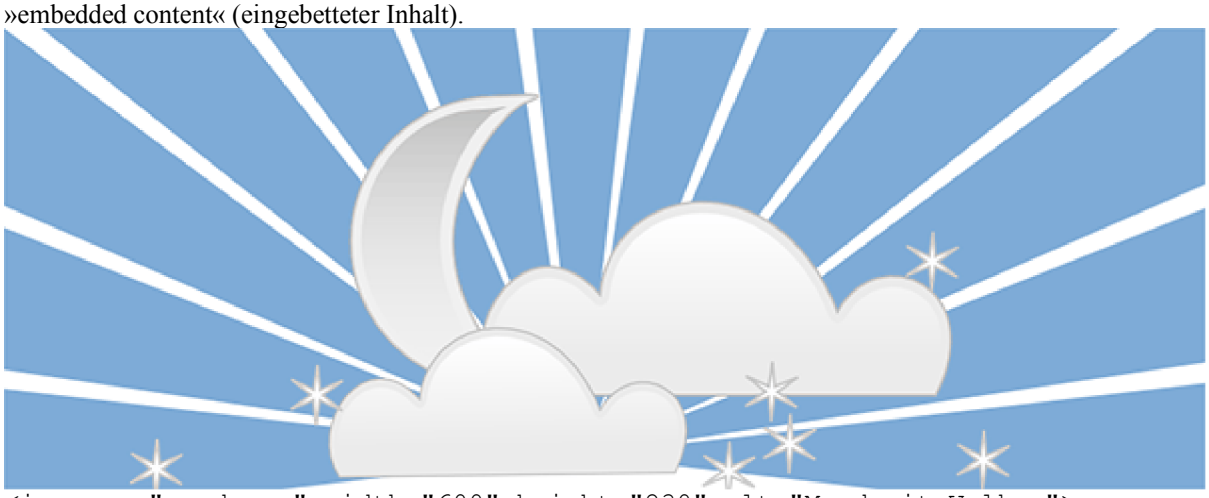
HTML img

HTML img fügt ein Bild oder Grafik in HTML-Seiten ein. Nur zwei Attribute werden gebraucht: *src*, der Pfad zum Bild und *alt*, ein alternativer Text für Browser ohne Bilddarstellung.
Browser können nur wenige Bild- oder Grafikformate zuverlässig darstellen. JPEG, GIF oder PNG sind die sichersten Formate für die Anzeige von Bildern in Webseiten. AB HTML5 kann SVG mit dem img-Tag eingebettet werden.
img fügt das Bild nicht in die Webseite ein, sondern ist ein Platzhalter und bildet einen Link zum Bild – Bilder sind »embedded content« (eingebetteter Inhalt).



```
<img src="mond.png" width="600" height="230" alt="Mond mit Wolken">
        |                   |           |           |
        |                   |           |           +--- Alternativer Text
        |                   |           |
        |                   |           +--- Höhe in Pixel
        |                   |
        |                   +--- Breite in Pixel
        |
        +--- src-Attribut mit Pfad zum Speicherort des Bildes
```

Das img-Tag bewirkt keinen Zeilenumbruch, sondern erzeugt ein Inlineelement. Das img-Tag kann in ein a-Tag gesetzt werden, um einen klickbaren Link zu erzeugen. Die Browser legen bei einem Bild innerhalb eines a-Tags einen Rahmen um alle vier Seiten, um das Bild als Link zu kennzeichnen. Der CSS-Stil *a img { border: none; }* entfernt den Rahmen.
Das img-Tag ist leer. Mit XHTML wurde ein Schrägstrich das Ende des Elements gesetzt werden: . In HTML5 kann der Schrägstrich beim img-Tag wegfallen.
Mit HTML5 figure und figcaption erhält das Bild einen Container, um es vom Text abzuheben, zu positionieren und eine Bildunterschrift oder Legende hinzuzufügen.

img Tag – Attribute

src
ist die URL der Bilddatei und muss immer gesetzt werden. Die URL muss den Dateinamen enthalten und kann einen relativen oder absoluten Pfad angeben.
alt
ist ein Text, der anstelle des Bildes von Browsern gezeigt wird, die Bilder oder Grafiken nicht darstellen können. Das alt-Attribut muss immer gesetzt werden.
crossorigin html5
regelt, ob und welche fremden Seiten per Skript (z.B. AJAX) direkt auf das Bild zugreifen können
height
Höhe des Bildes
ismap
ist ein boolescher Wert und kennzeichnet, dass es sich bei dem Bild um eine klickbare serverseitige Image Map handelt.
longdesc
ist die URL einer Quelle, die eine ausführliche Beschreibung des Bildes enthält. Das ist nützlich für serverseitige Image Maps und nichtgrafische Browser.
sizes html5
bestimmt, welchen Raum das Bild innerhalb des Layouts einnimmt – z.B. 33% der Breite des Viewports bei einem Spaltenlayout.
srcset html5
Liste von alternativen Versionen / Größen des Bildes für unterschiedliche Monitore.
usemap
Namen einer Image Map, die mit dem Element verknüpft werden soll. Es handelt sich hierbei um eine klickbare clientseitige Image Map.
width
Breite des Bildes
Die alten Attribute *align*, *border*, *hspace* und *vspace* sind nicht mehr in HTML5 vertreten.

