Zeichen, das Sie frei definieren können. In der Praxis hat sich hierfür das Pipe-Symbol `|` bewährt, aber auch alle anderen Zeichen können natürlich verwendet werden.

Um nun einen Zeilenumbruch für die Seite *Informationen über das Snowboardgebiet* zu ermöglichen, muss der Seitentitel um das Pipe-Symbol erweitert werden.

Zum Bearbeiten einer Seite klicken Sie in der Seitenstruktur auf das Icon vor dem Textlink und wählen aus dem sich öffnenden Pop-up-Menü den Eintrag *Bearbeite*

Seiten-Header aus. Sie können nun den Titel der Seite auf *Informationen über das|Snowboardgebiet* ändern, indem Sie das Pipe-Symbol zwischen die beiden Wörtern *das* und *Snowboardgebiet* einfügen. Achten Sie auch darauf, dass sich vor und nach dem Pipe-Symbol keine Leerzeichen befinden. Leerzeichen im Seitentitel stellen zwar für TYPO3 keine Problem dar, allerdings würde dann in diesem Fall die zweite Zeile mit einem Leerzeichen beginnen.

Wenn Sie sich das Ergebnis im Frontend betrachten, werden Sie erkennen, dass das Pipe-Symbol nun direkt im Text steht und der Text noch immer nicht vollständig angezeigt wird. Ein erhoffter Zeilenumbruch ist bisher nicht sichtbar und muss mittels TypoScript zunächst beschrieben werden.

Die Eigenschaft `.listNum` haben Sie bereits im Kapitel 6, *Grafiken mit TypoScript erstellen*, kennen gelernt, als Sie die Datei für den Trailer dynamisch aus einem Datenbankfeld *media* ausgelesen haben. Mit dieser Eigenschaft haben Sie angegeben, dass nur die erste hochgeladene Datei zu einer Seite ausgelesen werden soll, also das erste Element in einer Liste.

Mit der Eigenschaft `.listNum.splitChar` können Sie eine gelieferte Zeichenkette auch in eine so genannten Liste aufteilen. Welches Zeichen zur Aufteilung verwendet werden soll, geben Sie bei dieser Eigenschaft als Wert an. Die Eigenschaft `.listNum` erwartet als Wert die gewünschte Nummer des Listenelements. In Abbildung 7-11 finden Sie eine Übersicht, die zeigt, wie eine Zeichenkette, z.B. der Titel einer Seite, in einzelne Elemente aufgeteilt wird.



Bitte beachten Sie, dass in Programmiersprachen und auch in TypoScript die Kennzeichnung eines Listenelements häufig nicht mit der Eins, sondern mit einer Null beginnt.

Informationen über das Snowboardgebiet	
Aufteilung anhand des in <code>.listNum.splitChar</code> angegebenen Zeichens, z.B. dem Pipe-Symbol	
Informationen über das	Erstes Element, <code>listNum = 0</code>
Snowboardgebiet	Zweites Element, <code>listNum = 1</code>

Abbildung 7-11: Aufbau einer Liste mit `.listNum.splitChar`

Diese Funktionalität soll nun auch im Praxisbeispiel Anwendung finden. In Beispiel 7-15 wurde in Zeile 84 über die Eigenschaft `.listNum` angegeben, dass aus einer Liste von Elementen das erste Element verwendet werden soll. In Zeile 85 wird in der Eigenschaft `.listNum.splitChar` angegeben, durch welches Zeichen der Text getrennt und die Liste aufgebaut werden soll. Folglich befindet sich im ersten Ele-

ment der Liste (gekennzeichnet durch die Null) der Text *Informationen über das*, und auch nur dieser Text wird zunächst in der Grafik verarbeitet.

