

MOBA1

MOBILE WEB: STYLE AND LAYOUT

OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: CSS Flexible Box Layout Module
- Layouts: CSS Grid Layout
- Final Remarks

OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: CSS Flexible Box Layout Module
- Layouts: CSS Grid Layout
- Final Remarks

CASCADING STYLE SHEETS

From WEB1 (or WBE preparation course) you already know:

- Semantic markup
- CSS2 and CSS3 selectors
- Styling text and boxes
- Box model, positioning elements, floating boxes

CASCADING STYLE SHEETS

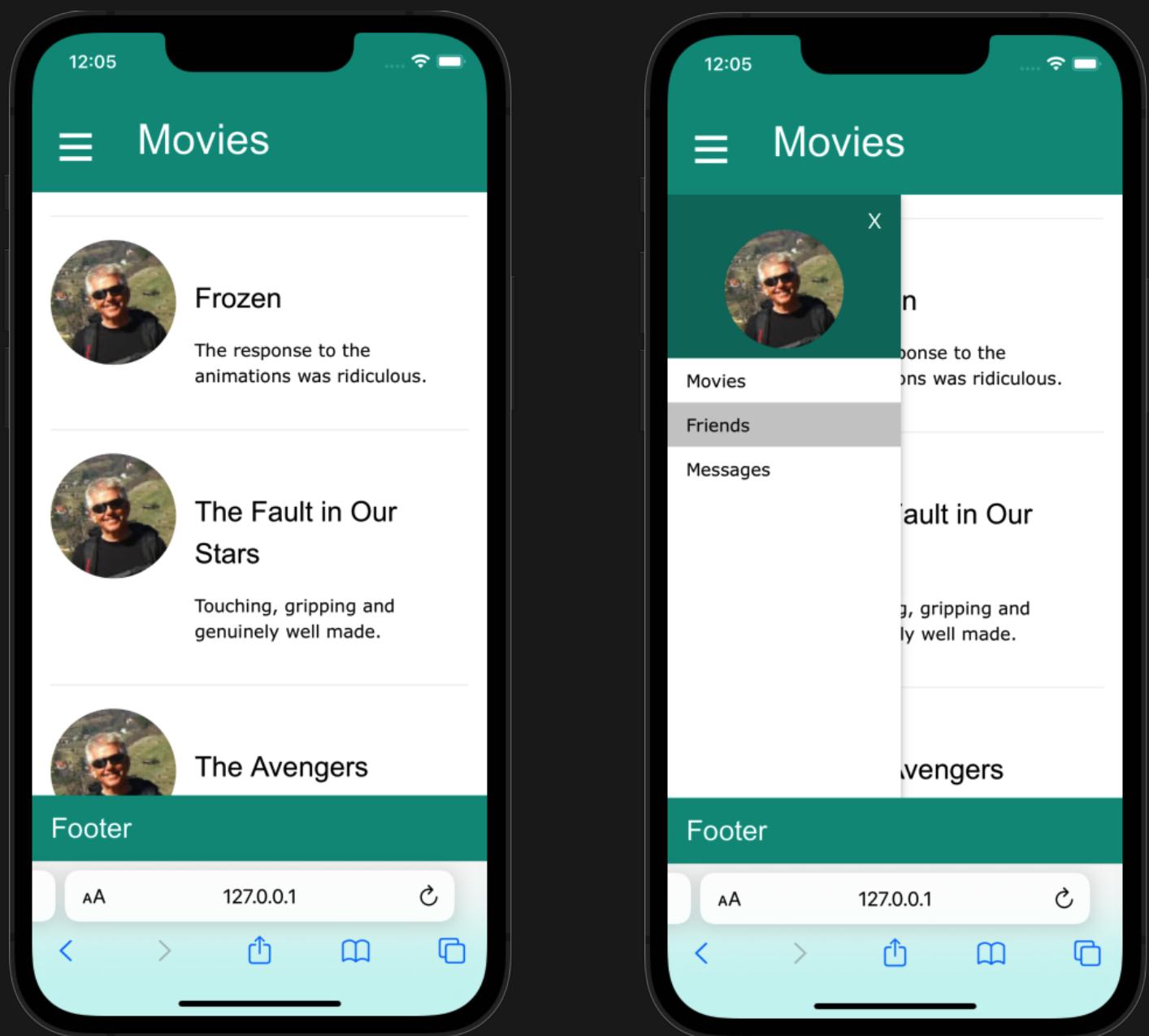
- If necessary consult:
 - WEB1 slides and related material
 - WBE preparation course
<https://moodle.zhaw.ch/course/view.php?id=1007>
 - Learn to Code HTML & CSS
<https://learn.shayhowe.com/html-css/>
- You probably also know:
 - Front-end frameworks like [Foundation](#) or [Bootstrap](#)
 - CSS preprocessors like [Less](#) or [Sass](#)

GRAPHICS AND LAYOUT

- Graphical effects:
Rounded corners, shadow effects, rotated content, animations
- Downloadable fonts, hyphenation
- Graphics and Graphics APIs:
SVG, Canvas, WebGL
- Complex layouts:
Flexbox, CSS Grid

<https://w3c.github.io/web-roadmaps/mobile/graphics.html>

GRAPHICS AND LAYOUT



FONTS

- Downloadable fonts:
WOFF File Format 2.0

Technologies in progress:

- CSS Font Loading: progressive availability of fonts
- `font-display`: how to display font while loading
- Font Variation Properties

SYSTEM FONTS

```
font-family: -apple-system, BlinkMacSystemFont,  
"Segoe UI", "Roboto", "Oxygen", "Ubuntu", "Cantarell",  
"Fira Sans", "Droid Sans", "Helvetica Neue",  
sans-serif;
```

- San Francisco in Safari on Mac OS X and iOS
- Helvetica Neue/Lucida Grande on older versions of Mac OS X
- BlinkMacSystemFont is the equivalent for Chrome on Mac OS X
- Segoe UI targets Windows and Windows Phone
- Roboto targets Android and newer Chrome OS
- ...

SYSTEM FONTS

This should be rendered in your system's UI font. The quick brown fox jumps over the lazy dog.

Using UI System Fonts In Web Design: A Quick Practical Guide

HYPHENS

- CSS `hyphens` property
- How words should be hyphenated when text wraps across lines
- Values: `none` | `manual` | `auto` | `inherit` | `initial` | `unset`
- Initial value: `manual`
- Language-specific: `lang` attribute

```
<p>Manual hyphens at &hyphen; or &shy; (if needed) :<br>
An extreme&shy;ly long English word</p>
```

```
<p>Automatic hyphens where the algorithm is deciding (if needed)</p>
```

OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: CSS Flexible Box Layout Module
- Layouts: CSS Grid Layout
- Final Remarks

DEVICE ADAPTATION

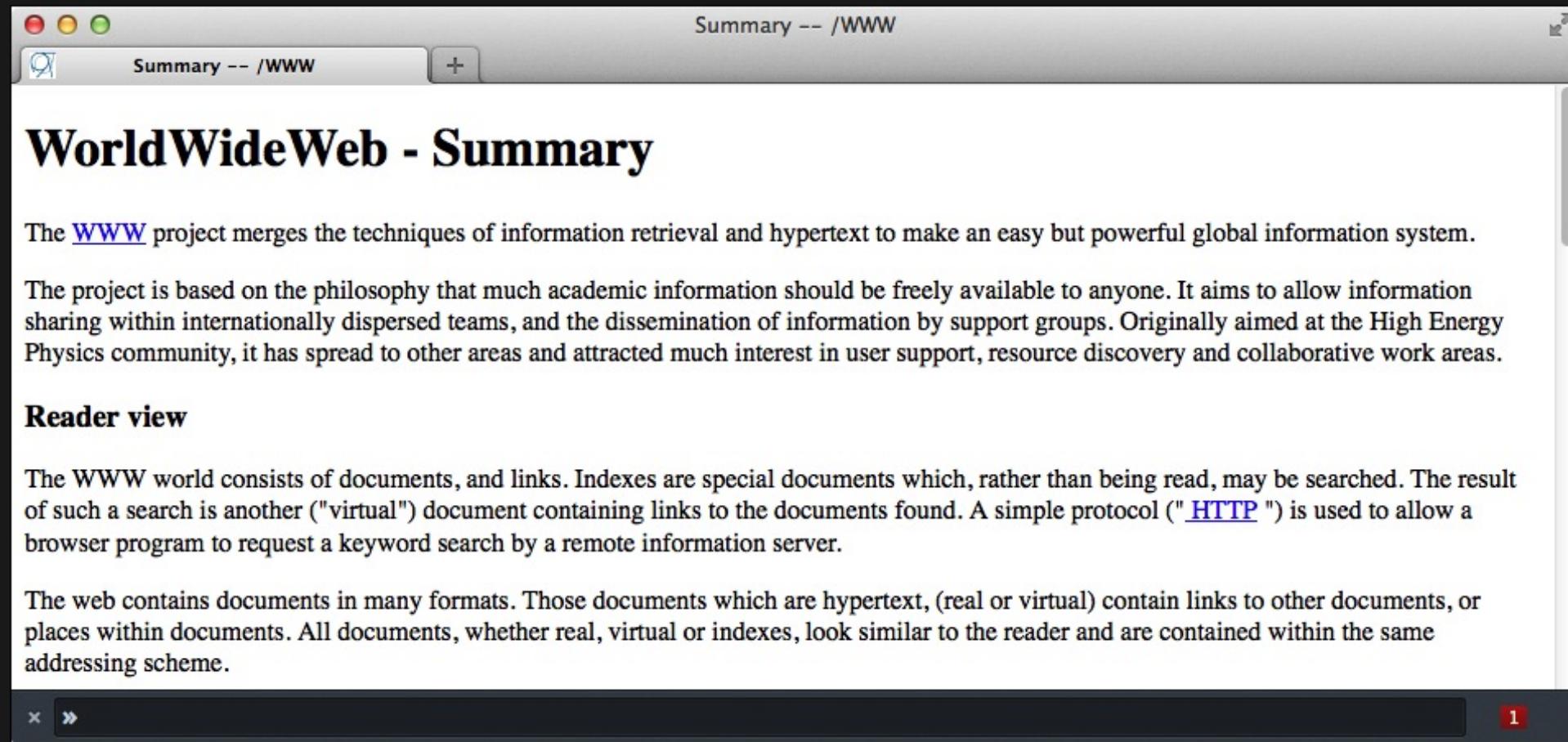
- Viewport-relative CSS units `vw`, `vh`, `vmin` and `vmax`
- CSS Media Queries, responsive design
- Responsive Images: `picture`, `srcset` (or: SVG)
- JS-based: Web Components

Technologies in progress:

- CSS Device Adaptation (could replace viewport meta tag)
- CSS Mobile Text Size Adjustment
- ...

