Anatomie

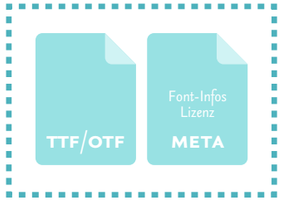
Auch bei der Formgebung einzelner Schriften und Schriftfamilien und der Zurichtung einzelner Schriftzeichen zueinander lässt sich eine Regelmäßigkeit und visuelle Geschlossenheit erkennen: Wiederkehrende Buchstabenformen und ein ausgewogenes Verhältnis zwischen Schwarz- und Weißräumen innerhalb einzelner Glyphen, Worte und Zeilen verleihen Schriften einen Rhythmus und sorgen für ein harmonisches Schriftbild. Zur genauen Einordnung und gezielten Beschreibung wurden Begriffe für bestimmte Merkmale von Schriften und Teilstücke von Buchstaben gefunden. Die wichtigsten werden in diesem Schaubild dargestellt, das auf Basis des Buches *Anatomie der Buchstaben* von Karen Cheng entstand. [1]

Webfont-Formate

Neben den herkömmlichen Font-Formaten existieren zwei weitere Formate, die speziell für den Einsatz im Web konzipiert wurden. Technisch handelt es sich bei beiden um keine gänzlich neuen Font-Formate, vielmehr wurden Arten von Containern für OpenType- und TrueType-Fonts entwickelt, um diese vor Installationen im System zu schützen.

EMBEDDED OPENTYPE (EOT) ist ein proprietäres Font-Format von Microsoft, das bereits Ende der 1990er Jahre entwickelt wurde. Das EOT-Format gibt es neben der klassischen Variante auch als EOT *lite*. EOT-lite-Fonts unterscheiden sich insofern, als dass sie an bestimmte Domains gebunden und komprimiert werden können, um eine Nutzung auf anderen Webseiten zu verhindern und die Dateigröße zu minimieren. EOT und EOT lite werden von keinem anderen Browser außer dem *Internet Explorer* unterstützt.

WOFF
Containerformat



WEB OPEN FONT FORMAT Das *Web Open Font Format* wurde im Zuge der Wiederentdeckung der `@font-face`-Regel im Jahr 2010 entwickelt und bietet die Möglichkeit, zusätzliche Metadaten wie Hersteller- und Lizenzinformationen in den Container einzuschließen. WOFF gilt heute als Standard-Font-Format für Webfonts und wird von allen aktuellen Browsern unterstützt.

Scalable Vector Graphics

Scalable Vector Graphics (SVG) ist – wie der Name schon vermuten lässt – kein Schriftformat, sondern dient der Beschreibung von Vektorgrafiken. Es wird im Webfont-Kontext dazu genutzt, die Konturlinien der Glyphen zu bestimmen, die vom Browser als gefüllte Flächen in Form der Schriftzeichen dargestellt werden können. Schriften, die mittels SVG-Technik wiedergegeben werden, weisen oftmals ein unregelmäßiges Schriftbild auf, da Laufweiten- und Kerninginformationen beim Konvertieren verloren gehen und sollten daher eher vermieden werden.

Zeichensatz und -kodierung

Zeichensatz

Abhängig davon, wie gut eine Schrift ausgebaut ist, kann ihr Zeichensatz z. B. nur die 26 Großbuchstaben des lateinischen Alphabets oder bis zu mehreren Tausend Zeichen verschiedener Sprachen und unterschiedlicher Stilvarianten umfassen. Um die Dateigrößen im Web möglichst gering zu halten, werden oft *Subsets* von Fonts angeboten, die nur einen ausgewählten Teil der verfügbaren Glyphen beinhalten.

