HTML Grafiken

Stand: 15.08.2022

Grafiken mit HTML

Image-Maps (Verweis-Sensitive Grafiken)

Anklickbare **Grafiken** sind auf fast jeder Webseite zu finden. Dies sind einfach Grafiken, die in einen Hyperlink eingebettet und dort anstelle eines Verweistextes notiert werden. Das Schema lautet `<a href=....><img></a>`.
Wichtig ist in diesem Fall zu wissen, dass Browser dem `img`-Element in diesem Fall, sofern Sie nichts anderes angeben, einen Rahmen in der Farbe für Verweise verpassen. Durch `<img ... style="border:none">` können Sie den Rahmen unterdrücken. Bedenken Sie jedoch, dass der Anwender dadurch möglicherweise übersieht, dass die Grafik ein Link ist.

> Imagemaps oder verweis-sensitive Grafiken gehen noch einen Schritt weiter als anklickbare Grafiken.

Ein Imagemap ist eine Grafik mit anklickbaren Bereichen, wobei jeder Bereich sein eigenes Linkziel haben kann. Ein Beispiel soll dies demonstrieren. Dabei soll ein vollständig grafisches Menü die Navigation auf einer Startseite ermöglichen. Die Grafik
sieht folgendermaßen aus: Aus den sechs beschrifteten Ellipsen sollen bei der Anzeige im Browser sechs anklickbare Schaltflächen werden, die beim Anklicken zu entsprechenden Verweiszielen führen. Der HTML-Code dazu sieht folgendermaßen aus:

```
<map name="navigation">
<area shape="rect" coords="5,6,274,66" href="/satzung/" alt="Satzung"
title="Satzung">
<area shape="rect" coords="55,72,325,135" href="/mitglieder/" alt="Mitglieder"
title="Mitglieder">
<area shape="rect" coords="113,146,380,206" href="/service/" alt="Service"
title="Service">
<area shape="rect" coords="530,6,800,66" href="/partner/" alt="Partner"
title="Partner">
<area shape="rect" coords="451,72,724,135" href="/aktionen/" alt="Aktionen"
title="Aktionen">
<area shape="rect" coords="395,146,661,206" href="/kontakt/" alt="Kontakt"
title="Kontakt">
</map>
<p>
<img src="navigation.jpg" style="width:807px; height:213px; border:none"
alt="Navigation" usemap="#navigation">
</p>
```

Info Die Grafik selbst wird wie im unteren Teil des Quelltextausschnitts zu sehen wie üblich über das `img`-Element referenziert. Der Bezug zum Imagemap wird über das zusätzliche Attribut `usemap=` hergestellt. Dem Attribut wird der URI einer zugehörigen **Map-Definition** im gleichen Dokument zugewiesen. Der Wert wird also genauso notiert wie ein Link zu einem Anker innerhalb des Dokuments, mit führendem Gatterzeichen (#),
gefolgt vom Ankernamen.

Der Anker, zu dem der Bezug führt, ist das

`map`-Element, markiert durch `<map> ... </ map>`. Dieses Element kann irgendwo zwischen `<body>` und `</body>` notiert werden. In der Regel wird es jedoch an
zentraler Stelle notiert, z.B. gleich hinter dem einleitenden `<body>`-Tag. Das `map`-Element dient als Behälter für Definitionen von Verweis-Sensitiven Flächen. Das obligatorische `name`-Attribut im einleitenden `<map>`-Tag erzeugt den Anker, der bei `usemap=` anzugeben ist. Die Definitionen der Verweis-Sensitiven Flächen werden durch `<area>`-Tags markiert. Diese haben in HTML kein End-Tag,
müssen in XHTML also in der Form `<area ... />` notiert werden. Die Verweis-Sensitiven Flächen werden in diesem Tag durch die Attribute `shape=` und `coords=` festgelegt. Das `shape`-Attribut
bezeichnet die Art der Fläche und das `coords`-Attribut die Pixelkoordinaten der Punkte, mit deren Hilfe sich die Fläche beschreiben lässt. Als Flächenformen stehen `shape="rect"` (Rechteck), `shape="circle"` (Kreis) und `shape="poly"` (Polygon, beliebiges Vieleck) zur Verfügung. Das Schema der Wertzuweisung an das `coords`-Attribut ist von der Angabe zum `shape`-Attribut abhängig.

