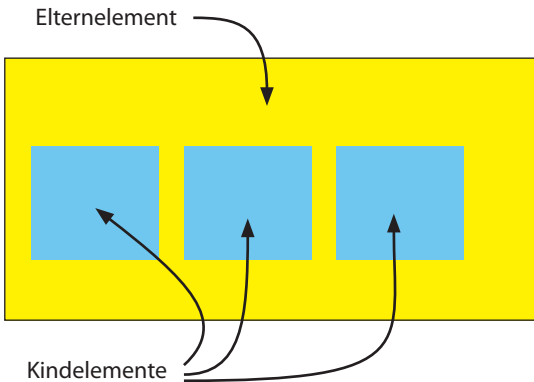


Flexbox

Mit Flexbox lassen sich mehrere Blockelemente flexibel innerhalb eines Elternelements positionieren. Wichtig: Flexbox funktioniert nur mit Blockelementen!

Eine Flexbox besteht immer aus einem Elternelement, in welchem ein oder mehrere Kindelemente angeordnet werden. Flexbox gilt als eindimensionales System, weil sich die Kindelemente entweder in Zeilen oder in Spalten anordnen lassen. Eine Kombination aus beiden ist nicht möglich. Eine zweite Generation („Enkelkindelemente“) gibt es nicht, aber es ist möglich, dass ein Kindelement Elternelement für darunter liegende Kindelemente ist.

Während die Begriffe „Eltern“ und „Kinder“ die Beziehung zwischen den Blockelementen verdeutlichen, sind in der Fachwelt auch die Begriffe „Container“ für Elternelement und „Item“ für Kindelement verbreitet. Beispiel einer einfachen Flexbox:

Code	Schematische Darstellung
<pre> 01 <style> 02 .container { 03 display: flex; 04 } 05 </style> 06 <body> 07 <div class="container"> 08 <div class="flex-item"></div> 09 <div class="flex-item"></div> 10 <div class="flex-item"></div> 11 </div> 12 </body> </pre>	

Es ist ausreichend, das Elternelement zu bestimmen und mit der CSS-Regel `display: flex;` die Eigenschaften einer Flexbox zuzuweisen. Damit werden die direkt darunter liegenden HTML-Blockelemente automatisch zu Flexbox-Kindelemente. Interessant ist, dass die Kindelemente standardmäßig nebeneinander dargestellt werden, obwohl Blockelemente einen Zeilenumbruch erzwingen.

Im weiteren Verlauf können dem Elternelement, den Kindelementen und auch einzelnen Kindelementen verschiedene Eigenschaften zugewiesen werden.

Elternelement und Kindelemente – Übungsaufgabe

Markieren Sie im folgenden Code Elternelemente gelb und Kindelemente blau. Achtung: Wenn Sie meinen, dass ein Element sowohl Kind- als auch Eltern ist, dann markieren Sie dieses bitte grün.

```

01 <body>
02   <nav>
03     <ul> <li> </li> <li> </li> </ul>
04   </nav>
05   <main>
06     <section> </section> <aside> </aside>
07   </main>
08 </body>

```

CSS-Eigenschaften von Flexbox-Container

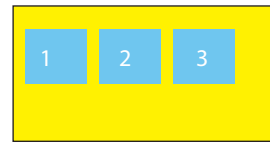
Nachdem ein Elternelement mit der CSS-Eigenschaft `display: flex;` als Flex-Container definiert wurde, kann mit weiteren Angaben das Verhalten Kindelemente (Flex-Items) bestimmt werden.