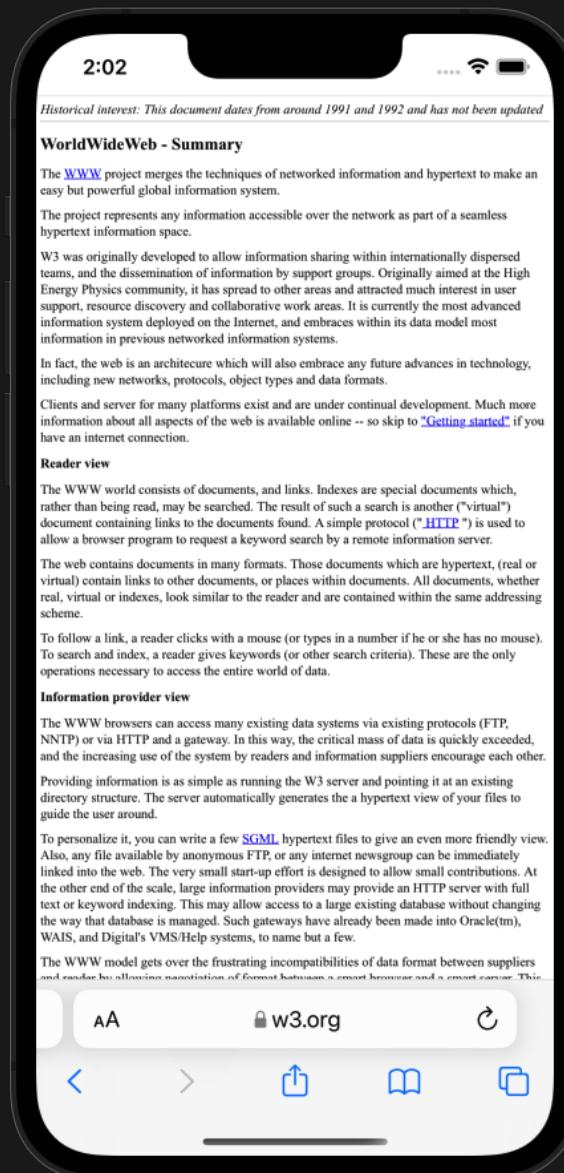
<https://w3c.github.io/web-roadmaps/mobile/adaptation.html>

WEBSITE FROM 1992 – RESPONSIVE?



<http://www.w3.org/Summary.html>

RESPONSIVE, BUT...



- Mobile browsers assume a page width of about 960px
- And scale the page respectively...

VIEWPORT



- **Viewport:** area of the browser in which the website is displayed
- Desktop websites typically assume a viewport width of about 960px
- Less space available on smartphones, e.g. 320px width
- Scaling necessary for desktop optimized pages

VIEWPORT

- Desktop optimized website: scaling suitable
- Mobile optimized website: scaling harmful
- Viewport statement to the rescue...

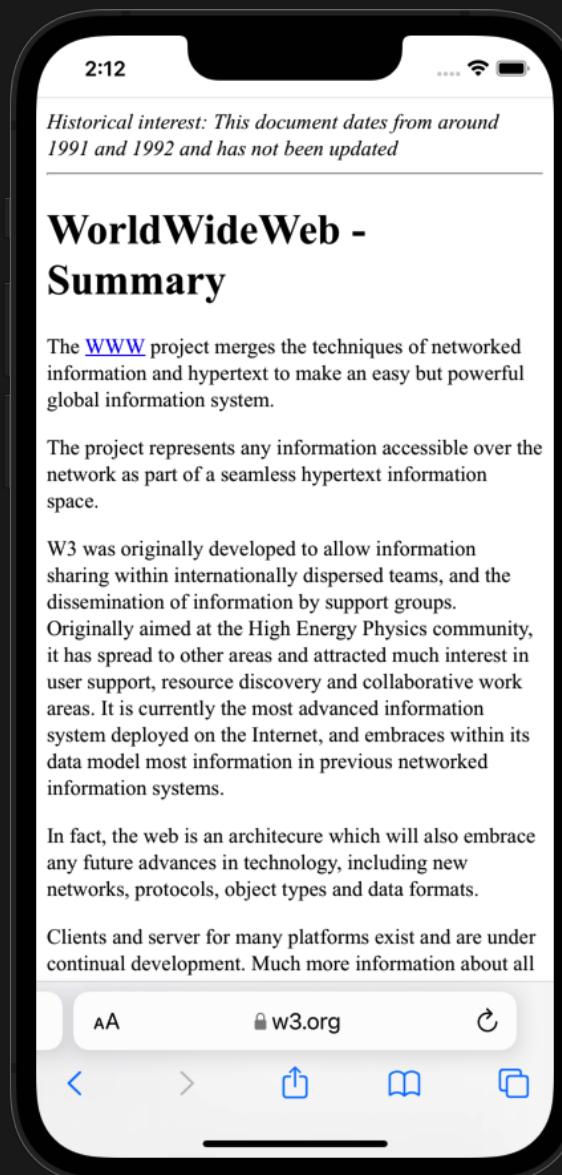
```
<meta name="viewport" content="width=320">  
<meta name="viewport" content="width=device-width">
```

VIEWPORT

```
<meta name="viewport" content="?">
```

content value	meaning
width	viewport width
initial-scale	initial zoom factor, default: 1
minimum-scale maximum-scale	minimum and maximum zoom factor
user-scalable:no	user cannot scale the page

WEBSITE FROM 1992



- With viewport statement
- Perfectly usable

FLUID LAYOUT

- Layout and container width adjusts to viewport width
- Main container with `max-width` (or percentage)
- Other sizes in percent

```
#container {  
    max-width: 960px; /* ... */  
}  
aside {  
    width: 29.16666667%; /* 280/960x100 */  
    padding: 2.08333333%; /* 20/960x100 */  
}
```

<http://media.kulturbanause.de/blog/2013/01/fluessiges-layout.html>

<http://resources.sameerast.com/responsive-web-design-formula-easy-calculator.html>

FLUID LAYOUT

- Images should not stick out of the surrounding container

```
img {  
    max-width: 100%;  
}
```

- But:
On small viewports the fluid layout is barely usable, too
- In that case the layout needs to be adjusted
- Tool: [CSS Media Queries](#)

CSS MEDIA QUERIES

```
<link rel="stylesheet" media="screen and (min-width: 400px)"  
      href="large.css">
```

```
@import url(large.css) screen and (min-width: 400px);  
@media screen and (min-width: 400px) { ... }
```

CSS MEDIA QUERIES

```
@media all and (min-width:500px) { }
@media (min-width:500px) { }
@media not screen and (color) { }
@media only screen and (color) { }
@media print and (min-width: 25cm) { }
@media screen and (min-width: 400px) and (max-width: 700px) { }
@media screen and (device-width: 800px) { }
@media screen and (device-height: 600px) { }
@media all and (orientation: portrait) { }
@media all and (orientation: landscape) { }
@media screen and (device-aspect-ratio: 16/9) { }
@media print and (min-resolution: 300dpi) { }
@media tv and (scan: progressive) { }
@media tv and (scan: interlace) { }
```

<https://www.w3.org/TR/css3-mediaqueries/>
<https://drafts.csswg.org/mediaqueries-5/>

RESPONSIVES WEBDESIGN

- Use media queries to adjust layout to the device's capabilities
- Example: blocks side by side if sufficient viewport width is available, one above the other otherwise

```
@media screen and (min-width: 480px) {  
    .column {  
        float: left;  
    }  
}
```

RESPONSIVE WEBDESIGN

- Ethan Marcotte: Fluid Grids
- Ethan Marcotte: Responsive Web Design

Examples

- <https://responsivedesign.is/examples/>
- <http://2012.buildconf.com>
- <https://alistapart.github.io/code-samples/responsive-web-design/ex/ex-site-larger.html>

MOBILE DISPLAY RESOLUTIONS

Device	Resolution
Typical desktop monitor	90-100 ppi
Original iPhone	164 ppi
Nexus One	266 ppi
iPhone 4	329 ppi
Google Pixel 4	441 ppi
iPhone 12 Pro	460 ppi
Samsung Galaxy Note10+	498 ppi

<https://www.sizescreens.com>

WHAT IS A PIXEL?

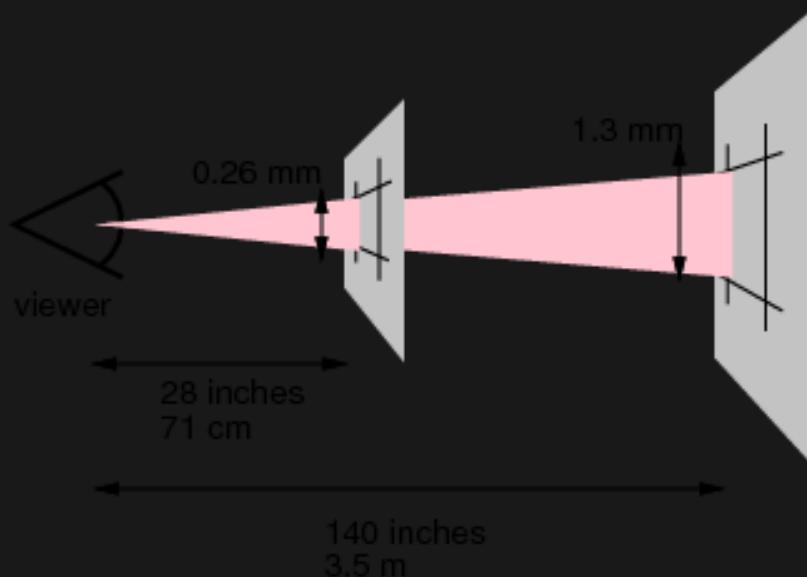
- According to the CSS 2.1 specification px is an absolute unit:

unit	description
pt	points – the points used by CSS are equal to 1/72nd of 1in
px	pixel units – 1px is equal to 0.75pt

- But: user agent can define the actual size
- Print: based on physical units like cm
- Screen: based on a *reference pixel*

REFERENCE PIXEL

The *reference pixel* is the visual angle of one pixel on a device with a pixel density of 96dpi and a distance from the reader of an arm's length.
[\(CSS 2.1 specification\)](#)



- Size of 1px based on this reference pixel
- Seems to be the same size
- Actually sizes can differ considerably

CSS PIXEL

- Screen size of 1px is based on the reference pixel
- Units like `cm` are calculated based on this pixel size
- The actual size of a CSS `cm` on a screen may differ from the common length unit *cm*
- Example: iPhone 12 Pro, 460 ppi, 2532×1170 px
1 CSS pixel equals 3 screen pixels