shape=	Schema für coords=	Erläuterung
`rect`	x1,y1,x2,y2	x1 = linke obere Ecke, Pixel von links y1 = linke obere Ecke, Pixel von oben x2 = rechte untere Ecke, Pixel von links y2 = rechte untere Ecke, Pixel von oben
`circle`	x,y,r	x = Mittelpunkt, Pixel von links y = Mittelpunkt, Pixel von oben r = Radius in Pixel
`poly`	x1,y1,x2,y2,... xn,yn	x = Pixel einer Ecke von links y = Pixel einer Ecke von oben Von der letzten definierten Ecke müssen Sie sich eine Linie zur ersten definierten Ecke hinzudenken. Diese schließt das Polygon.

Info Obwohl die zum Anklicken vorgesehenen Flächen in unserem Beispiel elliptisch sind, verwenden wir der Einfachheit halber `shape="rect"` zur Definition der Verweisflächen. Die Flächenform „Kreis“ ist jedenfalls ungeeignet.
Mithilfe der Flächenform „Polygon“ und einer ganzen Kette von Eckpunkten könnten wir die Ellipsen zwar einigermaßen genau beschreiben, doch für den gewünschten Zweck genügt es, gedachte Rechtecke um die Ellipsen als Verweis-Sensitiv zu kennzeichnen.
Im `href`-Attribut wird das Linkziel angegeben. Es kann sich um einen beliebigen URI handeln, also um absolute Internetadressen oder um lokal referenzierte Quellen. Ebenso wie im `img`-Element ist im `area`-Element
die Angabe eines `alt`-Attributs Pflicht. Der Grund leuchtet ein: Wenn die Grafik nicht angezeigt werden kann, wären auch die Links überhaupt nicht visualisierbar. Das `alt`-Attribut ermöglicht es dem Browser,

anstelle
der fehlenden grafischen Links Textlinks mit den `alt`-Texten als **Verweistexten** anzuzeigen. Leider machen die meisten Browser keinen Gebrauch davon. Das `title`-Attribut haben wir in unserem Beispiel spendiert, damit dem Anwender beim Überfahren der Verweis-Sensitiven Flächen zusätzlich ein Tooltipp-Fenster angezeigt wird.

Grafiken anstelle von Verweistext

Wenn Sie Verweise setzen, müssen Sie immer auch einen **Verweistext** definieren, also den Text, der dem Anwender als anklickbar dargestellt wird. Anstelle eines Verweistextes können Sie jedoch auch eine Grafikreferenz notieren.
Dann ist die gesamte Grafik anklickbar, und beim Anklicken der Grafik wird der Verweis ausgeführt. Von dieser Möglichkeit wird in der Praxis sehr oft Gebrauch gemacht, bei Werbebannern zum Beispiel, aber auch bei grafischen Navigationsleisten.

```
<a href="datei.htm"><img src="datei.gif"></a>
```

Das Verweisziel kann ein beliebiges erlaubtes Ziel sein. Anstelle des Verweistextes wird einfach eine Grafik eingebunden.

Zielfenster für Verweise

Normalerweise wird die HTML-Datei, die von einem Verweis aufgerufen wird, in das Browserfenster bzw. in den Frame referenziert, in dem auch der Verweis notiert wurde. Nun gibt es aber die Möglichkeit, mittels des Attributs `target=` die Zieldatei in einen bestimmten **Frame** zu referenzieren, indem als Wert der Name des Frames angegeben wird. Schließlich können auch noch reservierte Werte dem Attribut `target=` übergeben werden, diese sind `_blank`,
`_parent` und `_top`. Diese werden in Beenden eines Framesets abgehandelt.

```
<a href="datei.html" target="rechts">Verweistext</a>
```

Hier wird ein Verweis definiert, der die HTML-Datei datei.html in einen Frame namens rechts referenziert. Sollte kein Frame mit diesem Namen existieren, so hat dies dieselbe Wirkung wie die Angabe `_blank` beim `target=`-Attribut,
es wird ein neues Browserfenster geöffnet.