Bildgröße

width und *height* sind auch in HTML5 weiterhin Attribute für das img-Tag und werden nicht ins CSS verbannt. Die Browser nutzen die Angabe *width* und *height* für einen Platzhalter und bauen die Seite schon auf, während das Bild noch geladen wird.
Die Angaben *width* und *hight* müssen auch nicht die physikalische Pixelauflösung des Bildes angeben. Das Bild kann kleiner oder größer sein und wird für responsive Webseiten mit *CSS width:100% height:auto* an den umfassenden Block angepasst.

img srcset – Alternative Bildgrößen

Hoch auflösende Bilder mit ihren langen Ladezeiten sind ein Handycap für responsive Webseiten auf mobilen Geräte. Auf der einen Seite wollen wir eine hohe Qualität für große Monitore und Retina-Displays, auf der anderen Seite wollen wir Mobiltelefone und Tabletts nicht mit den großen Bildern erschlagen: Bandbreite bleibt kostbar.
Das srcset-Attribut des img-Tags erfasst individuelle Bilder für die unterschiedlichen Viewports der mobilen Geräte, für große Desktop-Monitore und Retina-Monitore.

```
<img alt="medium" srcset="medium.jpg 1x, large.jpg 2x">
```

Hier stehen also zwei Bildgrößen zur Verfügung, die je nach Auflösung des Monitors geladen und angezeigt werden. Der Browser entdeckt die Auflösung / Pixeldichte des Monitors und entscheidet, welches Bild er einsetzt.
1x bzw. 2x steht für *Device Pixel Ratio* – das Verhältnis [Anzahl der Pixel in der Breite] zu [Breite in Zoll].

- 1x steht für herkömmliche Monitore, die mit 72 bis 100 Pixeln pro Zoll auflösen,
- 2x steht für hochauflösende Monitore, die mit 200 Pixeln und höher pro Zoll auflösen – sogen. Retina-Monitore.

Es gibt auch Displays, deren Pixeldichte zwischen 1x und 2x liegen. Also wäre auch ein device-pixel-ratio von 1.5x eine denkbare Media Query auf die Auflösung des Monitors.

Device	Pixel per Inch (PPI)	Bildschirmauflösung (Pixel pro Zoll)
iPhone 6 Plus (6s Plus)	401	1920 × 1080
iPad 4. Generation) iPad Air 1./2. Generation) iPad Pro (9,7″)	264	2048 × 1536
iPad Pro (12,9″)	264	2732 × 2048
MacBook Pro – Retina Display (15")	220	2880 × 1800

Der Browser wird das 2x-Bild automatisch um den Faktor 2 verkleinert anzeigen, damit es denselben Raum ausfüllt wie das das Bild mit der 1x-Auflösung. Dieses Verhalten ähnelt der Angabe der DPI bei Bildern für den Druck. Der Zeitschriften- und Buchdruck arbeitet i.d.R. mit 300 dpi, während Bilder auf dem Monitor mit 72dpi angezeigt werden.
Alle modernen Browser außer IE bis einschließlich IE11 (diese Seite benutzt Picturefill-Polyfill für Browser ohne Support für srcset und picture-Element).
Allerdings: Die Angabe der Bildschirmauflösung als Kriterium führt dazu, dass nicht nur auf großen Retina-Monitoren wie dem MacBook Pro mit Retina Display, sondern auch auf kleinen Monitoren – z.B. Handy – die 2x-Auflösung angezeigt wird. Hier ist also eine weitere Differenzierung nötig.
Das Bild mit der zweifachen Auflösung soll nur geladen werden, wenn der Platz im Layout verfügbar ist.
Die Kondition für den Einsatz verschiedener Bildgrößen drei Parameter einbeziehen:

1. Bildgröße relativ zum Viewport (z.B. 33.3vw)
2. Breite des Viewports z.B. 640px oder 40em
3. Auflösung des Monitors 1x oder 2x

Was jetzt noch fehlt ist eine Mitteilung an den Browser, wie groß das Bild im Layout dargestellt werden soll – dafür gibt es das Attribut *sizes*.

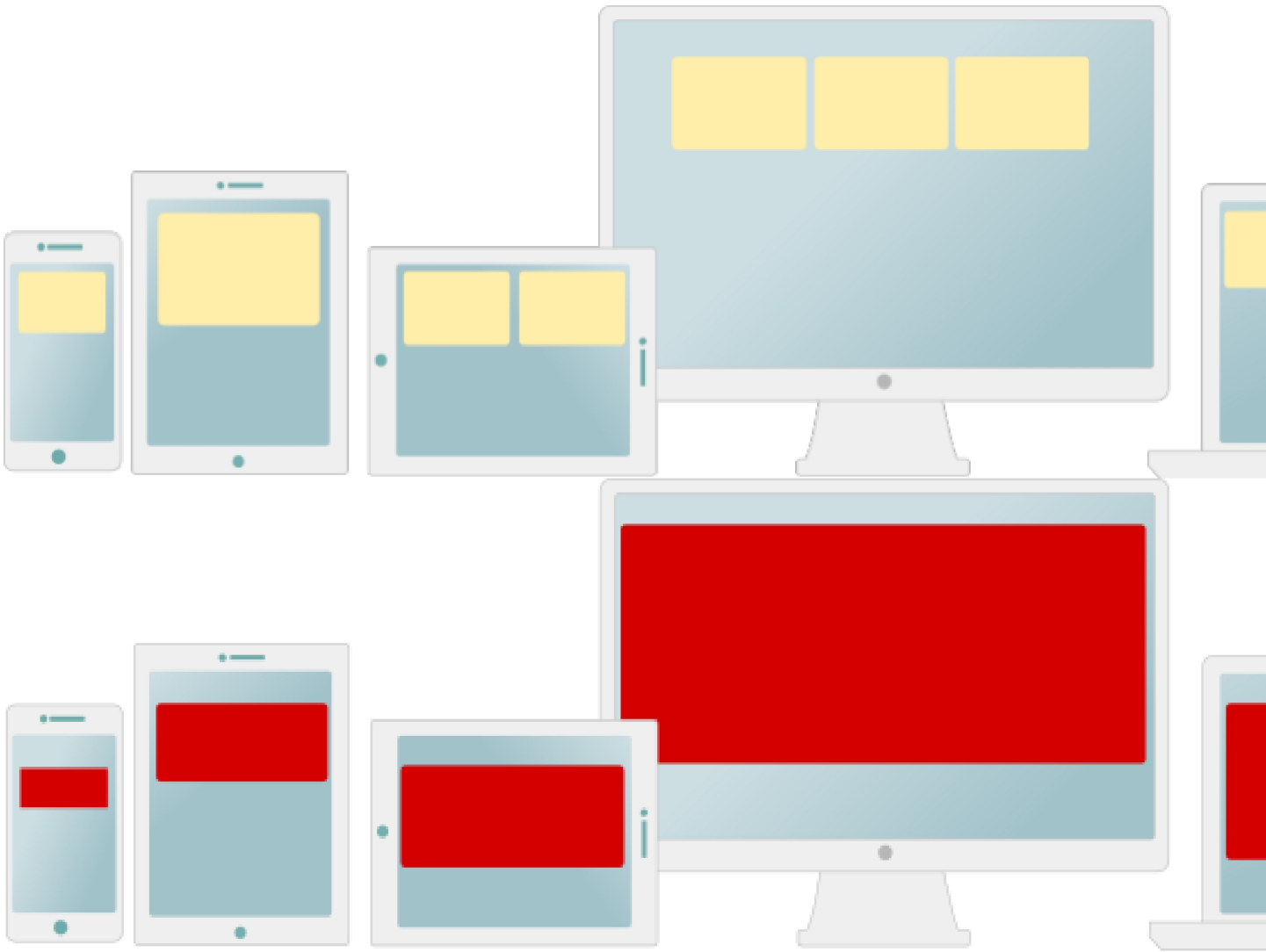
```
sizes = [media query] [length], [media query] [length], …

sizes = "(min-width: 42.5em) 50vw,100vw"
```