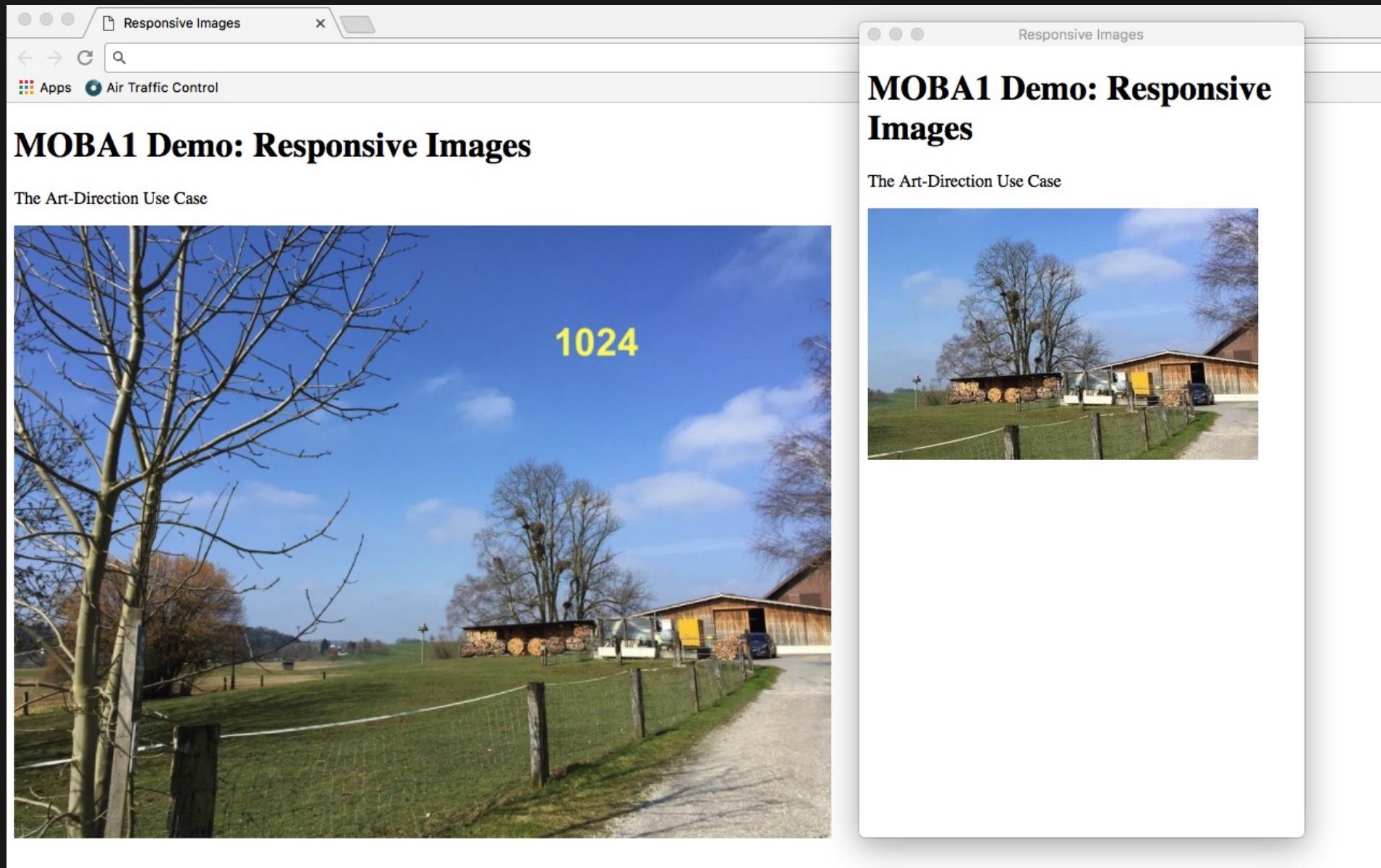
IMAGE RESOLUTION

- Example: JPEG 300x300px
- Placed on a website in a 300x300px box (CSS pixels)
- Capabilities of HD displays given away
- Options:
 - Provide image in higher resolution, scale in browser
 - Background image: provide various resolutions
 - Use scalable images (SVG, icon fonts)
 - Responsive images: `picture` and `srcset`

RESPONSIVE IMAGES

```
<picture>
  <source media="(min-width: 36em)"
    srcset="large.jpg 1024w, medium.jpg 640w, small.jpg 320w"
    sizes="50vw" />
  <source srcset="cropped-large.jpg 2x, cropped-small.jpg 1x" />
  
</picture>
```

RESPONSIVE IMAGES



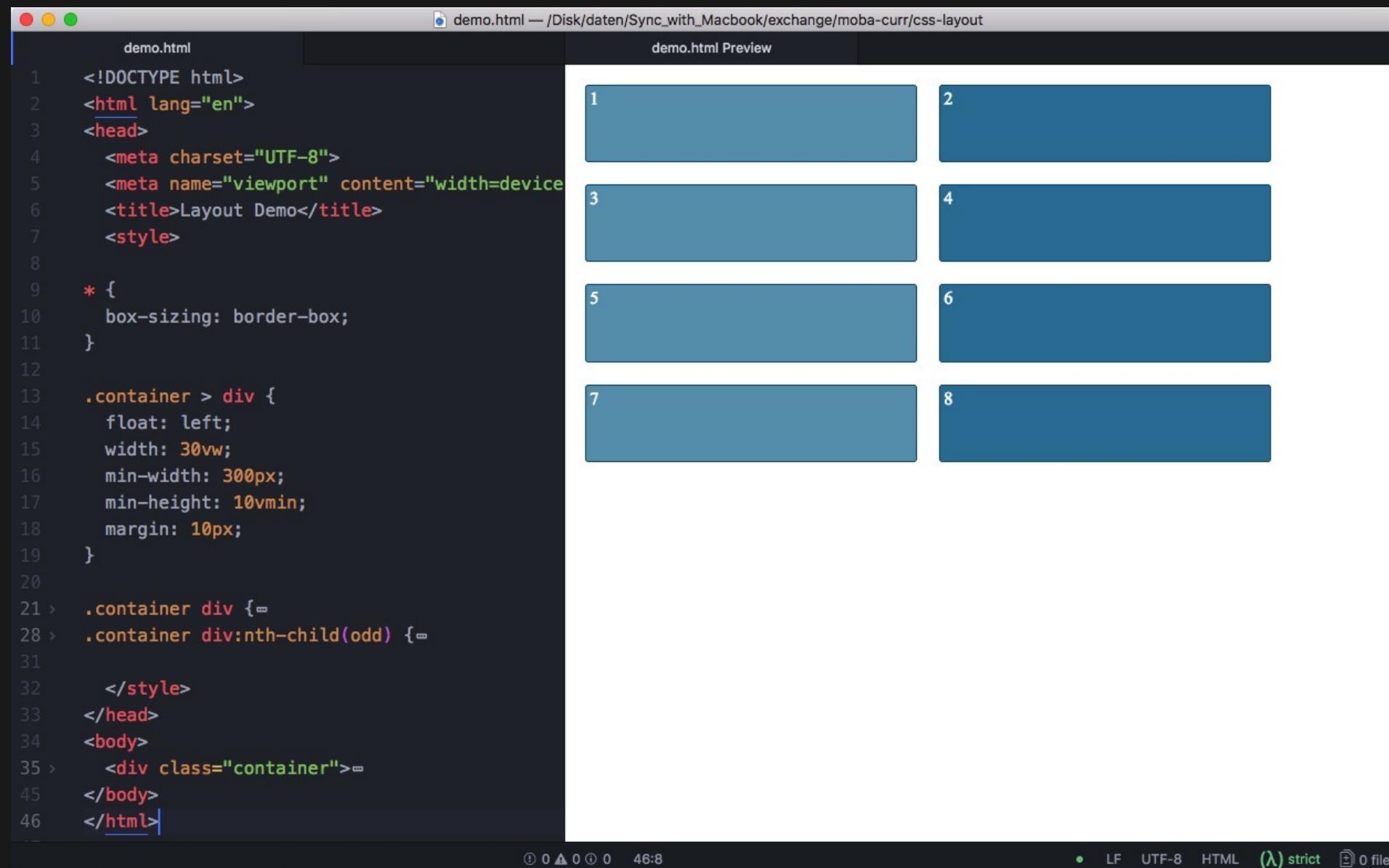
OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: **CSS Flexible Box Layout Module**
- Layouts: CSS Grid Layout
- Final Remarks

LAYOUT OPTIONS

- Traditional: CSS float, box model
- Or: inline-block, table-cell
- CSS Flexible Box Layout
- CSS Grid Layout
- Library/Framework (e.g., Bootstrap)