Die Eigenschaft **flex-direction** bestimmt die Hauptachse. Als Standard gilt die X-Achse (Horizontale) als Hauptachse. Mit dem Wert `column` bzw. `column-reverse` wird die Y-Achse (Vertikale) zur Hauptachse. Mögliche Werte sind:

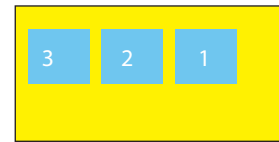
- » `row` (standard)
- » `row-reverse`
- » `column`
- » `column-reverse`

Beispiel:

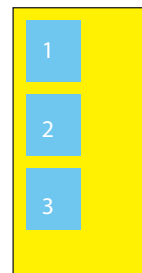
```
.container{
  flex-direction: row-reverse;
}
```



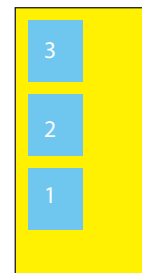
row



row-reverse



column



column-reverse

Mit der Eigenschaft **justify-content** werden die Flex-Items innerhalb des Flex-Containers auf der Hauptachse ausgerichtet. Mögliche Werte sind:

- » `flex-start` (standard)
Linksbündige Ausrichtung
- » `flex-end`
Rechtsbündige Ausrichtung
- » `center`
Mittige Ausrichtung
- » `space-between`
Gleichmäßige Verteilung der Flex-Items über die ganze Breite
- » `space-around`
Jedes Flex-Item hat den gleichen Außenabstand.
- » `space-evenly`
Flex-Items sind so verteilt, dass alle Abstände und Zwischenräume gleich sind.



flex-start



flex-end



center



space-between



space-around



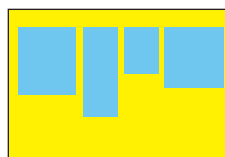
space-evenly

Beispiel:

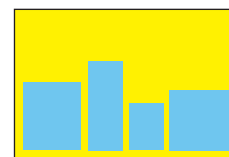
```
.container{
  justify-content: space-between;
}
```

Die Ausrichtung der Items auf der Querachse wird mithilfe der Eigenschaft **align-items** bestimmt. Mögliche Werte sind:

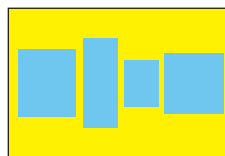
- » **flex-start**
Ausrichtung oben
- » **flex-end**
Ausrichtung unten
- » **center**
Ausrichtung in der Mitte
- » **stretch**
Flex-Items erstrecken sich über die gesamte Höhe.
- » **baseline**
Ausrichtung an der Grundlinie des Textes



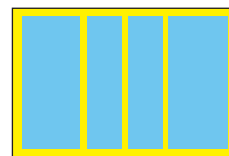
flex-start



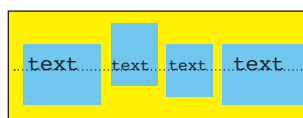
flex-end



center



stretch



baseline

Beispiel:

```
.container{
  align-items: center;
}
```

Die Ausrichtung von Zeilen mit Kindelementen gibt man mit der Eigenschaft **align-content** an. Mögliche Werte sind:

- » **flex-start**
Ausrichtung oben
- » **flex-end**
Ausrichtung unten
- » **center**
Ausrichtung in der Mitte
- » **stretch**
Zeilen erstrecken sich über die gesamte Höhe.
- » **space-between**
Zeilen werden gleichmäßig über die gesamte Höhe verteilt.
- » **space-around**
Die Zeilen haben oben und unten jeweils den gleichen Abstand.



flex-start



flex-end



center



space-between

Beispiel:

```
.container{
  align-content: flex-start;
}
```

Die Eigenschaft **flex-wrap** beeinflusst den „Zeilenumbruch“ der Flex-Items.

Mögliche Werte sind:

- » `no-wrap` (standard), kein Zeilenumbruch
- » `wrap`
- » `wrap-reverse`, Zeilenumbruch in umgekehrter Reihenfolge

Beispiel:

```
.container{
  flex-wrap: wrap;
}
```

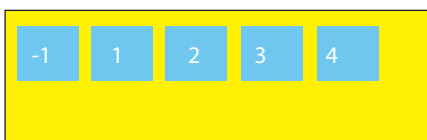
CSS-Eigenschaften von Flexbox-Items (Kindelemente)

Sobald ein Flexbox-Container definiert ist, sind alle direkten Kindelemente Flex-Items und können somit bestimmte Eigenschaften erhalten.