Verweis-Rahmen unterdrücken

Der Rahmen, den der WWW-Browser per Voreinstellung um eine Grafik zieht, wenn diese als Verweis eingebunden wird, ist in vielen Fällen störend. Sie können den Rahmen unterdrücken.

```
<a href="datei.htm"><img src="datei.gif" border=0></a>
```

Durch die zusätzliche Angabe `border=0` unterdrücken Sie den Rahmen. Die Grafik bleibt weiterhin als Verweis anklickbar. Wenn Sie Rahmen um Grafiken unterdrücken, die als Verweise dienen, kann der Anwender die Grafik nicht mehr
unmittelbar als Verweis erkennen. Nur wenn er mit dem Mauszeiger über die Grafik fährt, kann er an dem Mauszeigersymbol erkennen, daß es sich um einen Verweis handelt. Deshalb sollten Sie den verweiskennzeichnenden Rahmen nur dann unterdrücken,
wenn die Grafik auf den ersten Blick als Verweis erkennbar ist. Das kann zum Beispiel ein beschrifteter Button sein, der dem Anwender intuitiv als Verweis erscheint.

Grafik zum umgebenden Text ausrichten

Da das `img`-Element ein Inline-Element ist, können Grafiken mitten in einem Text platziert werden. Wenn nun aber die **Grafik** höher ist als die Zeilenhöhe, dann muss der Text der gleichen Zeile in irgendeiner Weise
zur Grafik ausgerichtet werden. Wenn Sie nichts anderes angeben, wird der Text untenbündig zur Grafik ausgerichtet. Sie können jedoch mit Hilfe eines Attributs selbst bestimmen, wie der Text zur Grafik ausgerichtet werden soll. Dieses
Attribut ist allerdings als deprecated gekennzeichnet und soll künftig aus dem HTML-Standard entfallen. Die gleiche Wirkung lässt sich nämlich auch mit Stylesheets erzielen.

Mit dem HTML-Attribut `align=` im `<img>`-Tag können Sie Text in der gleichen Zeile zur Grafikausrichten (align = Ausrichtung):

- Mit `align="top"` wird der Text obenbündig zur Grafik ausgerichtet (top = oben).
- Mit `align="middle"` wird der Text mittig zur Grafik ausgerichtet (middle = mittig).
- Mit `align="bottom"` wird der Text untenbündig zur Grafik ausgerichtet (boxom = unten).

Info Die Browser interpretieren zum Teil noch weitere Angaben. Diese sind jedoch nicht im HTML-Standard verzeichnet und führen deshalb zu ungültigem **HTML**. Erwähnt seien sie trotzdem:

- Mit `align="texttop"` richten Sie den Text obenbündig zur Grafik aus, und zwar an der Oberkante des kleinsten Textes in der Zeile (texttop = oben am Text).
- Mit `align="absmiddle"` richten Sie den Text absolut mittig zur Grafik aus, auch bei unterschiedlichen Textgrößen (absmiddle = absolute middle = in jedem Fall mittig).
- Mit `align="absbottom"` richten Sie den Text untenbündig zur Grafik aus, und zwar an der Unterkante des kleinsten Textes in der Zeile (absbottom = absolute bottom = in jedem Fall untenbündig).
- Mit `align="baseline"` richten Sie den Text genauso aus wie mit der Standardangabe `align="bottom"`. Benutzen Sie deshalb besser die Standardangabe.