CSS LAYOUT: FLOAT



The screenshot shows a code editor with two tabs: "demo.html" and "demo.html Preview". The "demo.html" tab displays the following CSS code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Layout Demo</title>
7   <style>
8
9     * {
10       box-sizing: border-box;
11     }
12
13     .container > div {
14       float: left;
15       width: 30vw;
16       min-width: 300px;
17       min-height: 10vmin;
18       margin: 10px;
19     }
20
21   > .container div { ...
22   > .container div:nth-child(odd) { ...
23
24     </style>
25   </head>
26   <body>
27     <div class="container">...
28   </body>
29   </html>
```

The "demo.html Preview" tab shows a layout consisting of eight blue rectangular boxes arranged in a 4x2 grid. The boxes are labeled 1 through 8. Boxes 1, 2, 5, and 7 are in the top row, while boxes 3, 4, 6, and 8 are in the bottom row. This demonstrates how floating elements are positioned relative to each other.

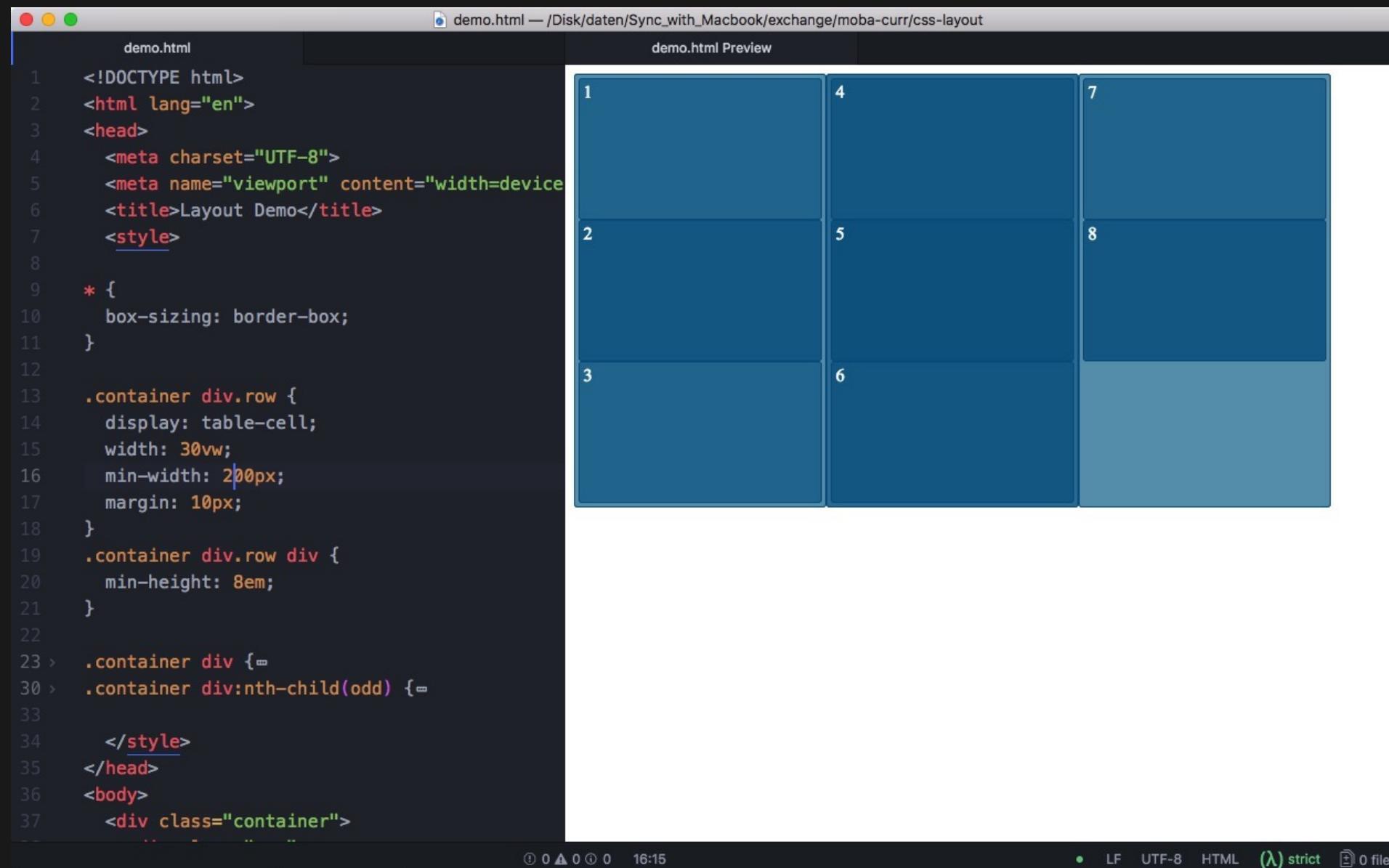
CSS LAYOUT: INLINE-BLOCK

The screenshot shows a code editor with two tabs: "demo.html" and "demo.html Preview". The "demo.html" tab displays the following CSS code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Layout Demo</title>
7   <style>
8
9     * {
10       box-sizing: border-box;
11     }
12
13     .container > div {
14       display: inline-block;
15       width: 30vw;
16       min-width: 300px;
17       min-height: 10vmin;
18       margin: 10px;
19     }
20
21   <.container div { = =
22   <.container div:nth-child(odd) { = =
23
24     </style>
25   </head>
26   <body>
27     <div class="container"> = =
28   </body>
29   </html>
```

The "demo.html Preview" tab shows eight blue rectangular boxes arranged in a 4x2 grid. The boxes are numbered 1 through 8. Boxes 1, 2, 5, and 7 are in the top row, while boxes 3, 4, 6, and 8 are in the bottom row. This visual representation demonstrates how multiple inline-block elements are rendered side-by-side by the browser.

CSS LAYOUT: TABLE-CELL



The screenshot shows a code editor with two tabs: "demo.html" and "demo.html Preview". The "demo.html" tab displays the following CSS code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Layout Demo</title>
7   <style>
8
9   * {
10     box-sizing: border-box;
11   }
12
13 .container div.row {
14   display: table-cell;
15   width: 30vw;
16   min-width: 200px;
17   margin: 10px;
18 }
19 .container div.row div {
20   min-height: 8em;
21 }
22
23 > .container div {
24 >   .container div:nth-child(odd) {
25
26     </style>
27   </head>
28   <body>
29     <div class="container">
30       <div class="row">
31         <div>1</div>
32         <div>4</div>
33         <div>7</div>
34       </div>
35       <div class="row">
36         <div>2</div>
37         <div>5</div>
38         <div>8</div>
39       </div>
40       <div class="row">
41         <div>3</div>
42         <div>6</div>
43       </div>
44     </div>
45   </body>
46 </html>
```

The "demo.html Preview" tab shows a grid layout with 8 numbered cells. The grid is 3 rows by 3 columns:

1	4	7
2	5	8
3	6	

CSS FLEXBOX

- Efficiently lay out, align and distribute items in a container
- Option to re-order items helps making responsive designs
- Used in React Native for laying out native apps
- Overall good support in current browsers

CSS Flexible Box Layout Module Level 1

Caniuse: Flexbox

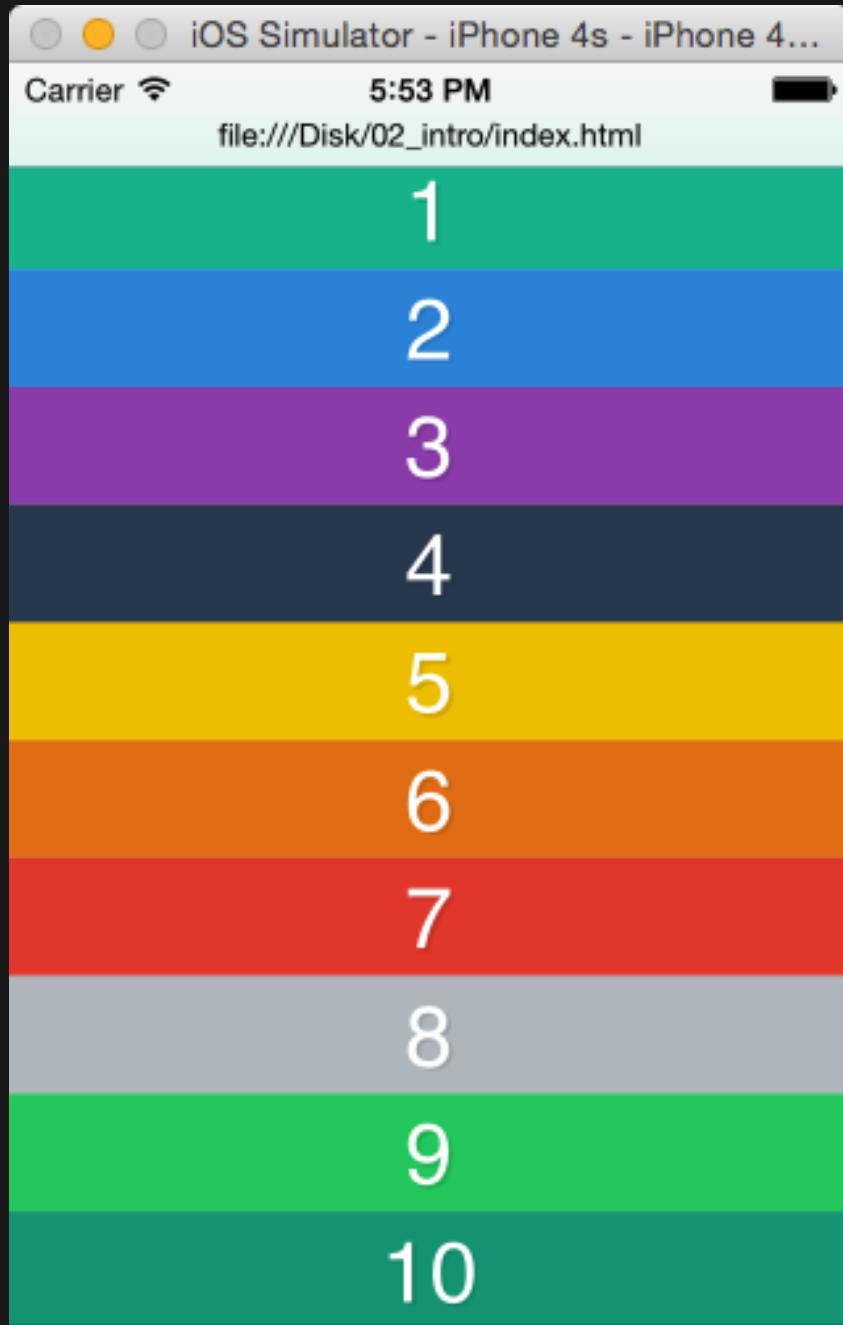
EXAMPLE

```
<!-- HTML -->
<div class="container">
  <div class="box box1">1</div>
  <div class="box box2">2</div>
  ...
  <div class="box box10">10</div>
</div>
```

```
/* CSS */
.box {
  padding: 10px;
  text-align: center;
  color: white;
  text-shadow: 4px 4px 0 rgba(0,0,0,0.1);
  font-size: 100px;
}
.box1 { background:#1abc9c; }
...
```

↓ preview ↓

EXAMPLE



EXAMPLE: FLEX CONTAINER

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    height: 100vh;  
}
```

- Flexbox consists of *flex containers* and *flex items*
- A flex container is declared with the `display` property
- Container is now a *flex container*
- Alternative that results in an inline element:

```
display: inline-flex
```