Beispiel 7-15: Den Titel in eine Liste aufteilen und nur das erste Listenelement ausgeben

```
[...]
70 MENU_LINKS {
    [...]
77     1.NO {
        [...]
81         # Text auf den Menüpunkt rendern
82         10 = TEXT
83         10.text.field = title
84         10.text.listNum = 0
85         10.text.listNum.splitChar = |
86         10.fontColor = #FFFFFF
87         10.fontFile = fileadmin/fonts/verdana.ttf
88         10.fontSize = 12
89         10.niceText = 1
90         10.offset = 14, 16
        [...]
    }
}
```

Das Realisieren eines zweizeiligen Menüs könnte theoretisch sehr einfach sein, gäbe es da nicht ein paar Kleinigkeiten, die berücksichtigt werden müssten. Aber sehen Sie sich diese Problempunkte zunächst selbst an. Zum Realisieren einer zweiten Zeile könnte jetzt recht einfach eine Ebene 15 implementiert werden, die den Text des zweiten Listeneintrags ausgibt. Diese Ebene 15 würde einen anderen Offset erhalten, damit Ebene 15 Ebene 10 nicht überdeckt, sondern beide Ebenen untereinander stehen.

Die Grafik muss allerdings so hoch sein, dass dieser Text noch Platz hat. Mit dem bisher gelernten Wissen könnten Sie an der Eigenschaft XY Änderungen vornehmen, so dass jeder Menüpunkt eine ausreichende Höhe hat. Diese Angabe muss aber nicht statisch mit einer absoluten Pixelzahl sein, Sie können auch berechnen lassen, wie hoch ein jeweiliger Menüpunkt sein muss, damit eine ggf. vorhandene zweite Zeile genug Platz hat. Dazu ist folgende Syntax notwendig:

$$XY = [\text{Ebene.w}] + Z, [\text{Ebene.h} + Z]$$

Die Angabe $XY = 178, 24 + [15.h]$ gibt somit an, dass das Menüelement 178 Pixel breit sein soll. Die Höhe wird dynamisch berechnet – in diesem Fall also 24 Pixel plus der Höhe von Ebene 15. Wenn Ebene 15 keinen Text enthält, führt dieses zu einer Addition von $24 + 0$. Enthält Ebene 15 einen Text, weil es ein zweites Listenelement gibt, wird die Höhe des Texts der Ebene 15 zu den 24 Pixeln hinzuaddiert.

In Beispiel 7-16 wird die Höhe der Grafik dynamisch berechnet, abhängig davon, wie hoch der Text der zweiten Ebene tatsächlich ist. Hierzu wird in Zeile 93 eine Ebene 15 eingeführt, die identisch mit Ebene 10 ist und die Eigenschaften `.listNum` und `.offset` überschreibt.

Theoretisch reicht eine Ebene 15 zum Berechnen der Höhe aus, jedoch gibt es ein Problem bei dieser Berechnung: ein Buchstabe *g* hat eine andere Höhe als der Buchstabe *a*. Sie können nicht wissen, welchen Text der Redakteur für den Titel wählt. Mit etwas Unglück lautet der Text für die zweite Zeile zum Beispiel *so war emma*, der eine andere Höhe einnimmt als der Text *es gibt Emma*. Durch eine Konvertierung des Texts in Großbuchstaben wird erreicht, dass die gleiche Höhe für beliebigen Text erreicht wird.

Um dieses Problem zu umgehen, wird in Zeile 98 eine zusätzliche Ebene 16 eingeführt, die ebenfalls eine identische Beschreibung der Ebene 10 enthält, praktisch nicht sichtbar ist und nur für die Berechnung der Höhe der Grafik benötigt wird. Das *Unsichtbarmachen* dieser Ebene wird durch einen Offset erreicht, der garantiert nicht mehr im Bereich der Grafik liegt. Durch einen Offset von jeweils 1.000 Pixeln nach rechts und nach unten (Zeile 102) wird der Textbereich somit außerhalb der Grafik liegen. In Zeile 99 lässt die Eigenschaft `.text.case` den Text nur in Großbuchstaben erscheinen. Die Schriftgröße wird in Zeile 101 auf einen höheren Wert gesetzt, damit die Grafik bei der Höhe auch einen Abstand zwischen den beiden Texten berücksichtigt.

In Zeile 78 wird die Berechnung der Höhe eines Menüpunkts dynamisch angegeben. Als Wert für die Höhe werden hier 24 Pixel zur Höhe der Ebene 16 hinzuaddiert. Beachten Sie aber, dass der tatsächliche Inhalt aus Ebene 15 kommt und nicht aus Ebene 16, die mit einem unmöglichen Offset außerhalb der Grafik liegt.



