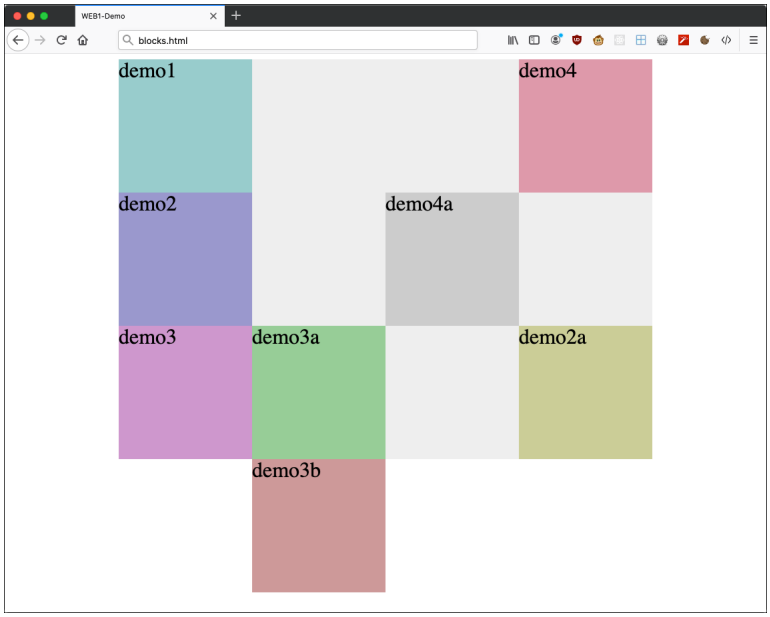
WBE-Praktikum 8

CSS-Layout

Aufgabe 1: Übung zur Positionierung

In dieser Aufgabe soll das Positionieren von Elementen geübt werden. Dazu werden in einem *main*-Element mit der Grösse 800x600px (grauer Hintergrund) mehrere Quadrate mit 200px Seitenlänge positioniert, so dass die nebenstehende (nicht allzu praxisrelevante) Darstellung resultiert.



Die HTML-Struktur und ein paar CSS-Regeln sind bereits vorgegeben. Sie finden die Datei im Verzeichnis *blocks* im Praktikumsarchiv. Ausnahmsweise sind die CSS-Regeln in der HTML-Datei untergebracht, damit Sie beim Positionieren der Blocks die HTML-Struktur im Auge behalten können.

- Ändern Sie die CSS-Regel für das *main*-Element so ab, dass das Layout horizontal zentriert ist.
- Muss an der Position von *demo1* und *demo2* etwas geändert werden? Warum oder warum nicht? Machen Sie falls nötig die Ergänzungen in den beiden Regeln.
- Positionieren Sie *demo2a* auf möglichst einfache Weise an der richtigen Stelle.
- Positionieren Sie *demo3* relativ. Setzen Sie explizit die Werte für *top* und *left*.
- Positionieren Sie *demo3a* und *demo3b* auf möglichst einfache Weise an der richtigen Stelle.
- Positionieren Sie *demo4* und *demo4a* auf möglichst einfache Weise an der richtigen Stelle.

Nun sollten alle Boxen an der gewünschten Stelle positioniert sein. Sie sollen nun aus Ausgangspunkt für ein paar weitere Experimente verwendet werden:

- Überlegen Sie, welche Auswirkungen es hat, wenn Sie in der Regel für *demo3* (relativ positioniert) *top:100px* und *left:-100px* setzen. Probieren Sie es aus.

- Füllen Sie *demo1* mit viel Text, so dass dieser nicht mehr in die Box passt. Dehnt sich die Box dadurch aus? Testen Sie in diesem Zusammenhang die verschiedenen möglichen Werte der Eigenschaft *overflow*.
- Überlegen Sie, welche Auswirkungen es hat, wenn Sie *demo1* mit der Positionierung *fixed* versehen, sowie *bottom:0* und *right:0*. Probieren Sie es aus. Hinweis: Der Wert 0 wird normalerweise ohne Grösseneinheit verwendet.

Noch ein paar weitere Versuche:

- Entfernen Sie das *display:none* von *demo1a*: In *demo1* wird ein grauer Balken angezeigt.
- Setzen Sie die Breite (*width*) des Balkens auf *15em*. Vermutlich wird der Balken nun über die Box hinausragen. Warum ist das so? Bei welcher Grösse in *em* ist der Balken genauso breit wie die Box? Wie gross ist *1em*?
- Können Sie berechnen, welche Grösse man in *cm* angeben müsste, damit der Balken genau die Breite der 200px-Box hat?