Mit der Eigenschaft **order** kann die Reihenfolge geändert werden. Als Wert wird eine Zahl erwartet. Als Standard gilt die Reihenfolge, welche die Elemente im Code haben. Beispiel:

```
.flex-item:last-child{
  order: -1;}

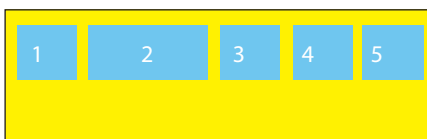
```



Mit **flex-grow** kann einem Kindelement eine individuelle Größe abhängig von den anderen Items zugewiesen werden. So kann z. B. ein Kindelement eine x-fache Größe im Vergleich zu den anderen Elementen haben. Beispiel für doppelte Größe:

```
.item-2 {
  flex-grow: 2;}

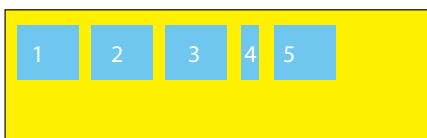
```



Genau wie **flex-grow** ein Kindelement vergrößert, wird ein Kindelement mit der Eigenschaft **flex-shrink** verkleinert. Beispiel für eine Verkleinerung auf ein Viertel:

```
.item-1-4{
  flex-shrink: 4;}

```



Mit der Eigenschaft **align-self** können einzelne Kindelement außerhalb der `align-items` Angabe des Containers positioniert werden. Mögliche Werte wie beim Flex-Container (`align-items`):

- | | |
|---|--|
| <ul style="list-style-type: none"> » <code>flex-start</code>
Ausrichtung oben » <code>flex-end</code>
Ausrichtung unten » <code>center</code>
Ausrichtung in der Mitte | <ul style="list-style-type: none"> » <code>stretch</code>
Kindelemente erstrecken sich über die gesamte Höhe. » <code>baseline</code>
Ausrichtung an der Grundlinie des Textes |
|---|--|

Pseudoklassen und Flexbox

Es ist vergleichsweise leicht, eine Flexbox zu beschreiben, in der sich alle Flexitems gleich verhalten. Möchte man unter den Kindelementen eine Variation haben, ist es nicht unbedingt notwendig, den Flex-Items eigene Klassennamen zu geben. Viel einfacher und flexibler ist der Einsatz der Pseudoklasse „child“, mit der man die Auswahl der Kindelemente begrenzen kann. Gegeben ist das folgende Beispiel einer Liste, deren Listenelemente Flex-Items sind:

```

01  <style>
02  ul {
03    display: flex;
04  }
05  </style>
06  <body>
07    <ul>
08      <li> </li> <li> </li> <li> </li>
09    </ul>
10  </div>
11  </body>

```

Beispiele, um im obigen Beispiel eine bestimmte Auswahl an Kindelementen auszuwählen:

<i>CSS-Selektor</i>	<i>Bedeutung, Auswahl</i>
ul{}	Flex-Container
ul>li{}	Alle Flex-Items
ul>li:first-child{}	Das erste Flex-Item
ul>li:nth-child(2){}	Das zweite Flex-Item
ul>li:nth-child(even){}	Jedes zweite Flex-Item
ul>li:nth-child(odd){}	Jedes Flex-Item an ungerader Position (1, 3, 5 ...)
ul>li:nth-child(3n+2){}	Jedes dritte Flex-Item ab dem zweiten Flex-Item
ul>li:last-child{}	Das letzte Flex-Item

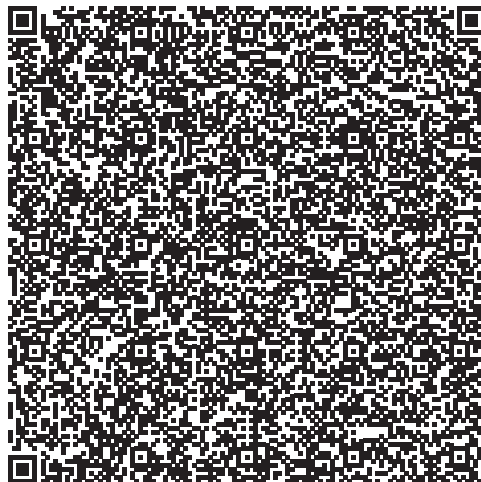
Navigationsleiste mit Flexbox - Übungsaufgaben