sizes legt den Raum fest, den das Bild innerhalb des Layouts einnimmt. Dann enthält *srcset* eine Liste der Bilder und ihrer jeweiligen Größe. Mit diesen Informationen wählt der Browser das kleinste Bild, das in das Layout passt und es ausfüllt.
Mit *sizes* und *srcset* bestimmt also nicht mehr die Größe des Viewports allein über die Wahl des Bildes.

```
                              1                          2              3
                              +                          +              +
                              |                          |              |
                              v                          v              v
<img sizes="(max-width:720px) 100vw, (max-width: 1260px) 70vw, calc(50vw -
100px)"
   srcset="picture-flowers-300.jpg 300w,
          /xhtml/img/picture-flowers-640.jpg 640w,
          /xhtml/img/picture-flowers-720.jpg 720w,
          /xhtml/img/picture-flowers-960.jpg 960w,
          /xhtml/img/picture-flowers-1120.jpg 1600w"
   src="/xhtml/img/picture-flowers-640.jpg"
   alt="Aquarellmalerei Blumen mit Goldsplitter">
```

1. (max-width:720px) 100vw – bis zu einer Breite des Viewports von 720px wird das Bild in voller Breite angezeigt
2. (max-width:720px) 100vw – bei einer Breite des Viewports bis zu 1260px tritt ein Spaltenlayout in Kraft und das Bild liegt in einer Spalte von 70% der Breite des Viewports.
3. calc(50vw - 100px) – bei noch größeren Viewports liegt das Bild in einer Spalte von 50% der Viewport-Breite (minus 100px Abstand zwischen den Spalten)

Natürlich müssen nicht gleich so viele Bildgrößen wie in diesem Beispiel angelegt werden. Hier haben wir nur eine Demo.



Wenn anstelle eines Bildes mit gleichem Ausschnitt und gleichen Seitenverhältnissen Bilder mit unterschiedlichen Ausschnitten oder in unterschiedlichen Formaten gezeigt werden sollen – z.B. ein quadratischer Ausschnitt für Handys im Hochformat –, dann ist ein HTML picture-Element u.U. die bessere Wahl als img mit sizes und srcset.

Bilder in Textabsätzen

Ohne CSS verhält sich ein Bild in einem Text immer wie ein einzelnes Zeichen im Text: Das Bild sitzt auf der Grundlinie der Zeile und dehnt die Textzeile nach oben aus.

```
<p> <img src="mond.png" style="float: left;" alt="" width="180" height="123">
   Ein Bild mit Mond neben einem Text</p>
```

Ein Bild mit Mond neben einem Text

Erst durch die CSS-Eigenschaft *float: left;* umfließt der Text das Bild.

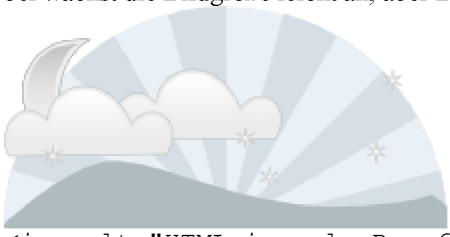
```
<img src="mond.png" style="float: left;" alt="" width="180" height="123">
```

Ein Bild mit Mond in einem Text

Das war ein beliebter Ansatz für Bilder in längeren Textabsätzen. Auf kleinen Monitoren sollte der Text automatisch unter das Bild wandern.