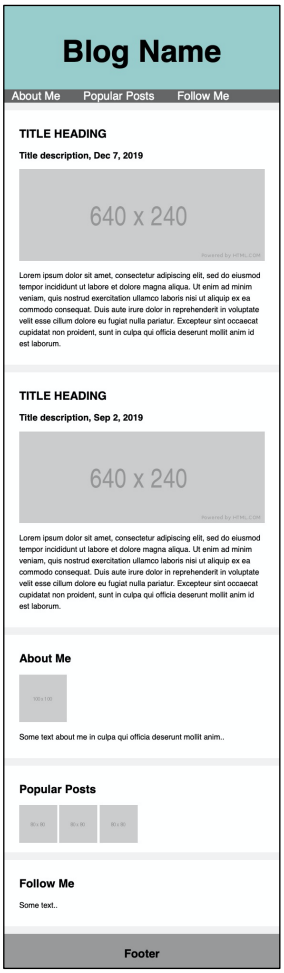
Es ist möglich, dass die Grössenangabe in *cm* nicht genau der physikalischen Grösse cm entspricht, da je nach Ausgabetyp die Grössenangabe entweder an den physikalischen Grössen (Beispiel: Ausdruck) oder an der Grösse eines so genannten Referenzpixels ausgerichtet ist. Zu den Grössenangaben: https://devdocs.io/css/length

Aufgabe 2: Layout für die Blog-Seite

Die im letzten Praktikum erstellte Blog-Seite soll nun Schritt für Schritt zu einem *responsiven Layout*, welches sich für verschiedene Bildschirmgrössen eignet, verbessert werden. Wegen der grossen Bedeutung der Mobilplattform beginnen wir mit der Darstellung für kleine Bildschirme (dieser Ansatz wird als *Mobile First*[1]) bezeichnet.

Wenn Sie das Gefühl haben, dass sich Ihre Lösung aus dem letzten Praktikum nicht für die Anpassungen eignet, können Sie auch die Musterlösung zum Praktikum als Grundlage für diese Aufgabe verwenden.

- Verwenden Sie die Eigenschaften des Box-Modells (*margin*, *padding*, …), um eine für kleine Bildschirme geeignete Darstellung (s. nebenstehendes Bild) zu erhalten. Blocks ohne Breitenangabe nehmen die verfügbare Breite ein, was hier von Vorteil ist.
- Testen Sie das Ergebnis mit verschieden grossen Browser-Fenstern. Die Darstellung sollte sich an die Fenstergrösse anpassen.



[1] Luke Wroblewski: Mobile First, A Book Apart 2011, http://mobile-first.abookapart.com

- Sind viele Artikel auf der Seite muss man zu den Einträgen *About Me*, *Popular Posts* und *Follow Me* weit nach unten scrollen. Daher ist es sinnvoll, im Header ein Navigationselement einzufügen, das Links zu diesen Inhalten enthält. Dazu werden die Inhalte mit einer *id* versehen und die Links so angelegt, dass beim Anklicken zu diesen Elementen gesprungen wird, zum Beispiel:
  `<a href="#about">About Me</a>`

- Versuchen Sie, die Navigation in etwa so zu gestalten wie es das Bild oben zeigt. Das braucht etwas Erfahrung mit CSS. Kein Problem also, wenn es zunächst noch nicht so schön aussieht.

Das bisher umgesetzte Layout eignet sich für schmale Bildschirme. Auf einem Desktop-Monitor mit grossem Browserfenster werden die Texte dann aber zu breit, um noch gut lesbar zu sein. Typografen verwenden hierfür übrigens die Bezeichnung *Satzbreite*. Und Webentwickler sprechen normalerweise nicht von der Grösse des Browserfensters, sondern vom *Viewport*. Das ist der Bereich des Browserfensters, in welchem die Webseiten angezeigt werden.

Mit Hilfe von *Media-Queries* können Sie die Darstellung an bestimmte Viewport-Eigenschaften, zum Beispiel dessen Breite, anpassen. Beispiel: CSS-Regeln für Viewport-Breiten ab 660px:

```
@media screen and (min-width: 660px) {
  /* CSS-Regeln */
}
```

- Ergänzen Sie Ihr Stylesheet so, dass die Breite des Inhaltsbereichs bei grossem Viewport so beschränkt wird, dass die Texte noch gut lesbar sind (*max-width*). Ausserdem können Sie den Inhaltsbereich mit *margin: 0 auto* zentrieren.

- Damit Bilder nicht aus ihrem Container herausragen, setzt man ihre maximale Breite auf 100%.

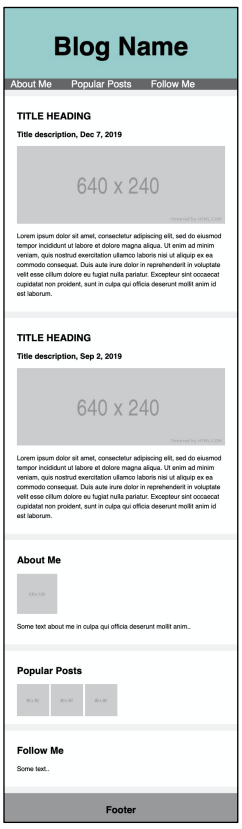
Die Darstellung der Seite passt sich nun an die Grösse des Viewports an. Steht genug Platz zur Verfügung, werden die Texte in der Breite limitiert und der gesamte Inhalt zentriert. Ist dies nicht der Fall, passt sich die Layout-Breite an die Viewport-Breite an. Etwas fehlt aber noch: Wenn genügend Platz vorhanden ist, könnten die Einträge *About Me*, *Popular Posts* und *Follow Me* neben den Artikeln statt darunter platziert werden.