Code-Ausschnitt für die Navigationsleiste:

```
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">Philosophie</a></li>
<li><a href="#">Produkte</a></li>
<li><a href="#">Team</a></li>
<li><a href="#">Kontakt</a></li>
</ul>
</nav>
```

Kompletter HTML - Code

Aufgabe1_navigationsleiste.html



Aufgabe 1

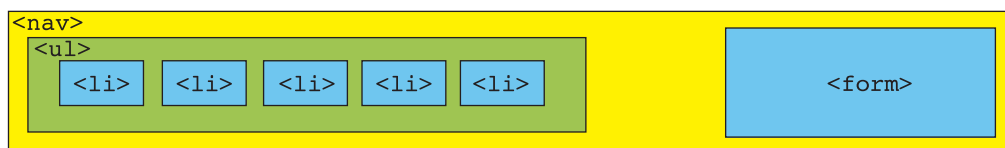
a) Ergänzen Sie den CSS-Code mit Hilfe von Flexbox, damit die Navigationspunkte nebeneinander stehen und mittig positioniert sind.

b) Ergänzen Sie das Menü mit einem Suchfeld und ordnen Sie die Menüpunkte linksbündig und das Suchfeld rechtsbündig an. HTML-Code für das Suchfeld:

```
<form> <input type="search"> <button>Suchen</button> </form>
```

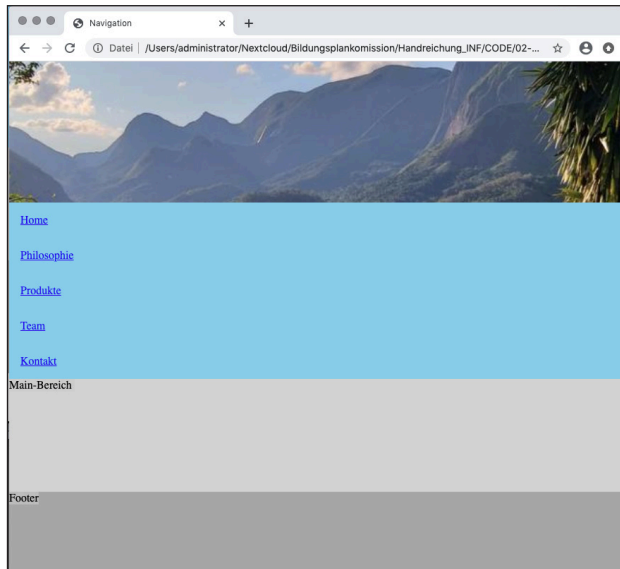
Hinweis: Um diese Aufgabe zu lösen, bietet es sich an, zwei Flexboxen `<nav>` und `` nach dem folgenden Schema zu erstellen.

Schema:

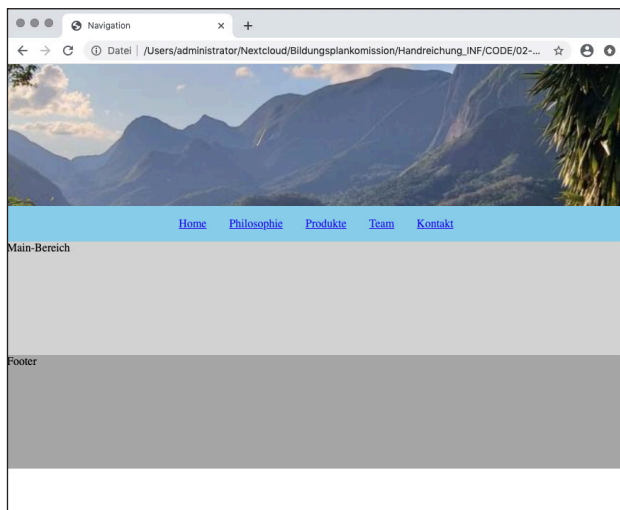


Ergänzen Sie die Eigenschaft `padding-right: 1em;` beim Selektor `nav`, damit die Suchfunktion Abstand zum rechten Rand erhält.