Die Berechnung der Höhe ist abhängig von der verwendeten TTF-Schriftart. Bei einigen exotischen Schriften führt obiger Trick mit der Umwandlung in Großbuchstaben ggf. nicht zum Erfolg, da in der verwendeten Schriftart auch dann noch Unterschiede bei der Höhe des Texts auftreten. Ändern Sie die Schriftart für die unsichtbare Ebene dann einfach in eine *normale* Schriftart, bei der solche Effekte nicht eintreten.

Nun muss auch bei der weißen Linie dieser berechnete Offset mit eingearbeitet werden, damit sich die Grafik auch immer am unteren Rand befindet. Die entsprechende Berechnung wurde mit in das folgende Beispiel in Zeile 111 eingearbeitet. Das Ergebnis können Sie in Abbildung 7-12 sehen.

Beispiel 7-16: Das Menü mit einem zweizeiligen Text darstellen

```
68     # Das grafische Menü erstellen
69     MENU_LINKS = HMENU
70     MENU_LINKS {
71         [...]
77         1.NO {
78             XY = 178, 24+[16.h]
79             backColor = #364497
80     }
```

Beispiel 7-16: Das Menü mit einem zweizeiligen Text darstellen (Fortsetzung)

```
81         # Text auf den Menüpunkt rendern
82         10 = TEXT
83         10.text.field = title
84         10.text.listNum = 0
85         10.text.listNum.splitChar = |
86         10.fontColor = #FFFFFF
87         10.fontFile = fileadmin/fonts/verdana.ttf
88         10.fontSize = 12
89         10.niceText = 1
90         10.offset = 14, 16
91
92         # Eine ggf. vorhandene zweite Textzeile
93         15 < .10
94         15.text.listNum = 1
95         15.offset = 14, 32
96
97         # Hilfebene zur Berechnung der Höhe von Menüpunkten
98         16 < .10
99         16.text.case = upper
100        16.text.listNum = 1
101        16.fontSize = 22
102        16.offset = 1000,1000
103
104        # Eine weisse Linie erzeugen
105        20 = IMAGE
106        20.file = GIFBUILDER
107        20.file {
108            XY = 178, 1
109            backColor = #FFFFFF
110        }
111        20.offset = 0, 23+[16.h]
112    }
```



Abbildung 7-12: Das zweizeilige Menü wird angezeigt

Fehlerhafte Anzeige im Trailer

Nachdem Sie den Seitentitel der Seite *Informationen über das Snowboardgebiet* bearbeitet und dem Titel das Pipe-Symbol hinzugefügt haben, wird im Trailer ebenfalls ein Pipe-Symbol angezeigt. Sie haben zur Behebung dieses Problems zwei Möglichkeiten:

- Der Seitentitel wird nochmals im Feld *Untertitel* angegeben – ohne Pipe-Symbol. Der Redakteur muss beim Anlegen oder Bearbeiten einer Seite, die im Menü einen Zeilenumbruch erhalten soll, darauf achten, einen Untertitel ohne dieses Pipe-Symbol anzulegen.
- Der grafische Trailer wird ebenfalls in eine Liste aufgeteilt, und die einzelnen Elemente werden dynamisch wieder zusammengesetzt. Diese Vorgehensweise ist die sicherste und unempfindlichere Möglichkeit. Hierbei wird das oben gelernte Wissen über Listen auch auf den Trailer angewendet.