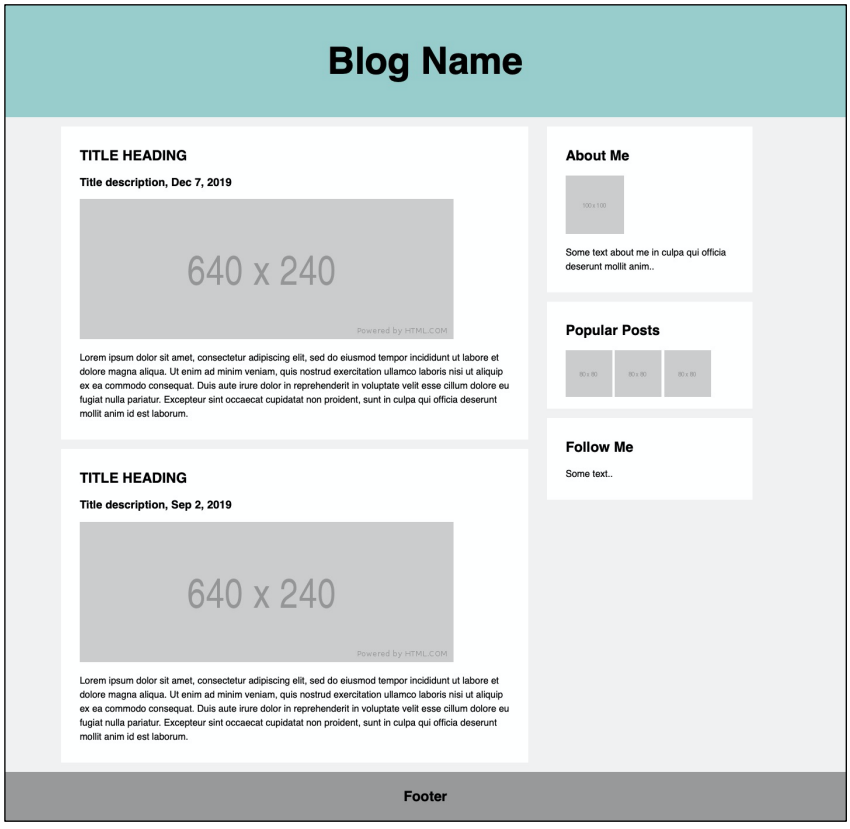
Wir haben es nun mit einem Mehrspalten-Layout zu tun, etwas, das traditionell mit CSS schwierig umzusetzen war. In der Anfangszeit des Webdesigns wurden zu diesem Zweck häufig Tabellen eingesetzt, was mit zahlreichen Problemen verbunden war. Ausserdem waren Inhalt und Darstellung auf diese Weise eng miteinander verknüpft. So blieb nur, das Ziel mittels fliessender Boxen (*float, clear*) zu erreichen – nicht ganz einfach umzusetzen und oft mit Kompromissen verbunden. Das führte dazu, dass viele Webdesigner auf Tools wie Bootstrap ausgewichen sind. Erst *Flexbox* und später *Grid-Layout* erlaubten eine saubere Umsetzung komplexer Layouts mit den Mitteln von CSS.

- Ergänzen Sie Ihr Stylesheet so, dass die die Einträge *About Me*, *Popular Posts* und *Follow Me* bei genügend Platz neben den Artikeln angezeigt werden (s. Bild). Sinnvollerweise sind diese Einträge dazu in einem *aside*-Element[2] zusammengefasst. Am besten verwenden Sie Flexbox dafür. Sie können natürlich auch Versuche mit *float/clear* machen.

- Das Menü mit den Links ist im breiten Layout eigentlich nicht nötig, da die Elemente direkt sichtbar sind. Es kann hier also ausgeblendet werden (*display: none*).

- Testen Sie die Seite mit verschieden grossen *Viewports*. Mit *Resize / View Responsive Layouts* in der *Web Developer Toolbar* können Sie die verschiedenen Grössen auch auf einer Seite im Überblick darstellen. Dazu muss allerdings über einem Server auf die Seite zugegriffen werden, zum Beispiel über den *Live Server* in *VSCode*.





[2] Streng genommen bedeutet *aside* nicht, dass ein Element *neben* einem anderen Element angezeigt wird, da HTML nichts mit der Darstellung zu tun haben soll. Es bedeutet eher: es hat inhaltlich nur indirekt mit dem Hauptinhalt zu tun.