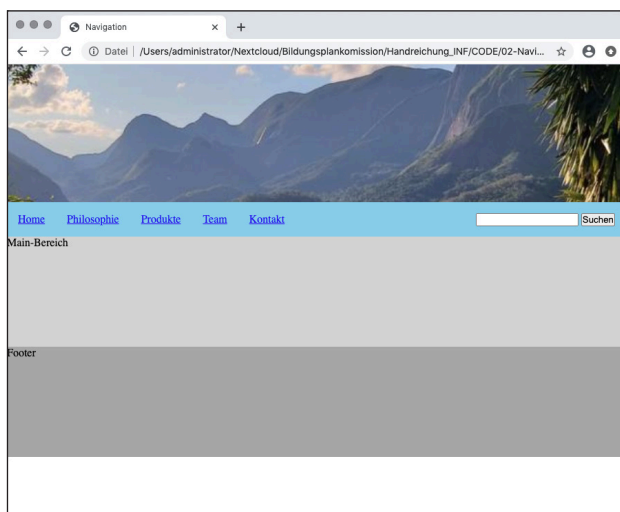
Screenshots zur Aufgabe1 „Navigationsleiste mit Flexbox“



unbearbeitete Datei



Horizontale Navigationsleiste mit Flexbox, mittig



Horizontale Navigationsleiste mit Flexbox linksbündig,
Suchfunktion rechtsbündig

Aufgabe 2

Ergänzen Sie den Code um eine Brotkrumennavigation nach folgenden Schema:

Home > Link1 > Link2

Fügen Sie dafür in Ihre HTML Dokument den nachfolgenden HTML-Code zwischen den Elementen `<nav>` und `<main>` ein.

```
01 <section id="breadcrumb"> <ul>
02 <li><a href="#">Home</a></li>
03 <li><a href="#">Link1</a></li>
04 <li><a href="#">Link2</a></li>
05 </ul>
06 </section>
```

Bedeutung:

Ergänzen Sie die folgenden CSS-Angaben in Ihrer CSS-Datei:

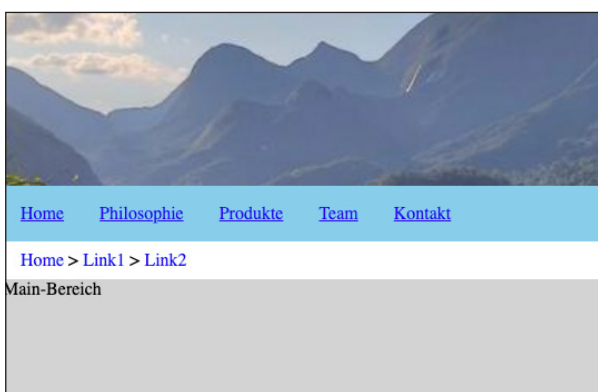
```
#breadcrumb ul{
    list-style-type: none;
    padding: .5em 1em;
    display: flex;
}
```

```
#breadcrumb a{
    text-decoration: none;
}
```

```
#breadcrumb ul li:after{
    content: „\0020>“;
    margin-right: .25em;
}
```

```
#breadcrumb ul li:last-child:after{
    content: „“;
    margin-right: 0;
}
```

Beschreiben Sie die Bedeutung der CSS-Regeln der Breadcrumb-Zeile.