Fehlender HTML-Zeilenumbruch

Sie könnten fast zufrieden sein, gäbe es da nicht noch einen kleinen Haken. Warum wird das Menü überhaupt untereinander dargestellt? Wie würde man denn ein Menü realisieren, das nebeneinander stehen soll? Ein Blick in den HTML-Quellcode (Abbildung 7-13) verrät Ihnen, dass die Grafiken tatsächlich ohne Zeilenumbruch dargestellt werden:

```
<td valign="top" align="left" width="178" bgcolor="#364497">
  <br>
  <br><a href="index.php?id=14"
onfocus="blurLink(this);" onmouseover="over('img14_29c6_0');" onmouseout="out('img14_29c6_0');"></a><a
href="index.php?id=13" onfocus="blurLink(this);" onmouseover="over('img13_29c6_1');" onmouseout="out('img13_29c6_1');"></a><a
href="index.php?id=12" onfocus="blurLink(this);" onmouseover="over('img12_29c6_2');" onmouseout="out('img12_29c6_2');"></a><a
href="index.php?id=11" onfocus="blurLink(this);" onmouseover="over('img11_29c6_3');" onmouseout="out('img11_29c6_3');"></a><a href="index.php?id=10" onfocus="blurLink(this);" onmouseover="over('img10_29c6_4');"
onmouseout="out('img10_29c6_4');"></a><a href="index.php?id=9" onfocus="blurLink(this);" onmouseover="over('img9_29c6_5');"
onmouseout="out('img9_29c6_5');"></a><br>
  <br>
  ###SUCHE###
</td>
```

Abbildung 7-13: Der HTML-Quellcode der Menüelemente enthält keinen `
`-Tag

Der Zeilenumbruch wird (freundlicherweise) vom Browser eigenmächtig vorgenommen, bedingt durch die Tabellenzelle, der eine Breite von 178 Pixeln in der Designvorlage zugewiesen wurde. Um den Zeilenumbruch dennoch *sauber* zu erstellen, muss nach jedem Menüeintrag ein `
`-Tag eingefügt werden. Dies wird in Beispiel 7-17 wieder über die `.wrap`-Funktion gelöst.

Beispiel 7-17: Ein Zeilenumbruch nach jedem Menüeintrag

```
[...]
68 # Das grafische Menü erstellen
69 MENU_LINKS = HMENU
```

Beispiel 7-17: Ein Zeilenumbruch nach jedem Menüeintrag (Fortsetzung)

```
70 MENU_LINKS {  
    [...]  
75     1 = GMENU  
76     1.NO = 1  
77     1.NO {  
78         wrap = |<br>  
79         XY = 178, 24+[16.h]  
80         backColor = #364497  
        [...]
```

Einen Klickpfad realisieren

Für die Bezeichnung *Klickpfad* werden häufig auch andere Bezeichnungen, wie z.B. *Brotkrumenpfad* oder das englische Wort *breadcrumb (trail)* verwendet, alle bezeichnen aber das gleiche Vorhaben: ein *Sie befinden sich hier*-Menü, wie Sie es in Abbildung 7-14 sehen können. Ein Platzhalter KLICKPFAD wurde bereits in der Designvorlage definiert und soll zur Umsetzung dieses Menüs verwendet werden.



Abbildung 7-14: Der Klickpfad auf unserer Website

Auch bei der Realisierung des Klickpfads handelt es sich wieder um ein HMENU-Objekt. Allerdings wird für die Eigenschaft `.special` dieses mal nicht als Wert *directory* angegeben, der ein normales Menü kennzeichnet, sondern der Wert *rootline*, mit dem sich ein Klickpfad realisieren lässt.

Auch bei der Erstellung eines Klickpfads muss eine Ebene definiert werden. Allerdings reicht es hier aus, eine Ebene 1 für den Zustand NO zu beschreiben, so wie in Beispiel 7-18 geschehen. Alle folgenden Ebenen werden identisch verarbeitet und müssen nicht explizit angegeben werden.

Beispiel 7-18: Einen Klickpfad erstellen

```
01 page = PAGE  
02 page {  
    [...]  
16     10.marks {  
        [...]
```

Beispiel 7-18: Einen Klickpfad erstellen (Fortsetzung)

```
130      # Einen Klickpfad erstellen
131      KLIICKPFAD = HMENU
132      KLIICKPFAD {
133          special = rootline
134          1 = TMENU
135          1.NO = 1
136      }
137  }
138 }
```

Beim Betrachten des Ergebnisses im Frontend werden Sie feststellen, dass der Klickpfad bereits angezeigt wird. Allerdings stimmen derzeit weder die Formatierung noch der Startpunkt, an dem der Klickpfad beginnt. Die Formatierung des Klickpfads lässt sich wieder mit der TMENU-Objekt-Eigenschaft `.linkWrap` realisieren. In folgendem Beispiel wird nach jedem Menüeintrag ein Leerzeichen, ein Schrägstrich und abschließend noch ein Leerzeichen ausgegeben.