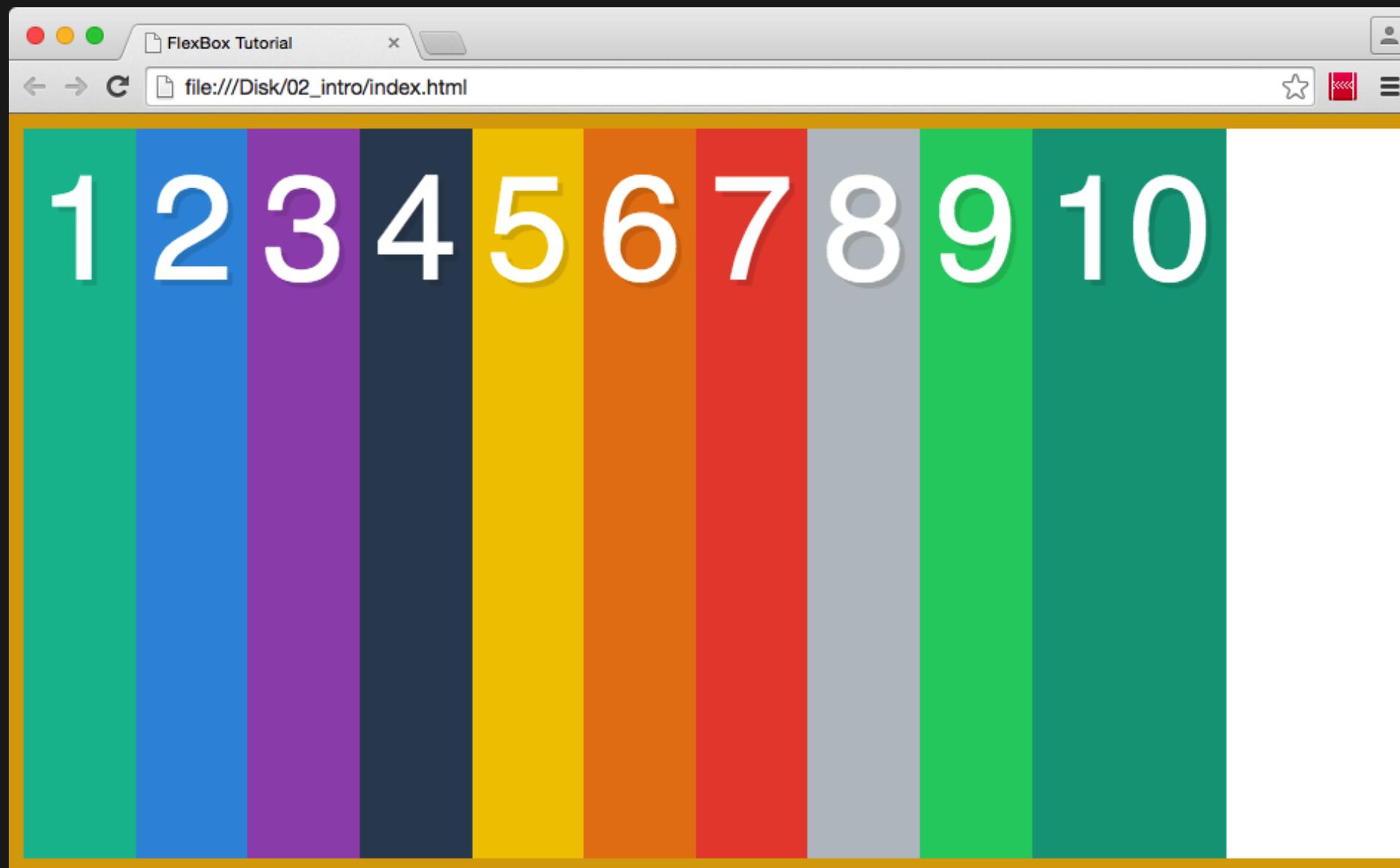
EXAMPLE: FLEX CONTAINER

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    height: 100vh;  
}
```

- 100vh – what's that ??
- It's the viewport height

↓ preview ↓

EXAMPLE: FLEX CONTAINER



FLEX ITEMS

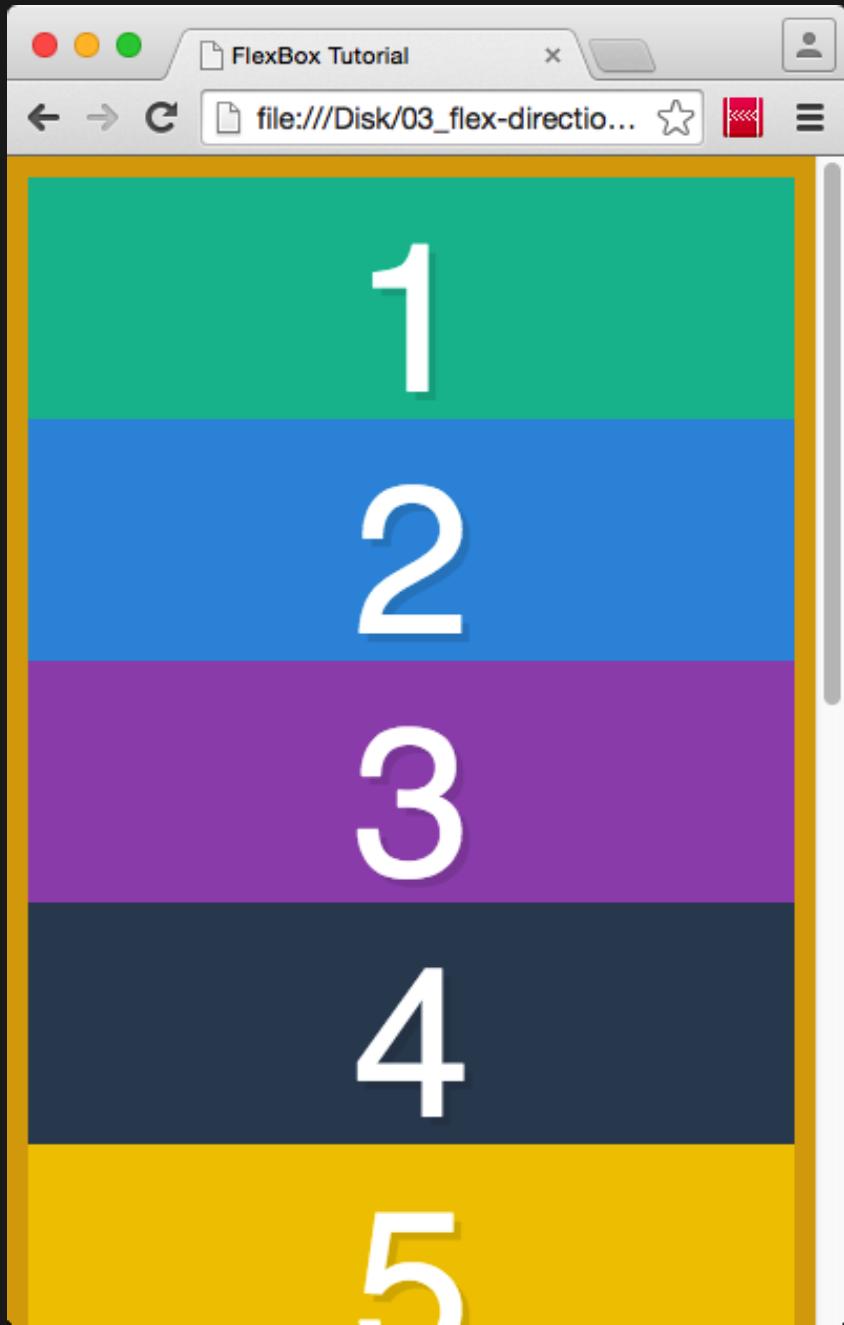
- Every child of a *flex container* is a *flex item*
- There can be any number of flex items
- Flexbox defines how flex items are laid out inside of flex containers

FLEX DIRECTION

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    min-height: 100vh;  
    flex-direction: column;  
}
```

↓ preview ↓

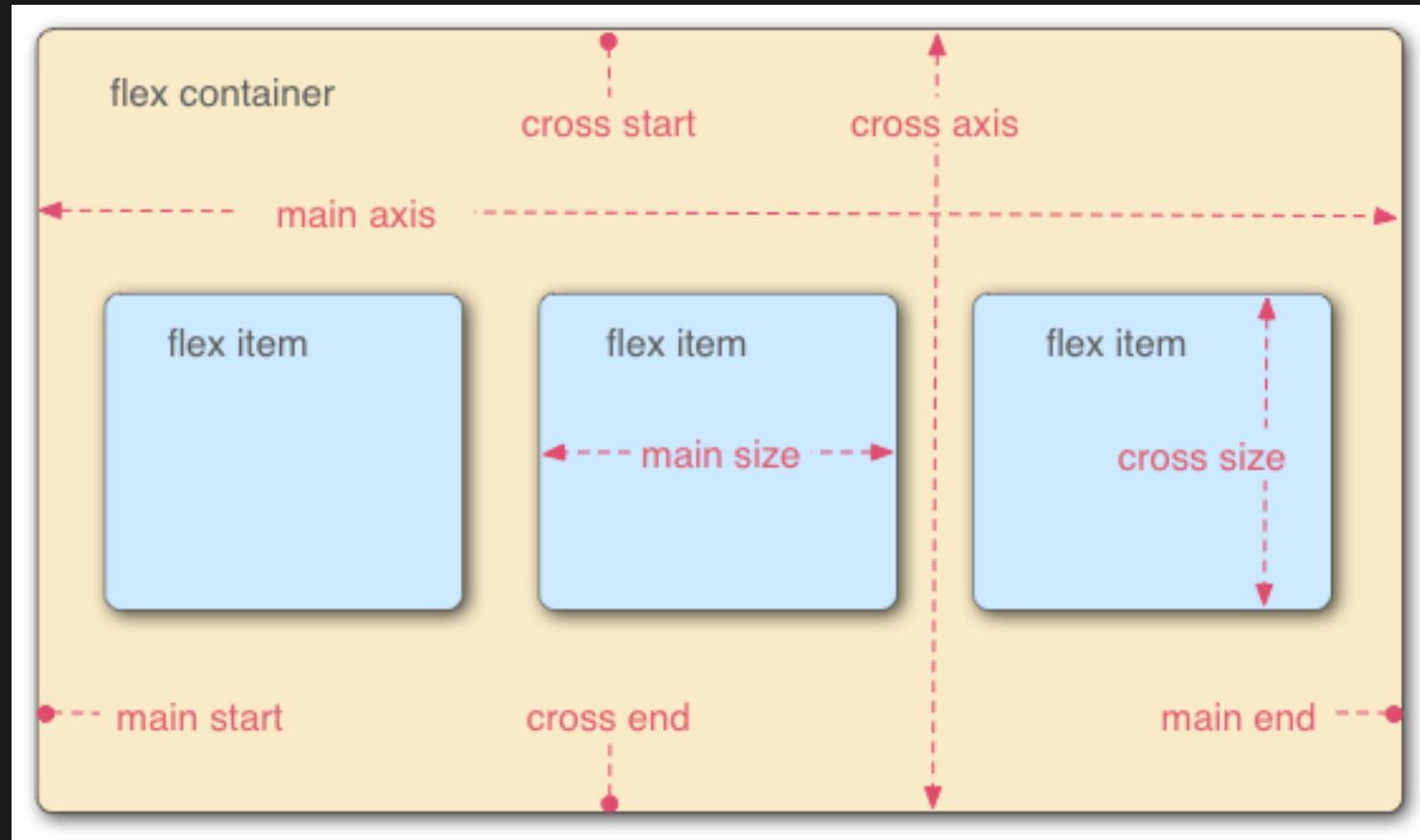
FLEX DIRECTION



FLEX DIRECTION

- CSS attribute: `flex-direction`
- Possible values: `row` (Default)
`row-reverse` | `column` | `column-reverse`
- Terminology:
 - Main direction (sometimes called the flow direction)
 - Main start and main end
 - Cross direction
 - Cross start and cross end