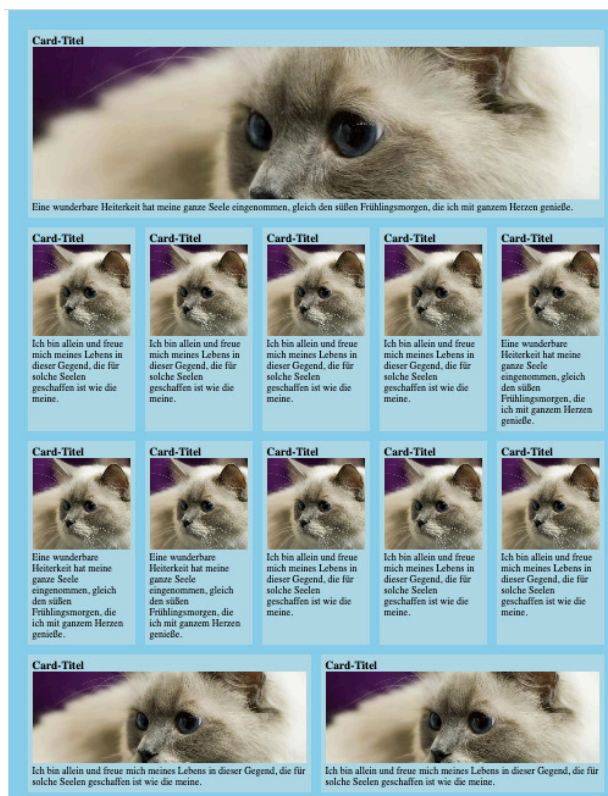
Bildausschnitt der Navigationsleiste mit „Breadcrumb“-Zeile.

Aufgabe 3

Gegeben ist eine HTML-Datei mit einem `main`, der mehrere „Cards“ enthält. Die Cards sind als `article` angelegt. Die Anzahl der Cards spielt keine Rolle.

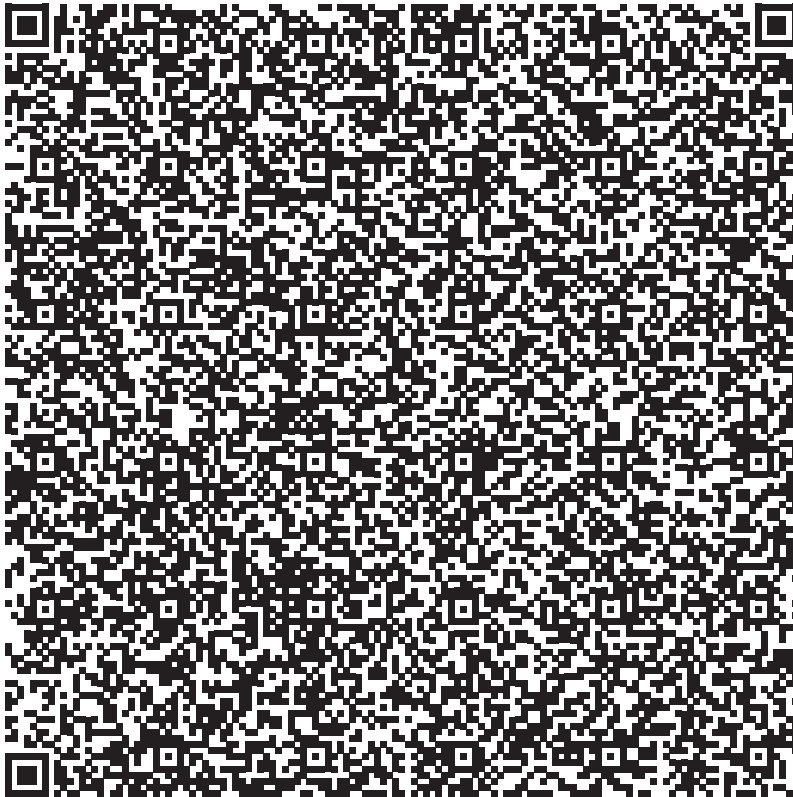
Ergänzen Sie den Code, damit `main` im Viewport ab einer Breite von 960px zentriert positioniert wird. Unterhalb von 960px soll `main` 100% Viewportbreite haben.

Tipp: Verwenden Sie flex-basis! Machen Sie aus `body` eine Flexbox und aus `main` ein Flex-Item. Optimieren Sie den Code für den 1. `article`, damit dieser immer die ganze Breite einnimmt. Die Breite der letzten `article` ist variabel und muss nicht codiert werden.