Zeichenkodierung

Da unsere Computer und wir grundsätzlich nicht dieselbe Sprache sprechen, ist eine Übersetzung zwischen der für uns lesbaren Schriftzeichen und den für den Computer verständlichen Nullen und Einsen notwendig. Dies erfolgt durch ein sogenanntes *Zeichenkodierungsschema*, in dem einem Schriftzeichen eine eindeutigen Folge aus Nullen und Einsen (sogenannte Bitfolgen [A]) zugewiesen ist. Diese Zuweisungen werden in einer je nach Kodierung variierenden *Übersetzungstabelle* festgehalten. Die Menge der in der Tabelle enthaltenen Zeichen wird *Zeichensatz* genannt. Da viele Kodierungsschemata nur eine stark begrenzte Anzahl an Zeichen beinhalten konnten und untereinander inkompatibel waren, weil sie z. B. verschiedenen Zeichen gleiche Bitfolgen zuwiesen, wurde Unicode [B] entwickelt. Unicode ist ein stetig wachsender Zeichensatz, der 1991 veröffentlicht wurde und alle weltweit bekannten Schriftzeichen umfasst. Dabei wird jedem Zeichen ein individueller *Kodierungspunkt* zugewiesen. Unicode bietet Platz für über eine Million Zeichen, von denen derzeit etwa 100 000 belegt sind. Für Unicode existieren drei Kodierungsschemata: UTF-8, UTF-16 und UTF-32 [C]. Im Web konnte sich UTF-8 als Standard etablieren, da es abwärtskompatibel zu der zuvor weit verbreiten ASCII-Zeichenkodierung [D] ist und besonders für europäische Sprachen weniger Bits verbraucht als die anderen beiden Kodierungen.

[A] Bitfolge

Die Bitfolge 00100001 entspricht bspw. dem Ausrufezeichen und 00111111 steht für das Fragezeichen

[B] Unicode

Unique universal and uniform character encoding

[C] UTF-X

Universal Character Set Transformation Format; die anschließende Zahl steht für die minimale Anzahl an Bits, die zur Kodierung eines Schriftzeichens aufgewendet werden muss.

[D] ASCII

American Standard Code for Information Interchange

Statische und flexible Raster

[A] Liquid Layout

Layout passt sich stufenlos der Bildschirmgröße an

[B] Adaptive Layout

Bestimmte Anzahl an in der Breite festgelegten Layouts, die je nach Bildschirmgröße aufgerufen werden

LIQUID UND ADAPTIVE LAYOUTS Im Kontext von Responsive Webdesign existieren verschiedene Ansätze bzgl. der zugrundeliegenden Raster: Zum einen gibt es die sogenannten Liquid Layouts [A] (etwa: *Fließende Layouts*), die sich jeder Bildschirmgröße stufenlos anpassen, d. h. die verfügbare Breite eines Bildschirms wird – zumindest bei kleineren Bildschirmen – immer maximal ausgenutzt. Daneben gibt es sogenannte Adaptive Layouts [B], die eine bestimmte Anzahl an Layouts mit unterschiedlichen, aber jeweils festen Breiten beinhalten, die je nach Bildschirmgröße angezeigt werden.

An welchen Stellen sich das Layout verändert, wird mittels sogenannter *Breakpoints* (etwa: *Umbruchspunkte*) definiert. Breakpoints sollten sich immer ausschließlich an dem Inhalt und der Struktur der Website orientieren, nicht an speziellen Geräten und deren Spezifikationen. Leidet zum Beispiel die Lesbarkeit eines Textes aufgrund zu kurzer Zeilenlängen oder ist die Hauptnavigation zu breit für eine zu schmale Spalte, sollte über eine dem Inhalt entsprechende Umverteilung nachgedacht werden.

Technisch gesehen ist der Hauptunterschied, dass fließende Layouts auf flexiblen Rastern basieren, adaptive Layouts hingegen auf mehreren statischen Rastern mit unterschiedlichen Breiten aufbauen. Flexible Raster sind insofern komplexer zu handhaben, als dass bei ihrer Verwendung eine lückenlose Reihe an funktionierenden Layout-Lösungen von breit bis schmal gewährleistet werden muss, wohingegen beim Einsatz von statischen Rastern in der Regel lediglich drei oder vier verschiedene fixe Layoutvarianten für verschiedene Bildschirmgrößen erstellt werden müssen.