FLEX DIRECTION



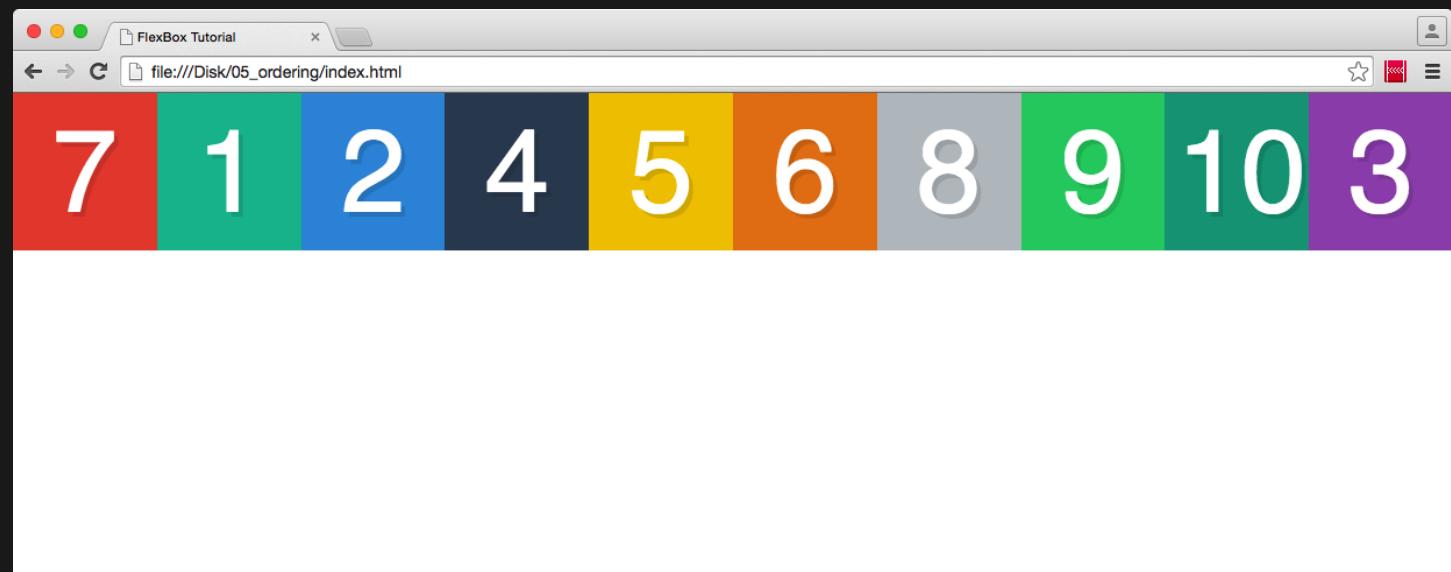
(Source: [MDN](#))

FLEX DIRECTION

```
body {  
  direction: rtl;      /* main direction right to left */  
}  
.container {  
  display: flex;  
  border: 10px solid goldenrod;  
  min-height: 100vh;  
  flex-direction: row-reverse; /* left to right again */  
}
```

FLEXBOX ORDERING

```
.container {  
    display:flex;  
}  
.box {  
    flex:1;  
    order:1;  
}  
.box3 {  
    order:3;  
}  
.box7 {  
    order:-2;  
}
```

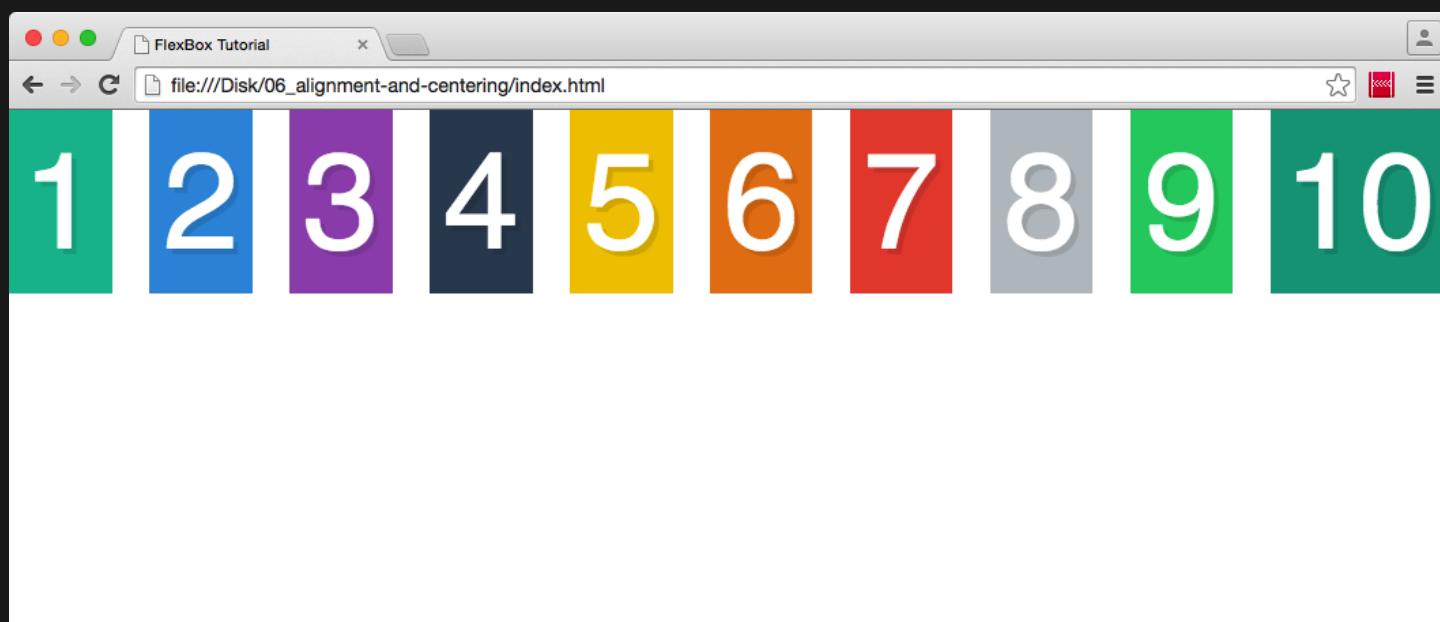


FLEXBOX ORDERING

- Flex items can be re-ordered with the `order` property
- Markup contents based on logical criteria and re-order as needed for various screen sizes
- Problems likely when selecting text over multiple items
- Initial value: 0

FLEXBOX ALIGNMENT

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



FLEXBOX ALIGNMENT

- Property: `justify-content`
- Defines the alignment along the *main axis*
- Possible values: `flex-start` (Default)
`flex-end` | `center` | `space-between` | `space-around`

FLEXBOX ALIGNMENT

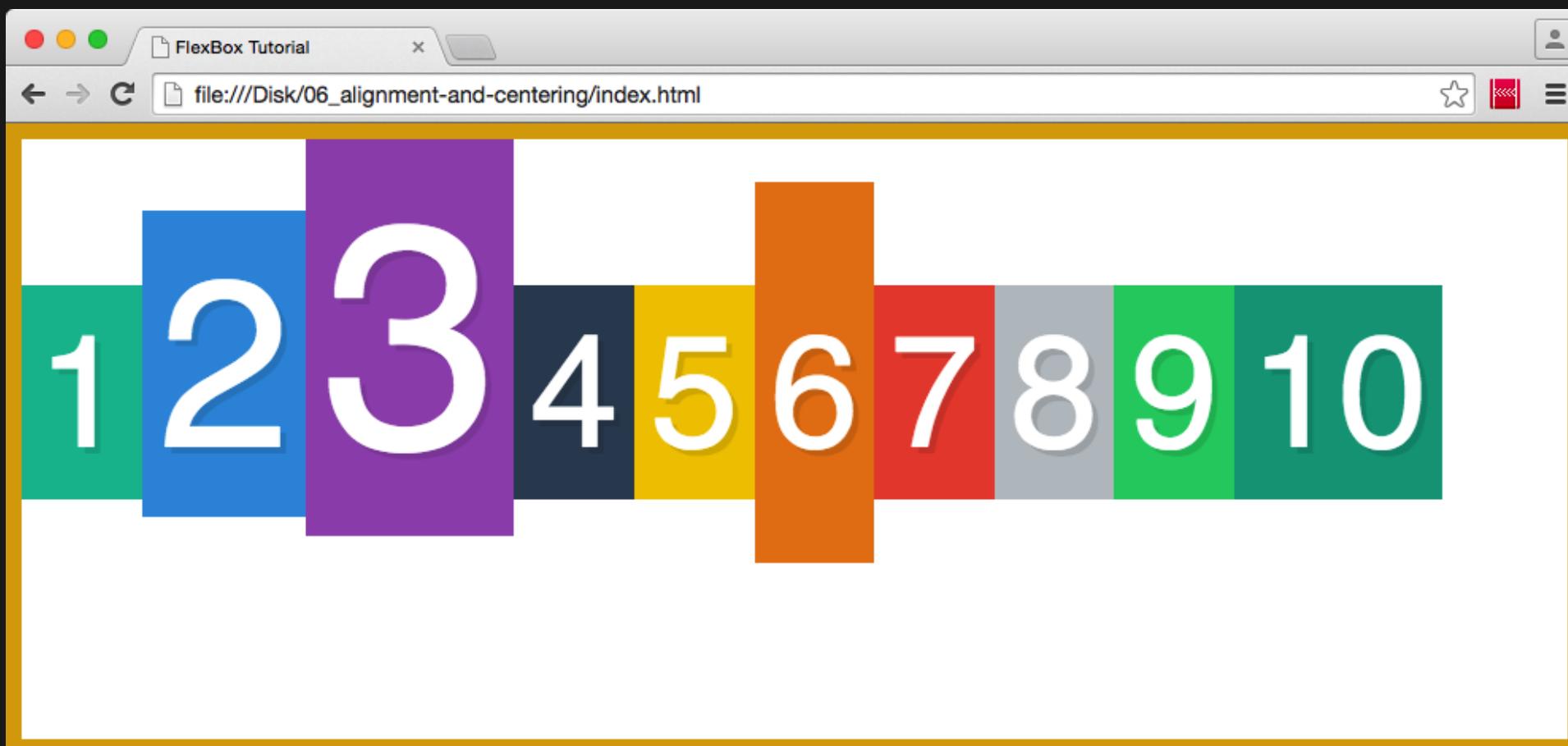
- Another property : align-items
- This one defines the alignment along the *cross axis*
- Possible values: stretch (Default)
flex-start | flex-end | center | baseline