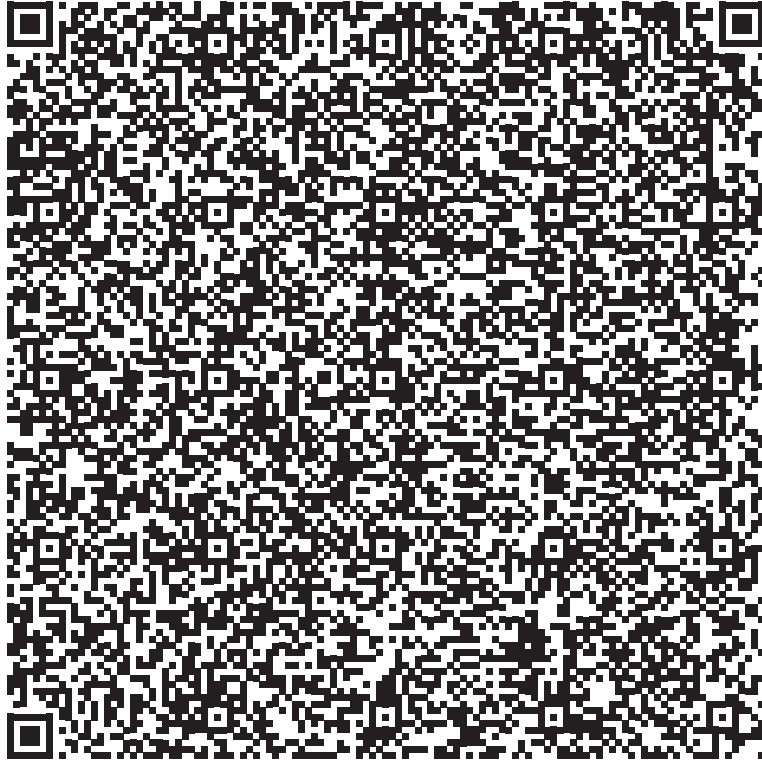
Darstellung der fertigen Datei.

Aufgabe 3 - Daten



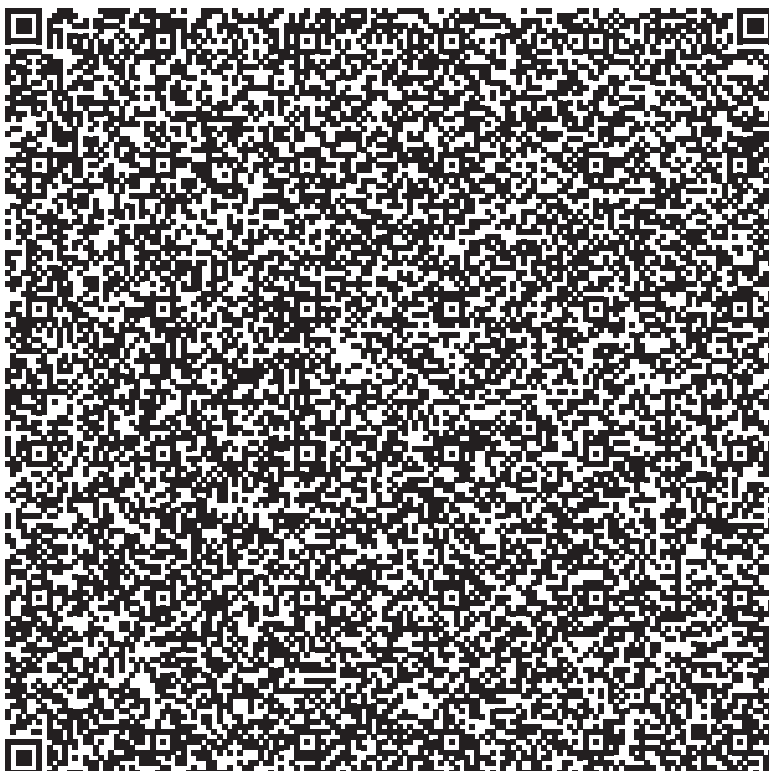
HTML-Datei mit einem
`article`-Element.
Multiplizieren Sie `article`
beliebig oft.

Lösungsvorschläge



Aufgabe 1

Aufgabe1_navigationslesite_LV.html



Aufgabe 2

Aufgabe1_navigationslesite_LV.html

Lösungsvorschläge



Aufgabe 3

nur das head-Element mit
dem `style`-Bereich