Beide Ansätze haben Vor- und Nachteile: Fließende Layouts ermöglichen die optimale Ausnutzung des verfügbaren Platzes, was insbesondere bei kleinen Bildschirmen wichtig ist, während beim Einsatz adaptiver Layouts potentiell Platz durch leere Ränder eingebüßt wird. Infolge von Rundungsfehlern kommt es bei flexiblen Rastern jedoch oft zu Ungenauigkeiten in der Darstellung, die je nach Design mal mehr und mal weniger ins Auge fallen. Bei statischen Rastern ist eine genauere Kontrolle über das Layout aufgrund mehrerer festgelegter Breiten möglich. Fließende Layouts gehen im Vergleich zu adaptiven Layouts zumeist mit einem höheren Konzeptions- und Umsetzungsaufwand einher.

Da Vor- und Nachteile sich nahezu die Waage halten, kann schwierig beantwortet werden, welcher Ansatz der bessere ist. Neben pragmatischen Aspekten wie Budget und Zeitfenster sollte bei der Entscheidung, auf welchen Ansatz zurückgegriffen wird, vor allen Dingen das Layout selbst im Vordergrund stehen.

Responsive Layouts werden mithilfe von *Media Queries* erstellt, die bestimmte Eigenschaften eines Gerätes auslesen können. In den meisten Fällen wird hierzu die Breite des Geräts unter Berücksichtigung der Device Pixel Ratio [C] bzw. die Breite des Browserfensters abgefragt, infolge dessen dann der zutreffende Teil des Stylesheets angewandt wird.

Media Queries

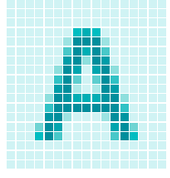
```
@media screen and (max-width: 500px) {
}  ···
@media screen and (min-width: 900px) {
}  ···
```

[C] Device Pixel Ratio

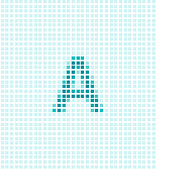
Da Smartphones und Tablets aufgrund ihrer hohen Pixeldichten mittlerweile über ähnliche Auflösungen wie Laptop- und Computerbildschirme verfügen, die Pixel jedoch im Vergleich zu durchschnittlich aufgelösten Bildschirmen deutlich kleiner sind, wurde eine Angleichung der Pixelgröße von hochauflösenden Geräten in Bezug auf durchschnittlich aufgelöste Bildschirme vorgenommen.

Der Grund ist, dass bspw. eine 12 Pixel große Schrift auf einem durchschnittlich aufgelösten Bildschirm problemlos lesbar ist, auf einem hochauflösenden Screen, dem auf einer vergleichbaren Fläche anstatt einem Pixel unter Umständen vier Pixel zur Anzeige der Schrift zur Verfügung stehen, würde die Schrift nur halb so groß und womöglich kaum mehr lesbar dargestellt.

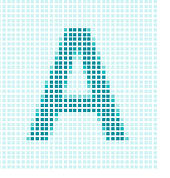
Um diesem Effekt entgegenzusteuern, wird von hochauflösenden Geräten eine Art Größenverhältnis mitgeliefert, inwieweit die Größe der Pixel von der eines herkömmlichen Pixels abweicht. Diese Zahl nennt sich *Device Pixel Ratio*. Die in diesem Beispiel verwendete Schriftgröße von 12 Pixel entspricht daher auf einem hochauflösenden Gerät mit einer Device Pixel Ratio von 2 24 Pixel (12 Pixel × 2).



Herkömmlicher Screen
mit 320 × 480 Pixel



Hochauflösender Screen
mit 640 × 960 Pixel
ohne Berücksichtigung der Pixel Ratio



Hochauflösender Screen
mit 640 × 960 Pixel
mit Berücksichtigung einer Pixel Ratio von 2