Grafiken, die Sie mit dem `<img>`-Tag referenzieren, können Sie links oder rechts ausrichten. Der umgebende Text fließt dabei um die Grafik. Mit zwei weiteren Attributen können Sie Abstand zum umfließenden Text erzeugen, damit
der Text nicht direkt an der Grafik „klebt“. Den automatischen Textumfluss können Sie auch abbrechen und die Textfortsetzung unterhalb der Grafik erzwingen. Alle dazu notwendigen Attribute sind allerdings aLs deprecated gekennzeichnet und
sollen künftig aus dem HTML-Standard entfallen. Die gleiche Wirkung lässt sich nähmlich auch mit

Stylesheets erzielen.
Info Mit `align="left"` richten Sie eine Grafik linksbündig aus. Nachfolgend notierter Text fließt rechts um die Grafik. Mit `align="right"` können Sie die Grafik rechtsbündig ausrichten. Der nachfolgend notierte
Text fließt dann links um die Grafik. Von „nachfolgend notiertem Text“ wird hier nur der Einfachheit halber gesprochen. Es kann sich um beliebige Elemente handeln, also z.B. auch andere Grafiken, Tabellen, Multimedia-Referenzen usw. Um Abstand
zwischen Grafik und umfließendem Text zu erzeugen, stehen die Attribute `hspace=` und `vspace=` zur Verfügung:

- Mit `hspace=` [Pixel] bestimmen Sie den Abstand zwischen Grafik und anderen Elementen links bzw. rechts davon (hspace = horizontal space = horizontaler Abstand)
- Mit `vspace=` [Pixel] bestimmen Sie den Abstand zwischen Grafik und anderen Elementen darüber bzw. darunter (vspace = vertical space = vertikaler Abstand).

Das Attribut `hspace=` betrifft immer den linken und den rechten Randabstand von der Grafik `vspace=` immer den oberen und den unteren Randabstand. Wenn Sie also beispielsweise bei einer links ausgerichteten Grafik `hspace=` definieren, wird nicht nur rechts der Grafik zum nebenstehenden Text ein Abstand erzeugt, sondern auch links zum Rand des Anzeigefensters hin. Um dies zu verhindern, können Sie exakter arbeiten, indem Sie Stylesheets verwenden. Sie können
beide Attribute notieren oder auch nur eines davon.

Um den automatischen Textumfluss zu unterbrechen und zu erzwingen,

dass alles was folgt unterhalb der Grafik angezeigt wird, können Sie einen Zeilenumbruch mit dem Attribut `clear=` notieren:

- Mit `<br clear="all">` erreichen Sie, dass der Textfluss ab der nächsten Zeile in jedem Fall unterhalb der Grafik fortgesetzt wird.
- Mit `<br clear="left">` erreichen Sie, dass der Textfluss ab der nächsten Zeile unterhalb einer linksbündig ausgerichteten Grafik fortgesetzt wird.
- Mit `<br clear="right">` erreichen Sie, dass der Textfluss ab der nächsten Zeile unterhalb einer rechtsbündig ausgerichteten Grafik fortgesetzt wird.

Grafiken ausrichten mit Stylesheets

Stylesheets bieten Eigenschaften an, mit deren Hilfe die beiden Funktionen, Grafik zum umgebenden Text ausrichten und Text um Grafik fließen lassen, ebenso realisierbar sind wie mit HTML. Das „Fein-Tuning“ ist mit Hilfe von CSS sogar deutlich genauer.
Mit CSS müssen Sie arbeiten, wenn Sie die in diesem Abschnitt beschriebenen Effekte erzielen und dabei aber mit der HTML-Variante „Strict“ arbeiten wollen. Maßgeblich sind im hier beschriebenen Zusammenhang folgende CSS-Eigenschaften:

- `vertical-align` (Vertikale Ausrichtung), `float` (Textumfluss), `clear` (Fortsetzung bei Textumfluss), Außenrand und Abstand.

Info Mit `style="vertical-align:text-top"` erreichen Sie in einem <img>-Tag das Gleiche wie mit `align="top"`, mit `style="vertical-align:middle"` das Gleiche wie mit `align=,"middle"`, und mit `style="vertical-align:text-bottom"` das Gleiche wie mit `align="bottom"`. Mit `style="float:left; margin-right:20px; margin-bottom:10px"` wird erreicht, dass die Grafik links ausgerichtet wird und
nachfolgender Text rechts um die Grafik fließt. Dabei wird zwischen Grafik und Text ein Abstand von 20 Pixeln gehalten (links von der Grafik wird dagegen, anders als beim HTML-Attribut `hspace=`, kein Abstand von 20 Pixeln erzeugt).
Nach unten hin, also zum Text unterhalb wird ein Abstand von 10 Pixeln definiert.