Bild mit Base64 einbetten

Wenn die Bilder klein sind, kann es sinnvoll sein, sie im Base64-Format direkt in das HTML-Markup einzubetten. Dabei wächst die Bildgröße leicht an, aber Base64-Bilder sparen einen HTTP-Request beim Laden der Seite.



```
<img alt="HTML img als Base64 einbetten"
    src=" … >
```

HTML picture per source

HTML picture liefert zusammen mit dem srcset-Attribut und dem media-Attribut ein passendes Bild an den Browser des Benutzers. Als Konditionen kommen z.B. die Auflösung des Monitors, die Größe des Viewports oder die Ausrichtung des Geräts in Frage.
Gegenüber dem img-Tag, dass mit srcset und sizes unterschiedliche Auflösungen (z.B. 1x oder 2x- Retina-Display) und das Platzangebot des Layouts unter einen Hut bringt, bietet HTML picture dem Browser unterschiedliche Dateiformate und Seitenverhältnisse.
Das picture-Element funktioniert ähnlich wie ein HTML video- oder audio-Tag und braucht nur wenig zusätzliches HTML-Markup, sondern nutzt Media Queries. Damit kann picture unterschiedliche Bildgrößen oder Bilder mit anderen Seitenverhältnissen je nach Auflösung des Monitors einsetzen. Außerdem listet das source-Tag innerhalb des picture-Elements alternative Bildformate für Browser, die das bevorzugte Format des Bildes nicht unterstützen.
So kommen Handys und Tabletts im Hochformat (Portrait) einfach an einen aussagekräftigen Bildausschnitt bei Slideshows.

```
<picture>
   <source media="(min-width: 1440px)" srcset="pic-920.jpg 1x, pic-1840.jpg 2x">
   <source media="(min-width: 680px) and (orientation:portrait)" srcset="pic-
960-quad.jpg">
```

```
   <source media="(min-width: 680px)" srcset="html-pic-920.jpg">
   <source media="(min-width: 300px) and (orientation:portrait)" srcset="pic-
600-quad.jpg">
   <source media="(min-width: 300px)" srcset="pic-600.jpg">
   <img src="pic-960-quad.jpg" width="960" height="670" alt="HTML picture-Tag">
</picture>
```

1x bzw. 2x steht für *Device Pixel Ratio* – das Verhältnis [Anzahl der Pixel in der Breite] zu [Breite in Zoll].

- 1x steht für herkömmliche Monitore, die mit 72 bis 100 Pixeln pro Zoll auflösen,
- 2x steht für hochauflösende Monitore, die mit 200 Pixeln und höher pro Zoll auflösen – sogen. Retina-Monitore.

Natürlich bleiben die Breakpoints selbst für ein derart detailliertes Bilderpaket Kompromisse.

Picture Element und source-Tag

- Im picture-Tag liegen source-Elemente, am Ende folgt ein img-Tag als Fallback für Browser, die keine Support für picture bieten.
- Jedes source-Element hat ein srcset-Attribut, in dem ein oder mehrere Pfade zu Bildern notiert sind.
- Das media-Attribut enthält Media Queries.
- Anstelle oder zusätzlich zum media-Attribut kann ein sizes-Attribut das jeweilige srcset auswählen.

Der Browser nimmt das erste source-Element, dessen media-Attribut zutrifft – also werden bei Media Queries mit *min-width* die source-Optionen mit den größeren Auflösungen zuerst aufgeführt.
Bei Media Queries mit *max-width* beginnt die Source-Liste bei den kleineren Auflösungen und endet mit der höchsten Auflösung.
Anders als video- oder audio-Tags zeigt das *picture* Element selber nichts an, sondern der Browser wählt nur eine Alternative aus der source-Liste.

Alternative Ausschnitte / Dateiformate

Mit HTML *picture* und *srcset* kann nicht nur die Auflösung des Monitors durch eine Media Query festgestellt werden, sondern wir können den Browsern auch unterschiedliche Bildformate vorsetzen.

Statt die Bilder der Slideshow im Kopf der Seite auf kleinen Monitoren als schmalen Streifen zu setzen, ist ein Ausschnitt in einem eher quadratischen Format angemessener.

```
<picture>
   <source media="(min-width: 64em)" srcset="lemon-960.jpg 1x, lemon-1920.jpg
2x">
   <source media="(min-width: 42.5em) and (orientation:portrait)" srcset="lemon-
quad-960.jpg">
   <source media="(min-width: 20em) and (orientation:landscape)" srcset="lemon-
960.jpg">
   <source media="(min-width: 20em) and (orientation:portrait)" srcset="lemon-
quad-600.jpg">
   <img src="lemon-quad-960.jpg" >
</picture>
```

Wer Format und eine angemessene Auflösung der Bilder sehr ernst nimmt, kann mit einem Picture-Pack aus sechs unterschiedlichen Auflösungen und Seitenverhältnissen eine große Spannbreite von Geräteklassen abdecken.

```
<source
    media="(min-width: 90em)"
    srcset="slide-2880.jpg" />
```

```
<source
    media="(min-width: 64em)"
    srcset="slide-1440.jpg" />
```

```
<source
    media="(min-width: 42em) and (orientation:portrait)"
```

```
    srcset="slide-960-quad.jpg" />
```

```
<source
    media="(min-width: 42em)"
    srcset="slide-960.jpg" />
```

```
<source
    media="(min-width: 20em) and (orientation:portrait)"
    srcset="slide-600-quad.jpg" />
```