FLEXBOX ALIGNMENT

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    min-height: 100vh;  
    align-items: baseline;  
}  
.box2 {  
    font-size: 150px;  
}  
.box3 {  
    font-size: 200px;  
}  
.box6 {  
    padding-bottom: 50px;  
    padding-top: 75px;  
}
```

↓ preview ↓

FLEXBOX ALIGNMENT



FLEXBOX ALIGNMENT

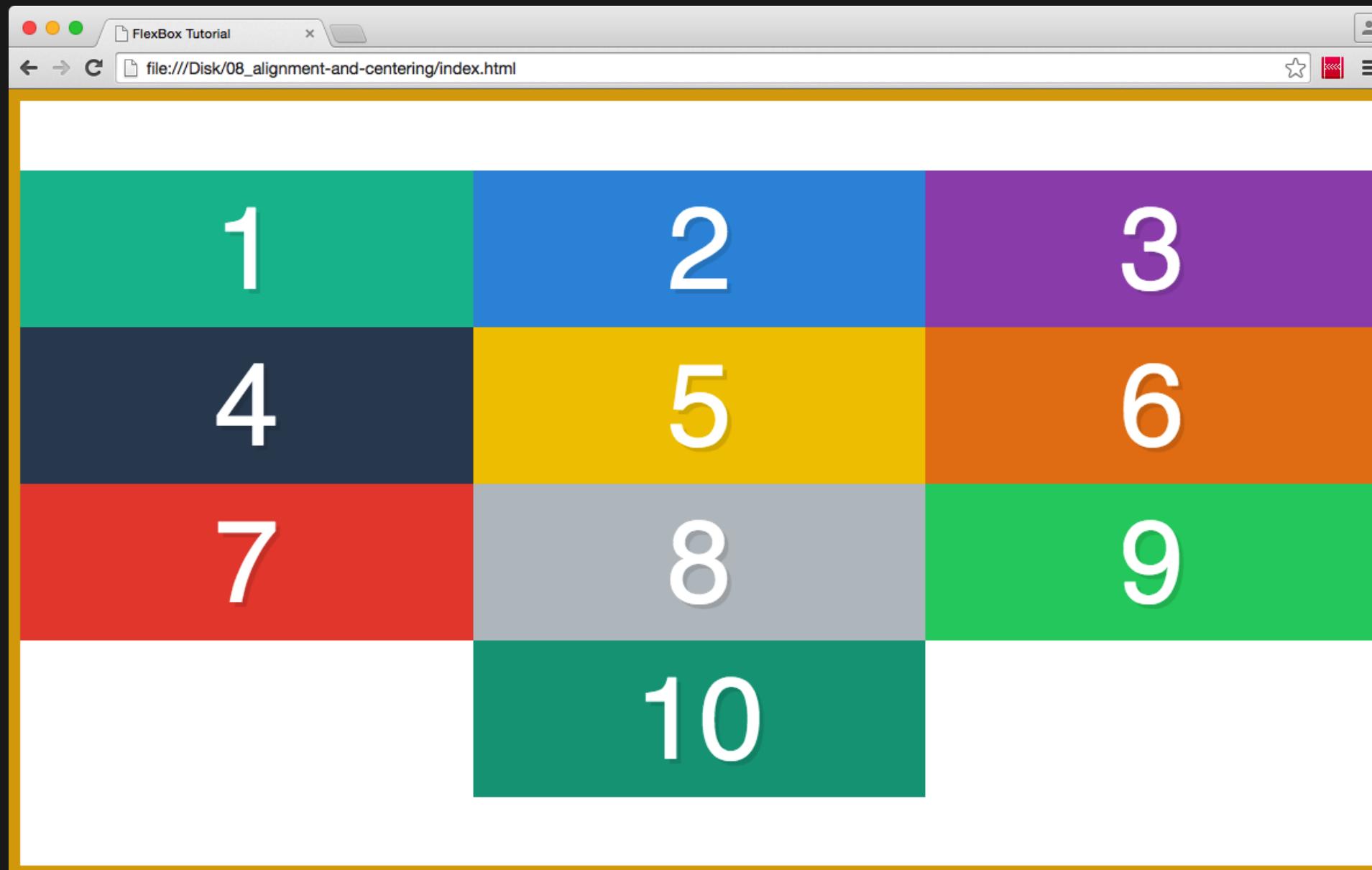
- One more property : align-content
- Alignment along the *cross axis* for the whole content
- Possible values: stretch (Default)
flex-start | flex-end | center | space-between |
space-around

FLEXBOX ALIGNMENT

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    min-height: 100vh;  
    flex-wrap: wrap;  
    justify-content: center;  
    align-content: center;  
}  
.box {  
    width: 33.33333%;  
}
```

↓ preview ↓

FLEXBOX ALIGNMENT

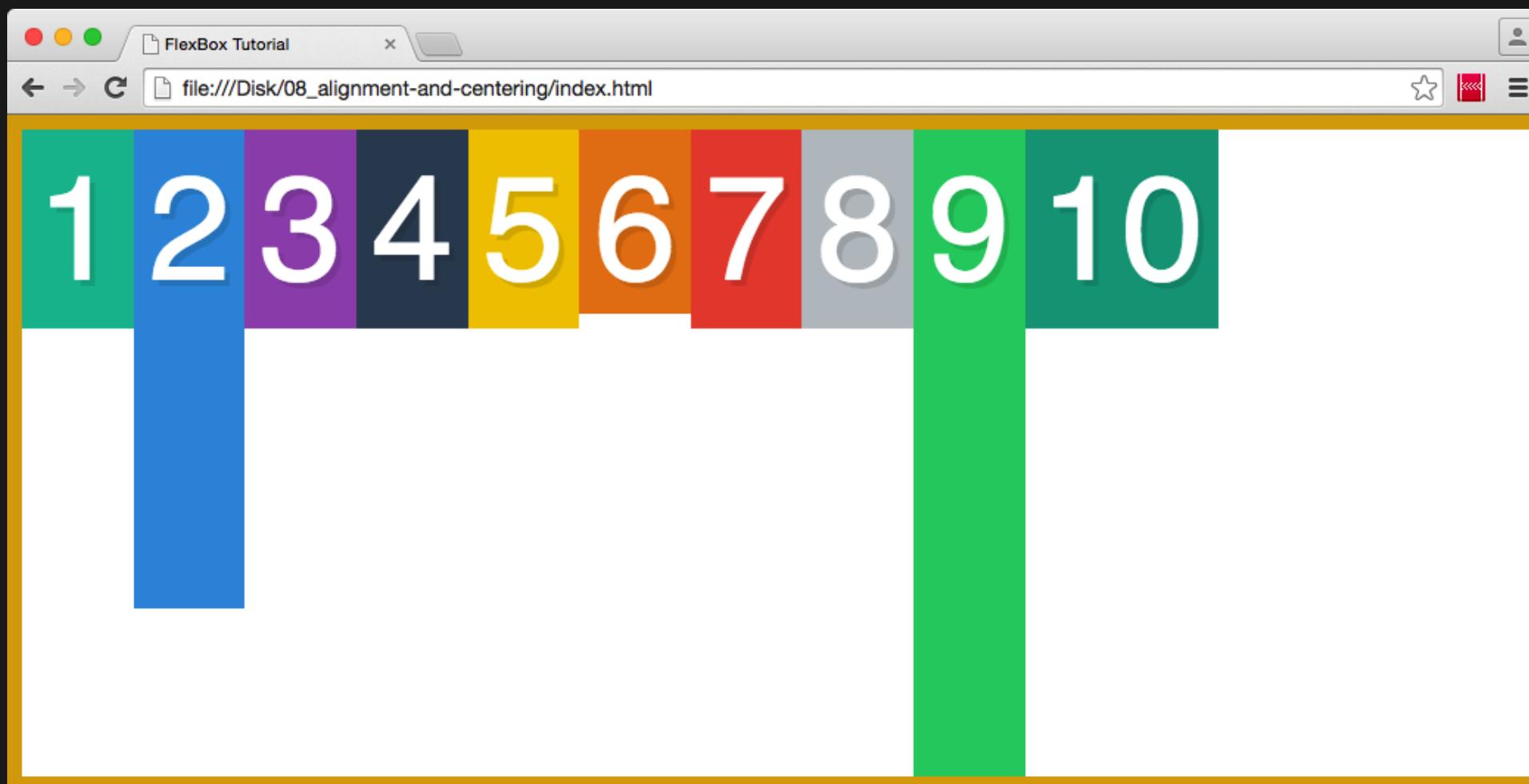


FLEXBOX SELF ALIGNMENT

```
.container {  
    display: flex;  
    border: 10px solid goldenrod;  
    min-height: 100vh;  
    align-items: flex-start;  
}  
.box2 {  
    padding-bottom: 200px;  
}  
.box6 {  
    padding-bottom: 0;  
}  
.box9 {  
    padding-bottom: 50px;  
    align-self: stretch;  
}
```

↓ preview ↓

FLEXBOX SELF ALIGNMENT



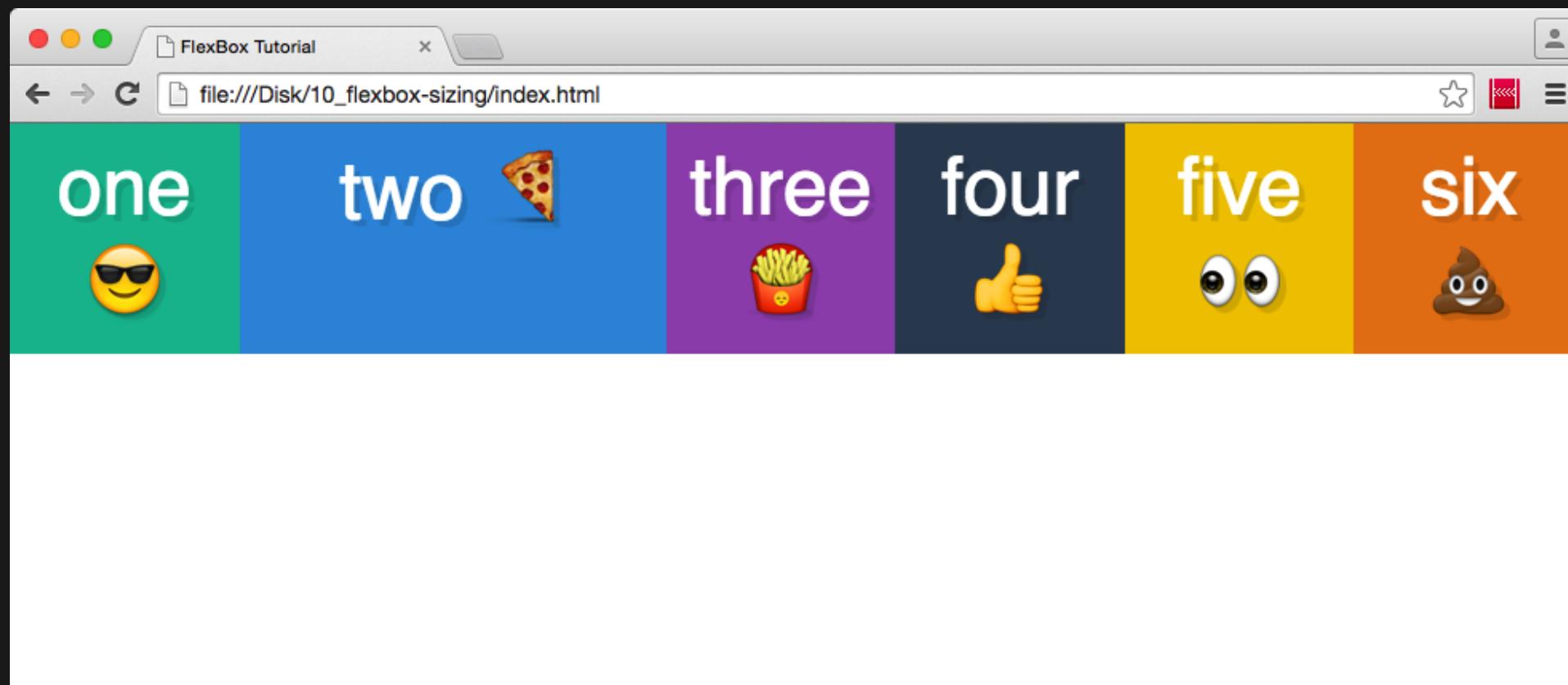
FLEXBOX SIZING

```
<!-- HTML -->
<div class="container">
  <div class="box box1">one 😎</div>
  <div class="box box2">two 🍕</div>
  <div class="box box3">three 🍟</div>
  ...
</div>
```

```
/* CSS */
.container {
  display: flex;
}
.box { /* ... */
  flex: 1;
}
.box2 {
  flex: 2;
}
```