Grafikreferenzen

Grundsätzlich wird eine **Grafik** in HTML wie folgt eingebunden:

```
<img src="bild.png" alt="Bildkurzbeschreibung">
```

Info Das `img`-Element, ein alleinstehendes Element ohne End-Tag (XML-Notation: `<img…/>`), dient zur Referenzierung einer Grafik. Im obligatorischen `src`-Attribut wird der URI
der gewünschten Grafik angegeben. Es kann sich um eine absolute Internetadresse oder um eine lokal adressierte Grafikdatei handeln. Das `alt`-Attribut ist ebenso obligatorisch wie das `src`-Attribut. Da Grafiken aus verschiedensten
Gründen nicht angezeigt werden können – z.B. weil der Anwender die Grafikanzeige in seinem Browser aus Performance-Gründen deaktiviert hat oder weil er die Übertragung der Daten zur Webseite abgebrochen hat, sobald der gewünschte Text lesbar
war, oder weil er einen Browser in einer Textmodus-Umgebung wie einer Unix-Shell benutzt. In all diesen Fällen wird anstelle der Grafik der Alternativtext angezeigt, den Sie dem `alt`-Attribut zuweisen. Das `alt`-Attribut
ist nicht zu verwechseln mit dem `title`-Attribut, obwohl manche Browser das `alt`-Attribut beim Überfahren der Grafik mit der Maus ebenso wie das `title`-Attribut als Tooltipp-Fenster einblenden. Um einen
solchen Effekt gewollt in allen modernen Browsern zu erzielen, sollten Sie das title-Attribut zusätzlich zum alt-Attribut notieren.

> In der Regel sollten Sie dem Browser auch noch mehr Daten zur Grafik mitteilen.

Dazu gehören vor allem Angaben zu Höhe, Breite und Rahmen, aber auch zum Abstand zu Nachbarelementen, zur Ausrichtung usw. All diese Eigenschaften können Sie mithilfe von **CSS** zuweisen.

```
<img src="estland.jpg" style="width:300px; height:240px; border:none;
float:left;
margin-right:20px; margin-bottom:20px; alt="Typisches Motiv in Estland">
```

Info Weil die Angabe von Breite und Höhe eine besonders wichtige Information für den Browser darstellt – er kann dann den Raum für die Grafik beim Seitenaufbau bereits reservieren, noch bevor die Daten der Grafik geladen sind -, sind
auch im `Strict`-Standard die beiden HTML-Attribute `width=` und `height=` im `img`-Element erlaubt. Als Wert wird die Pixelgröße ohne Maßeinheit zugewiesen. Da jedoch meistens noch andere Eigenschaften zu notieren sind, die CSS erfordern, ist es zweckmäßig, auch die Angaben zu Breite und Höhe mithilfe der dafür vorgesehenen CSS-Eigenschaften `width` und `height` zu notieren.

Das Beispiel weiter oben zeigt, wie eine Grafik mithilfe von CSS

im Textfluss ansprechend positioniert wird. Die Angabe `float:left` bewirkt, dass nachfolgender Inhalt rechts um die Grafik herumfließt. Damit der Inhalt nicht unmittelbar an der Grafik klebt, wird mit Hilfe der Eigenschaften `margin-right` und `margin-bottom` für die betroffenen Seiten ein Sicherheitsabstand von je 20 Pixel definiert. Ein Rahmen wird mit `border:none` explizit unterdrückt. Das `img`-Element zählt zu den Inline-Elementen. Eine
Grafik wird also, wenn Sie nichts anderes festlegen, mitten im umgebenden Fließtext angezeigt. Nützlich ist dies vor allem bei Kleingrafiken mit Icon- oder Symbolcharakter. Viele Websites benutzen solche Grafiken beispielsweise, um **Hyperlinks** optisch zu typisieren.

```
<p>Sie können den Jahresbericht <img src="doc.gif" style="width:14px;
height:12px; border:none" alt="Webseite">
<a href="jb/index.htm">als Webseite</a></p>
```

oder als

```
<p><img src="pdf.gif" style="width:14px; height:12px; border:none" alt="PDF">
<a href="jb/jb.pdf" type="application/pdf">als PDF-Dokument</a> einsehen.</p>
```