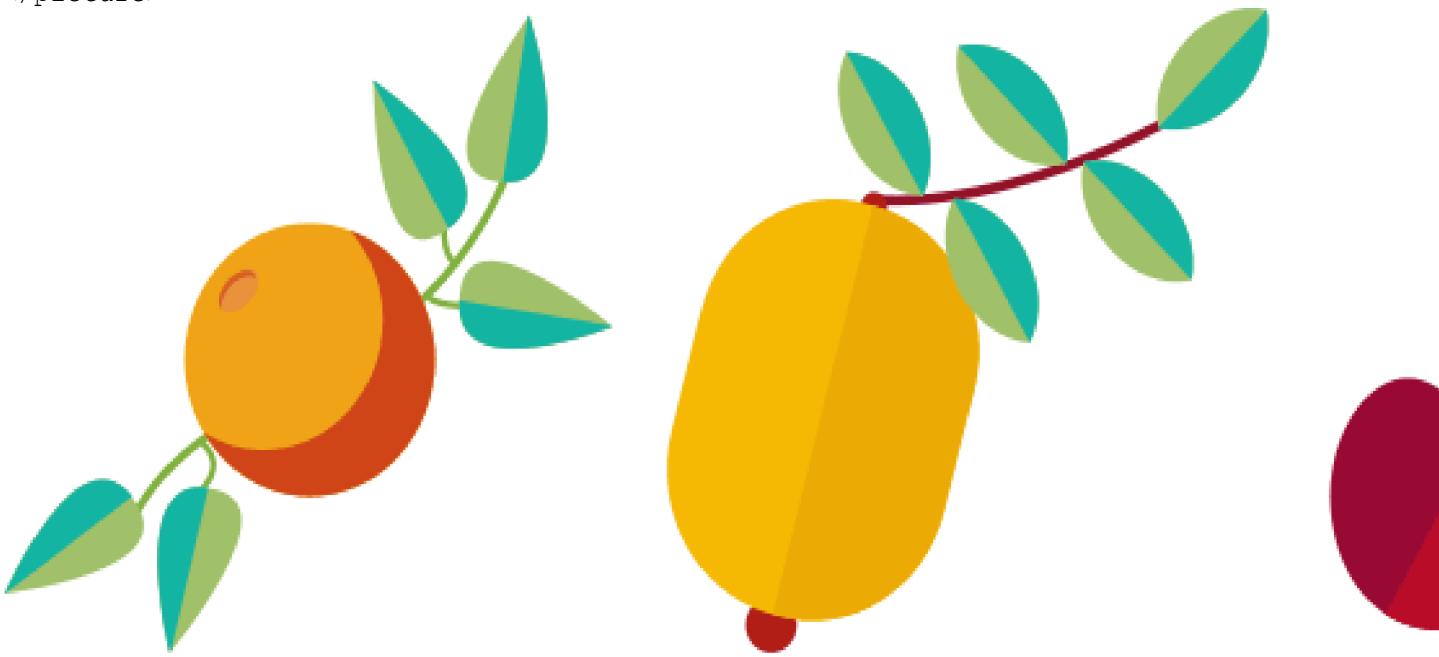
```
<source
```

```
    media="(min-width: 20em) and (orientation:landscape)"
    srcset="slide-600.jpg" />
```
Derartige Picture-Packs und das HTML können vom Content Management System automatisch erzeugt werden, aber einen sinnvollen Ausschnitt wählt man besser im Bildbearbeitungsprogramm.

picture für unterschiedliche Bildformate

Eine weitere Anwendung für das picture-Tag ist SVG – Scalable Vector Graphic, das Internet Explorer erst ab Version 9 erkennt. Browser, die den Mime-Typ des Bildes im source-Tag nicht unterstützen, überspringen das source-Element und zeigen das Bild im img-Tag.

```
<picture>
    <source srcset="fruity-icons.svg" type="image/svg+xml">
    <img src="fruity-icons.png" alt="fruity-icons" />
</picture>
```



Blaue Blätter? Das ist ein SVG. Grüngelbe Blätter? Ein PNG.
Ein relativ junges Dateiformat ist WebP, das Fotos, aber vor allem Bilder mit flachen Farben effizienter komprimieren kann als JPEG. WebP wird zurzeit nur von Google Chrome und Opera unterstützt. Mit dem picture-Tag können sich Browser, die Support für WebP bieten, das kleinere WebP-Bild aussuchen. Alle anderen Browser können sich an JPEG halten.

```
<picture>
   <source srcset="DSC07263.webp" type="image/webp">
   <img src="DSC07263.jpg" width="100%" alt="HTML picture-Tag mit JPEG und
WebP">
</picture>
```

picture oder img sizes und srcset?

Wenn angemessene Bilder je nach Größe und Auflösung des Viewports sowie nach dem Platzangebot des Layouts geladen werden sollen, reicht im Grunde genommen das img-Tag mit der Kombination sizes und srcset.

Schöner Trick von **Alexey Ten**, der Bilder gleichzeitig als PNG für ältere Browser und als SVG für alle anderen in HTML einbindet: Eine SVG-Grafik wird innerhalb eines SVG-Elements mit dem SVG image-Tag geladen.
Browser ohne SVG-Support halten das image-Tag für ein img-Tag und holen sich ein Bitmap-Bild.
Eigentlich ist das SVG-image-Tag dazu bestimmt, Bitmap-Bilder in eine SVG-Grafik zu laden. Aber das xlink:href-Attribut lädt nicht nur Bitmap-Bilder, sondern auch SVG.

```
<svg width="96" height="96">
    <image xlink:href="svg.svg" src="svg.png" width="96" height="96"/>
</svg>
```

Browser ohne SVG-Support überlesen das SVG-Tag, halten *image* für einen Schreibfehler und greifen gnädig zum img-Tag.
Mit einem xlink:href-Attribut rendern SVG-fähige Browser die SVG-Grafik und überlesen das src-Attribut. Browser ohne SVG-Unterstützung zeigen das Bildmap-Bild im src-Attribut.

Links: IE8, rechts: Safari
Noch ein weiterer Schritt, damit sich das Bitmap-Bild und die SVG-Grafik an den Platz anpassen und responsive werden:

SVG responsive

```
<div class="svg-frame">
   <svg width="100%" height="100%" viewbox="0 0 600 493" preserveAspectRa-
tio="xMinYMin meet">
      <image xlink:href="camera.svg" src="camera.png" alt="" width="600"
height="493" />
   </svg>
</div>
```

Im svg-Tag ersetzt das viewBox-Attribut die Angabe von Breite und Höhe durch *width* und *height*. Dann passt sich die SVG-Grafik responsive an die Maße des umfassenden Containers an.

```
.svg-frame {
   width:300px;
   position: relative;
   display:inline-block;
   padding-bottom: 50%;     // wird 100%, wenn die Grafik
                            // höher als breit ist
   vertical-align: middle;
   overflow:hidden }
.svg-frame svg { position: absolute; }
image { width:100%; height: auto; }
```

Die blaue Rakete ist das Bitmap-Bild, die grüne ist eine SVG-Grafik.