Beispiel 7-19: Die einzelnen »breadcrumbs« formatieren

```
[...]
130      # Einen Klickpfad erstellen
131      KLIICKPFAD = HMENU
132      KLIICKPFAD {
133          special = rootline
134          1 = TMENU
135          1.NO = 1
136          1.NO.linkWrap = |&nbsp;/&nbsp;
137      }
[...]
```

Den Startpunkt des Klickpfads setzen

Im Backend können Sie erkennen, dass die Hilfsseiten Bestandteil des Klickpfads sind. Diese Hilfsseiten sollen im Klickpfad allerdings nicht erscheinen und werden in Beispiel 7-20 über die Eigenschaft `.special.range` des HMENU-Objekts aus dem entsprechenden Bereich genommen.

Hierzu wurde in Beispiel 7-20 in Zeile 134 der Eigenschaft `.special.range` der Wert `2|-1` zugewiesen. Dieser Wert gibt an, dass der Klickpfad erst an der zweiten Ebene der Root-Ebene beginnen und nie enden (`-1`) soll. Die zweite Ebene ist eine öffentliche und sichtbare Seite, da die Seite *root* sich auf Ebene 0 befindet und die Hilfsseiten *Menü links* und *Menü oben* auf Ebene 1 (siehe auch Abbildung 7-9).

Beispiel 7-20: Den Start- und Endpunkt mit `.special.range` angeben

```
[...]
130      # Einen Klickpfad erstellen
131      KLIICKPFAD = HMENU
```


An einem Platzhalter mehrere Objekte ausführen

Die dritte Möglichkeit, die hier ausführlich vorgestellt wird, erlaubt es, an einem Platzhalter gleich mehrere Objekte in einer angegebenen Reihenfolge auszuführen. In der Praxis ist es allerdings nicht möglich, einem Platzhalter mehrere Objekte zuzuweisen, daher wird ein neues Objekt COA eingeführt. Das Objekt COA kann mehrere Objekte mit ihren jeweiligen Definitionen in einer numerischen Liste aufnehmen. Diese numerische Liste wird wieder mit den Zahlen 10, 20, 30, ... beschrieben, wobei jedes einzelne Listenelement genau einem Objekt zugewiesen werden kann. Die Reihenfolge der Ausführung wird mit der Zahl bestimmt. Die Zahlen der numerischen Liste müssen nicht in Zehnerschritten angegeben werden, dieser Zahlenabstand hat sich allerdings als praktisch erwiesen, um nachträglich Korrekturen vornehmen zu können.

Es kann nun erreicht werden, dass vor der Ausführung des HMENU-Objekts zunächst ein TEXT-Objekt ausgeführt wird, das den Text *Sie befinden sich hier* ausgibt. In folgendem Beispiel wird in Zeile 133 bei dem Platzhalter KLICKPFAD eine COA-Instanz erzeugt. Der vorgeschaltete Text wird an Position 10 mit dem TEXT-Objekt und der Eigenschaft .value ausgegeben. An Position 20 folgt die Ausgabe des eigentlichen Klickpfads:

Beispiel 7-21: Das Objekt COA anwenden

```
[...]  
130 # Den Platzhalter KLICKPFAD ansprechen  
131 # Objekt COA wird zwischengeschaltet, um einen zusätzlichen  
132 # Text angeben zu können.  
133 KLICKPFAD = COA  
134 KLICKPFAD {  
135     # Der vorgeschaltete Text  
136     10 = TEXT  
137     10.value = Sie befinden sich hier:&nbsp;  
138  
139     # Den Klickpfad an Position 20 darstellen  
140     20 = HMENU  
141     20 {  
142         special = rootline  
143         special.range = 2|-1  
144         1 = TMENU  
145         1.NO = 1  
146         1.NO.linkWrap = |&nbsp;/&nbsp;  
147     }  
148 }  
[...]
```