Info Um eine Grafik aus dem Inline-Kontext zu lösen, müssen Sie das `img`-Element entweder in ein Block-Element einbinden, z.B. durch `<div><img...></div>`, oder dem Element mittels
CSS explizit Block-Charakter zuweisen (`<img ... style="display:block">`). Das `img`-Element kann natürlich auch in Tabellenzellen, in absolut positionierten Elementen usw. vorkommen. Dies erlaubt zusätzliche
Möglichkeiten der Anordnung der Grafik und der Zuordnung von Grafik und anderen Inhalten.

Grafikformate

HTML ist als **Auszeichnungssprache** für Text mit Hypertext-Funktionalität konzipiert. Es bietet keine eigene Auszeichnungsmöglichkeit für grafische Elemente an. Grafiken können ebenso wie beliebige andere multimediale Elemente
über entsprechende Schnittstellen-Elemente referenziert werden. Moderne grafische Browser integrieren die referenzierten Inhalte optisch an der Stelle, an der das jeweilige Element notiert ist.

Dieses Konzept drückt allerdings auch eine Gewichtung aus: Grafik und Multimedia sind im W3C-Konzept von HTML optionale, aber keine prägende Features. Deshalb haben immer noch zahlreiche Webdesigner, die eine grafische Ausbildung
haben und „grafisch denken“, ihre Probleme mit dem textorientierten Strukturgedanken von HTML. Die Folge sind meist Webseiten, die nur wenig, meist lieblos hingerotzten und fehlerhaften oder von WYSIWYG-Editoren generierten HTML-Code enthalten
– und dieser besteht vorwiegend aus Grafik- und Flash-Referenzen. Denn das, was solche Webdesigner sich unter der „Message“ oder dem „Feeling“ vorstellen, die von Webseiten transportiert werden sollten, ist für sie nur über Grafiken und animierte
Effekte realisierbar.

> Über die Philosophie von HTML und die Berechtigung des Wunschs nach rein grafischen Webseiten

lässt sich vermutlich endlos streiten. Wer sich jedoch auf HTML einlässt, sollte die text- bzw. hypertextorientierte Grundausrichtung dieser Markup-Sprache akzeptieren. Wer komplett grafisch denken will, dem bleibt die Möglichkeit, anstelle von HTML-Seiten
gleich Flash-Movies auf seinen Webspace zu laden und gegebenenfalls den Webserver so zu konfigurieren, dass er z.B. auch Dateinamen wie `index.swf` als Default-Dateinamen erkennt und *swf*-Dateien mit einem entsprechenden
HTTP-Header ausliefert. Der aufrufende Browser bekommt dann beim Aufruf nichts anderes als ein Flashmovie gesendet, welches er – so er denn über ein geeignetes Flash-Plug-In verfügt – im gesamten Anzeigefenster darstellen kann. Es spricht
nichts gegen eine solche Radikallösung. Auch *Tim Berners Lee*, der Gründervater des Web, hat stets betont, dass das Web nicht auf bestimmte Dateiformate fixiert sei. HTML ist einfach eine Standardsprache für Webseiten, doch wenn Design-Vorstellungen
zu stark mit deren Konzepten konfligieren, ist ein Komplettverzicht auf **HTML** allemal besser und sauberer als die zahlreichen Sprachvergewaltigungen, die man im Web findet.