SVG in Facebook, Pinterest, Google+ und Twitter

Jetzt ist IE8 zwar so gut wie außer Betrieb, aber auch nach IE8 bleibt ein wichtiger Bereich des Webs von SVG unberührt: Facebook und Pinterest, Google+ und Twitter leugnen die Existenz von SVG auch weiterhin. Ein Zweibild als Bitmap ist auch in den nächsten Jahren das Beiwerk, um den sozialen Medien die Grafiken näher zu bringen. Der »SVG-in-einem-image-Tag-Trick« bringt das nicht zustande.

Ein Ausweg für Google+ und Facebook ist ein Open-Graph-Metatag.

```
<meta property="og:image" content="https://www.mediaevent.de/wp-con-
tent/uploads/2014/03/svg-in-einem-image-tag.png" />
```

HTML picture öffnet dem SVG ebenfalls den Weg zu Google+ und Facebook (und nebenbei auch zu alten Browsern ohne SVG). Browser, die SVG nicht unterstützen, überspringen das source-Element.

```
<picture>
    <source srcset="fruity-icons.svg" type="image/svg+xml">
    <img src="fruity-icons.png" alt="fruity-icons" />
</picture>
```

SVG in WordPress laden

Am Ende wird der Media Upload in WordPress noch SVG-ready, wenn die Datei functions.php um SVG erweitert wird. Dann kann die WordPress Mediathek auch mit SVG beladen werden.

```
function cc_mime_types( $mimes ){
    $mimes['svg'] = 'image/svg+xml';
    return $mimes;
}
add_filter( 'upload_mimes', 'cc_mime_types' );
```

Allerdings kann der WordPress-Media Upload die Größe der SVG-Grafik nicht bestimmen und setzt width="1″ height="1″. Auf die HTML-Ansicht umschalten und die gewünschte Größe einstellen.
Damit der WordPress-Editor HTML5-Tags wie svg, figure, …) nicht auslöscht, muss der wpautop-Filter in der Datei functions.php des Themes deaktiviert werden.

```
remove_filter('the_excerpt', 'wpautop');
remove_filter('the_content', 'wpautop');
```

Vor gut 15 Jahren hat die Entwicklung von SVG als Standard begonnen. Jetzt dürfen wir uns endlich die Einzelteile aus einem Puzzle zusammentackern.
Wenn SVG-Grafiken dann auf der Seite immer noch nicht angezeigt werden, ist die Konfiguration des Servers angestaubt oder wurde von einem Spartaner angelegt: Der Mime-Type SVG fehlt. In der htaccess-Datei nachholen:

```
AddType image/svg+xml svg
AddType image/svg+xml svgz
```

SVG einbetten

==========

Die Clous von SVG sind das nahtlose Einbetten von SVG-Grafiken ins HTML der Webseite und die Integration in altgediente Techniken des Webdesigns. Wir können SVG inline in HTML einbetten, mit einem HTML img-Tag laden oder als Hintergrundbild in CSS einsetzen.
Früher habe ich die Grafiken für Vorlesungen und Webseiten mit Photoshop angelegt und als PNG exportiert. Bei jeder Änderung muss das Original – sei es nun ein TIFF in Photoshop oder eine SVG-Grafik aus Inkscape oder Illustrator – wieder hervorgekramt und geändert werden. Dann die Grafik wieder als PNG oder GIF exportieren, erneut in die Webseite einbinden … das nervt …
SVG für Grafiken ist flexibler als GIF- oder PNG-Bilder, kann mit PHP dynamisch erzeugt werden (z.B. für Statistiken und Graphen) und erlaubt obendrein Links aus der Grafik. Und nicht zu vergessen: Javascript kann in die Grafik eingreifen und die Tortengrafik in Bewegung setzen, SVG kann mit CSS animiert werden und hat ein eigenes Animationsmodell.

1. SVG mit dem object-Tag einbinden,
2. SVG im src-Attribut des img-Tags einsetzen,
3. SVG direkt in das HTML-Markup schreiben,
4. als Hintergrundbild via CSS einsetzen,
5. via iframe einsetzen,
6. mit SVG image-Tag
7. SVG direkt in die CSS-Datei einbetten

1. SVG im img-Tag

Ohne großen Aufwand wird eine einfache SVG-Grafik mit einem img-Tag in der Webseite angezeigt.

```
<img src="on-the-beach.svg" alt="SVG mit img Tag laden" width="600" height="350"
>
```

- **Vorteile**
  Das img-Tag bietet die bekannteste Syntax. Der WordPress-Editor setzt das SVG als img-Tag.
- Wenn das img-Tag im CSS für ein responsives Design vorbereitet sind, wird die SVG-Grafik genauso an den Monitor angepasst wie jpg oder png.
- **Nachteile**
  Links und Javascript in der SVG-Grafik werden nicht unterstützt. Der Image-Upload von Drupal 7 / Drupal 8 verweigert bei SVG.

2. SVG mit iframe in HTML

Natürlich gibt auch HTML *iframe* eine SVG-Grafik in der Webseite aus.

```
<iframe height="180" width="100%" src="dashboard.svg" scrolling="no">
   <p>SVG-Grafik – hier eine Kopie als PNG</p>
   <img src="dashboard.png" width="220" height="220" alt="Dashboard" />
</iframe>
```

- **Vorteile**
  Genauso wie mit einem object-Tag stehen mit *iframe* die Links der SVG-Datei in der Webseite zur Verfügung.
- Javascript wirkt ebenfalls, wenn SVG via *iframe* in die Webseite gesetzt wird.
- iframes dürfen Cross-Domain (domainübergreifend) eingesetzt werden, vor allem aber können sie die Struktur der Herbergsseite nicht durch Fehler erschüttern.
- **Nachteile**

- Ein responsives iframe ist aufwändig und nicht intuitiv.

Sowohl der WordPress-Editor als auch der Drupal 8-Editor setzen ein HTML-iframe ohne Widerstand. Ein iframe muss in der Text-Ansicht (WordPress) oder Source/Quelltext-Ansicht (Drupal 8) eingefügt werden oder der Editor braucht eine Erweiterung.

3. SVG im object-Tag

```
<object data="fruity-icons.svg" type="image/svg+xml" width="600" height="193">
<!-- Für Browser ohne SVG-Unterstützung -->
<img src="fruity-icons.png" width="600" height="193" alt="Alternatives PNG-
Bild">
</object>
```

- **Vorteile**
  Fallback durch Inhalt des object-Tags für IE: Ein alternatives Bild zwischen dem öffnenden und schließenden object-Tag kann die alten Versionen von IE abspeisen.
- Formen können im SVG-Quelltext mit Links versehen werden.
- Die SVG-Grafik kann mit Javascript animiert werden (Javascript onmouseover auf dem Kreis im Hintergrund, Link auf der vorderen Sonnenblume).
- **Nachteile**
  Die Editoren der Content Management Systeme unterstützen das Einsetzen von object-Tags nicht unbedingt. Der WordPress-Editor muss in den Text-Modus geschaltet werden, der Drupal 8-Editor ersetzt das date-attribut durch about:blank.

4. SVG als Hintergrundbild

Ein SVG wird genauso wie ein Bitmap-Bild als background-image eingesetzt.

```
<div style="background: url(pattern.svg);">
</div>
```

- **Vorteil:**
  Passt sich an die Größe des Viewports an und kann durch Conditional Comments mit einem akzeptablen Ersatz für alte Browser versehen werden.

5. SVG direkt im HTML-Markup

```
<div class="svg">
    <svg width="100%" height="100%" viewBox="0 0 2287 1276">
    </svg>
```

```
</div>
```

- **Vorteile**
  Laden einer weiteren externen Datei entfällt.
- Formen können mit Links versehen werden
- kann direkt mit Javascript und CSS animiert werden.
- **Nachteile**
  Bei größeren Grafiken wird das Markup verd… unübersichtlich

6: SVG image-Tag

Und dann gibt es noch einen sechsten Weg, SVG in eine Webseite zu setzen – in einem svg-image-Tag. Browser, die SVG nicht unterstützen, korrigieren das SVG-<image>-Tag zum HTML-<img>-Tag und laden ein alternatives Bitmap-Bild. Diese Methode ist so genial, dass sie einen eigenen Beitrag verdient: SVG in einem image-Tag.

7: SVG als data-URL im CSS

Tipp von Andreas unten in den Kommentaren, der das SVG so elegant aus dem HTML in seinem Apfelgame heraushält: SVG-Hintergrundbilder via Data URL direkt in die CSS-Datei einbinden. Das hält das HTML sauber und vermeidet einen weiteren HTTP-Request.
SVG als CSS background via data-URL:

```
.svgbg { width:100%; height:150px; background: url("data:image/svg+xml;utf8,<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 100 44' height='44' width='100'><path d='m 0,11 100,0' style='stroke:#efefef;stroke-width:21.5px' /></svg>") }
```

Nur dabei daran denken, in Data-URLs die Sonderzeichen < und > und insbesondere das ‚#‘ zu encodieren. Damit SVG als *Data URL* auch in Firefox und in Internet Explorer angezeigt wird, lautet der korrekte String also

```
background: url("data:image/svg+xml,");
```

UTF8 kann man sich sparen, weil das Dokument schon auf UTF8 basiert.

SVG im HTML picture-Element

Und dann hätten wir noch das HTML picture-Tag. picture für sich alleine ist leer, es entfaltet sich erst durch eine Liste von alternativen Bildformaten.

```
<picture>
    <source srcset="fruity-icons.svg" type="image/svg+xml">
    <img src="fruity-icons.png" alt="fruity-icons" />
</picture>
```

Browser ohne Support für den Mime-Typ des Bildes oder denen die gesamten picture / source-Tags unbekannt sind (die letzten überlebenden IE8s) überspringen das source-Element und zeigen das Bild im img-Tag.

SVG mit Alternative: picture Tag

Das HTML picture-Tag teilt dem Browser Bilder und Grafiken in unterschiedlichen Formaten oder Seitenverhältnissen zu. Das img-Tag innerhalb des picture-Tags ist ein Fallback für Browser, die das picture-Tag noch nicht erkennen. Aber Browser ohne SVG-Support zählen heute zu den Raritäten.

```
<picture>
<source srcset="fruity-icons.svg" type="image/svg+xml">
<img src="fruity-icons.png" alt="fruity-icons" />
</picture>
```

SVG viewbox

SVG-Grafiken sind fast immer deutlich kleiner als Bitmaps und passen sich eigentlich ohne Verluste an das Platzangebot an.
Das »eigentlich« verrät schon die Crux: Die Anpassung der Größe funktioniert nicht in Internet Explorer. Responsive SVGs braucht ein paar Kniffe, um sich widerstandslos an die Größe des Containers anzupassen.
Das SVG-*viewBox*-Attribut bestimmt den sichtbaren Ausschnitt und das Seitenverhältnis. Nur nicht übersehen, dass *viewBox* mit einem großen »B« geschrieben wird! Und: Ältere Versionen von Inkscape haben das *viewBox*-Attribut nicht gesetzt.

```
               +-+-- Nullpunkt der Grafik
               | |
<svg viewBox="0 0 345 207" width="100%" height="100%">
                 |   |            |             |
                 V   V            V             V
          absolute    absolute    relative     relative
          Höhe        Höhe        Breite       Breite
             …
</svg>
```

Wenn das viewBox-Attribut vorhanden ist und *width* und *height* auf 100% gesetzt werden, passen alle modernen Browser die Grafik an das Platzangebot des umfassenden Containers an. Ist ja auch eigentlich kein Problem: Der dritte und vierte Posten in der *viewBox* gibt das Seitenverhältnis der Grafik an.

```
<figure style="max-width: 480px;margin: 1em auto">
<svg width="100%" height="100%" viewBox="0 0 345 207">
</figure>
```

Alle modernen Browser außer … IE9, IE10 und IE11. Hier ist die Grafik klein: 150px hoch.