Geeignete Grafikformate

Es gibt in HTML keine Vorschrift dafür, welches Format eine eingebundene Grafik haben sollte. Darüber, welche Formate angesagt sind, haben letztlich die grafischen Browser entschieden, und zwar einfach dadurch, welche Formate sie in der Lage sind, ohne
Aufruf eines Fremdprogramms direkt anzuzeigen. Dabei haben sich in den 90er Jahren die beiden Pixel- oder Rastergrafikformate GIF und JPEG etabliert. Beide Formate zeichneten sich dadurch aus, dass sie gut komprimieren konnten und dass sie
– zumindest scheinbar – frei verwendbar waren. Beim GIF-Format erwies sich Letzteres allerdings als Trugschluss. Ende der 90er Jahre, als die ganze Welt das Ende der 60er Jahre entwickelte und zunächst vom Online-Dienst CompuServe benutzte,
unscheinbare GIF-Format für Millionen von Webgrafiken nutzte, startete der Lizenzinhaber des von GIF benutzten Kompressionsalgorithmus plötzlich eine Kampagne, um die Pfründe abzustecken. Software, die in der Lage war, GIF-Grafiken anzuzeigen
oder zu bearbeiten, sollte fortan Lizenzgebühren an den Lizenzgeber zahlen.

Aus diesem Schlamassel ging ein neues, lizenzfreies und offen dokumentiertes Format hervor,

das vor allem als Alternative zum GIF-Format entwickelt wurde, zum Teil aber auch mit dem JPEG-Format konkurriert: das *PNG-Format*. Von der „Szene“ im Netz wird dieses neue Format, welches ebenso wie HTML und CSS vom **W3-Konsortium** spezifiziert wird, als das neue Universal-Grafikformat gepriesen, und vor allem das GIF-Format gilt in solchen Kreisen mittlerweile als Igitt-Format. Um sich ein objektiveres Urteil zu bilden, sollte sich jeder Webdesigner mit den Grundeigenschaften
und Unterschieden der drei genannten Formate vertraut machen. Die nachfolgende Tabelle listet wichtige Faktoren und Unterschiede auf:

	PNG	**JPEG**	**GIF (89er-Format)**
Kompressionsart	verlustfrei	verlustbehaftet	verlustfrei
Kompressions-algorithmus	zlib-Technologie (lizenzfrei)	JPEG ist ein eigener Kompressionsalgorithmus	lzw-Technologie (lizenziert)
Interlacing	ja, optional	ja, optional („progressive“)	ja, optional
Transparenz	ja, echte Transparenz	nein	ja, jedoch nur einfarbige Transparenz
Animationsfähigkeit	nein	nein	ja
Max. Farbtiefe	bis zu 48 Bit (281 Trillionen unterschiedliche Farben), herunterskalierbar bis auf 8 Bit	24 Bit (16,7 Mio. unterchiedliche Farben pro Bild)	8 Bit (256 unterschiedliche Farben pro Bild)
Truecolor-Unterstützung	ja	ja, immer	nein
Paletten-Unterstützung	ja, optional	nein	ja, immer
Graustufen-Unterstützung	ja, bis zu einer Farbtiefe von 16 Bit (65536 Graustufen-Palette)	ja, mit einer Farbtiefe von 8 Bit (256 Graustufen)	ja, mit der Farbtiefe von 8 Bit (256 Graustufen)

Die meisten heute verbreiteten Grafikeditoren und Bildbearbeitungsprogramme für Pixelgrafiken unterstützen beim Abspeichern alle drei Formate. Achten Sie bei der Wahl eines Grafikprogramms aber auch darauf, welche Optionen es beim Abspeichern in diesen

Formaten anbietet. Denn viele letztlich ausschlaggebende Faktoren wie die Kompressionsoptionen, Interlacing und andere werden nicht während der Bildbearbeitung, sondern erst beim Abspeichern eingestellt. Speichern Sie wichtige Grafiken auf
jeden Fall auch in einem geeigneten Originalformat. Wenn Sie beispielsweise mit Photoshop oder Paint Shop Pro arbeiten, sollten Sie z.B. Grafiken, die auf der Arbeit mit Layern basieren, stets auch im programmeigenen Format als Sicherungskopie
speichern, damit Sie die Grafik zu einem späteren Zeitpunkt wieder optimal bearbeiten können.