SVG inline in Internet Explorer

Wenn der umfassende Container die Höhe nicht explizit ausweist, setzt IE die Höhe der SVG-Grafik mit 150px fest und passt dann die Breite an. Immerhin kann IE das Seitenverhältnis korrekt berechnen. Erst mit Microsoft EDGE endet dieses Dilemma.
Ein vielfach propagierte Lösung setzt auf – nicht ganz unkompliziertes – CSS. Der umfassende Container wird relativ, das SVG-Element absolut im umfassenden Container positioniert.
Die SVG-Grafik sitzt in einem Container mit class="svgframe".

```
<style>
.svgframe { width:100%; position: relative;
    display:inline-block; vertical-align: middle;
    padding-bottom: 50%;  // berechnet aus dem Seitenverhältnis height/width
    overflow:hidden }

.frame svg { position: absolute }
</style>

<div class="svgframe">
<svg viewBox="0 0 1489 724"
    width="100%" height="100%"
    preserveAspectRatio="xMinYMin meet">
…
</div>
```

Detaillierte Beschreibung der einzelnen Schritte Make SVG Responsive.
Das Gelbe vom Ei ist diese Technik nicht, denn für jede Grafik muss das Seitenverhältnis *Höhe:Breite* individuell berechnet und als *padding* in CSS angegeben werden.

SVG mit Javascript anpassen

Einige Zeilen Javascript reichen, um padding-bottom für alle Grafiken der Seite anzupassen.
preserveAspectRatio ist nicht nötig. Das Wesentliche ist und bleibt das viewBox-Attribut.

```
<div class="svgframe">
   <svg width="100%" height="100%" viewBox="0 0 1353 442">
      <rect fill="gold" x="0" y="0" width="400" height="100" />
      …
   </svg>
</div>

<script>
var svg = document.querySelectorAll('.svgframe svg');
var len = svg.length;
for (var i=0; i<len; i++) {
   if (svg[i].getAttribute('width') == '100%' && parseInt(svg[i].parentNo-
de.clientWidth) > 0) {
      var viewBox = svg[i].getAttribute('viewBox');
      viewBox = viewBox.replace(/\s\s+/g, ' ');
      var w = viewBox.split(' ')[2];
      var h = viewBox.split(' ')[3];
      var x = h / w * 100;
      svg[i].parentNode.setAttribute('style', 'padding-bottom:' + x +'%');
   }
}
</script>
```

SVG in einem img-Tag

Das ist die einfachste aller Varianten: Die URL der Grafik sitzt im src-Attribut wie bei einem Bitmap-Bild und das Bild passt sich automatisch an die Größe des umfassenden Containers an.
Das funktioniert sogar in Internet Explorer: IE9, IE10, IE11 und Edge.

```
<style>
img { width: 100%}
</style>
   …
<figure class="svgimg">
```

```
   <img src="accordion.svg" width="480" height="288" alt="SVG in einem image
Tag">
</figure>
```
Wenn die SVG-Grafik mit einem HTML img-Tag eingesetzt wird, reicht es, img { width: 100% } im CSS zu notieren. Die neueren Version von Adobe Illustrator und Inkscape setzen das viewBox-Attribut automatisch, aber ältere Versionen von Inkscape nicht.
Das gilt sowohl für inline-SVG als auch für SVG in einem img-Tag. Bei einem img-Tag muss nur im CSS *img { width: 100%; height: auto }* notiert sein – .
In einem img-Tag funktioniert die Anpassung in allen Browsern – vorausgesetzt, das viewBox-Attribut ist in der SVG-Datei gesetzt. Immer das viewBox-Attribut in das SVG-Tag der Grafik setzen und width- und height-Attribut der Grafik auf 100% setzen – das haben wir für ein responsives Webdesign sowieso bereits in der Hinterhand.

Ausschnitt mit viewBox festlegen

Das viewBox-Argument des SVG-Tags bestimmt den sichtbaren Ausschnitt einer Grafik mit nur vier Werten: x (Nullpunkt), y (Nullpunkt), width und height. x und y sind der Ursprung oder Nullpunkt des Koordinatensystems. Einen Bildausschnitt legt man durch einen neuen Nullpunkt und die Größe des Ausschnitts fest.
Auf kleinen Monitore macht es Sinn, nur einen Ausschnitt des Bildes / des Grafik zu zeigen. Das funktioniert mit Javascript und matchMedia.

```
var postman = document.querySelector ('#postman');
var mql = window.matchMedia("(min-width:740px)");

clipImage(mql); // Initialen Ausschnitt beim Laden der Seite festlegen

mql.addListener(clipImage); // Spezieller Event Listener für MediaQueryList

function clipImage(mql) {
    if (mql.matches) {
        postman.setAttribute('viewBox','0 0 1436 731');
    } else {
        postman.setAttribute('viewBox','180 120 600 600');
    }
}
```

Javascript matchMedia wird von Safari, Chrome, Opera und Firefox unterstützt, IE ab Version IE10.
Damit IE das Bild beschneidet, muss svg:not(:root) { overflow: hidden } im CSS notiert werden (das ist aber in allen normalize / boilerplate-Variationen schon vertreten).