↓ preview ↓

FLEXBOX SIZING



GROW AND SHRINK FLEX ITEMS

- **flex-grow**
 - Defines the ability for a flex item to grow if necessary
 - Unitless value that serves as a proportion
 - Default: 0
- **flex-shrink**
 - Defines the ability for a flex item to shrink if necessary
 - Default: 1
- **flex-basis**
 - Size of an element before the remaining space is distributed
 - Default: *auto*

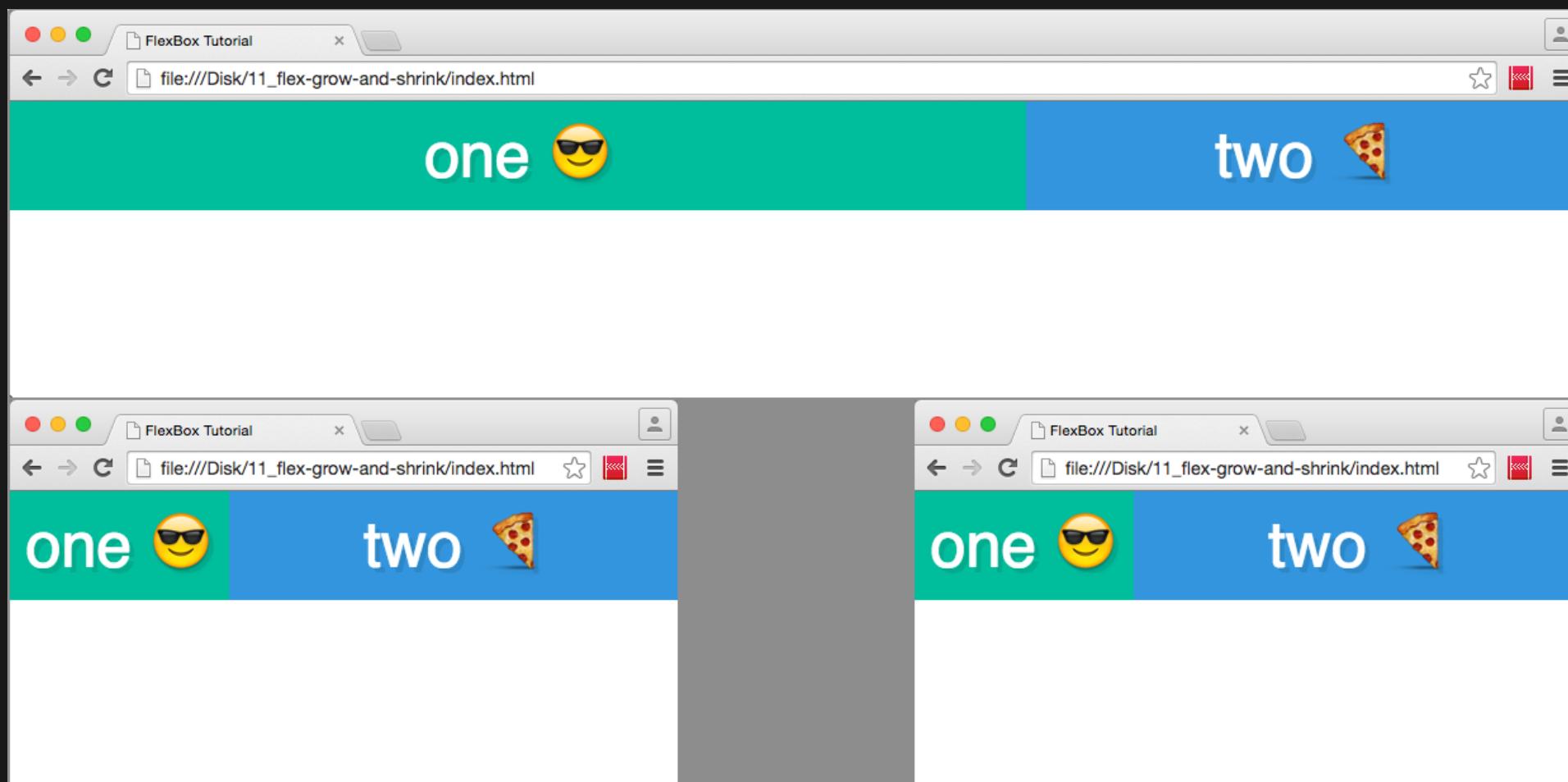
PROPERTY FLEX

```
.box1 {  
    flex: 10 5 400px;  
}  
.box2 {  
    flex: 1 1 400px;  
}
```

- Shorthand for these properties combined:
`flex-grow`, `flex-shrink`, `flex-basis`
- Second and third parameters are optional
- Default: `0 1 auto`

↓ preview ↓

GROW AND SHRINK FLEX ITEMS



RESIZING COMBINED WITH WRAPPING

- Resizing/wrapping combined allows for flexible layouts
- To demonstrate we use a container with six flex items

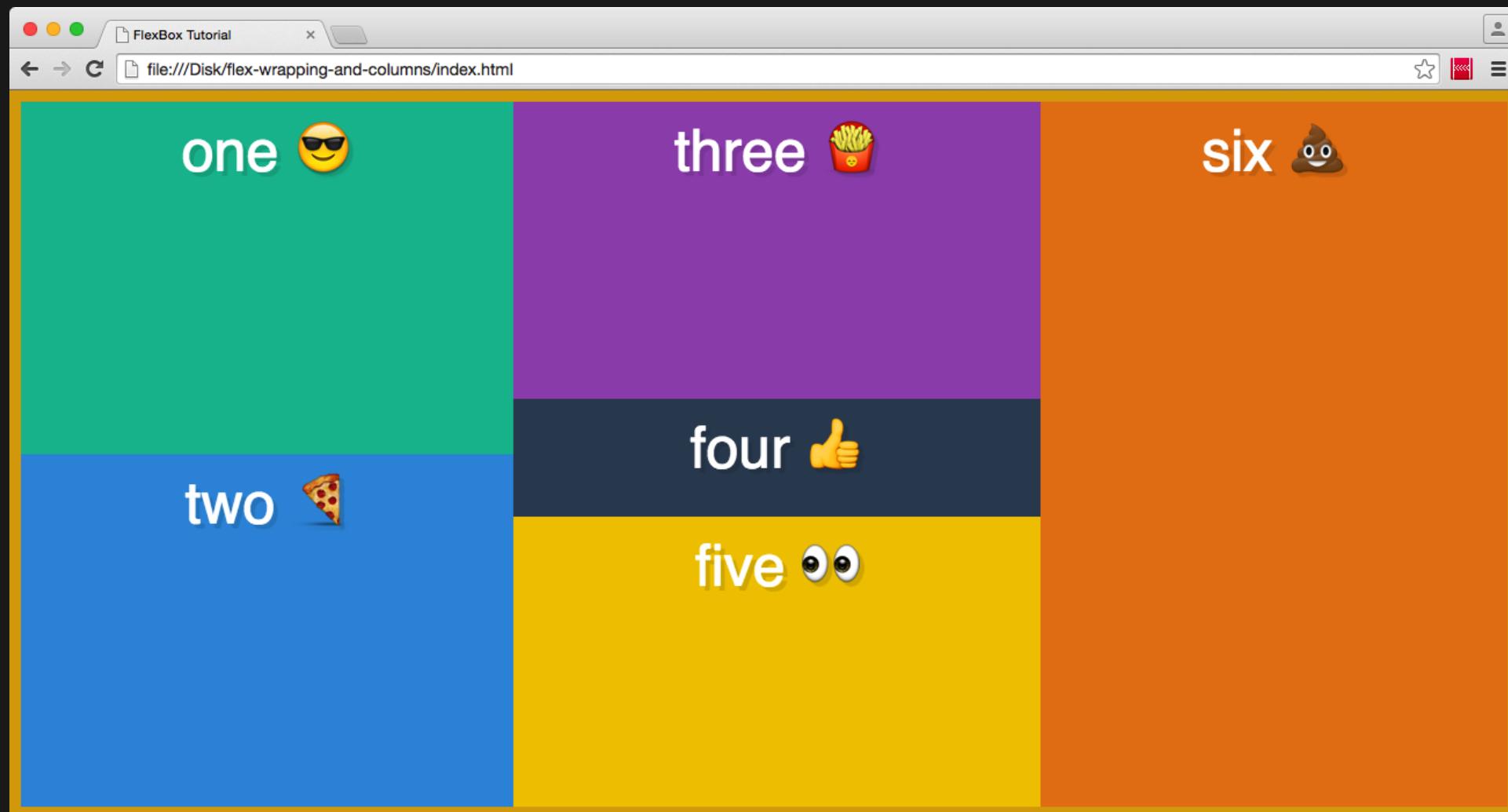
```
<div class="container">
  <div class="box box1">one 😎</div>
  ...
  <div class="box box6">six 💩</div>
</div>
```

RESIZING COMBINED WITH WRAPPING

```
.container {  
    display: flex;  
    flex-wrap: wrap;  
    flex-direction: column;  
    border: 10px solid goldenrod;  
    height: 100vh;  
}  
.box {  
    flex-basis: 250px;  
    flex-grow: 1;  
}  
.box3 {  
    flex-grow: 5;  
}  
.box4 {  
    flex-basis: 100px;  
}
```

↓ preview ↓

RESIZING COMBINED WITH WRAPPING



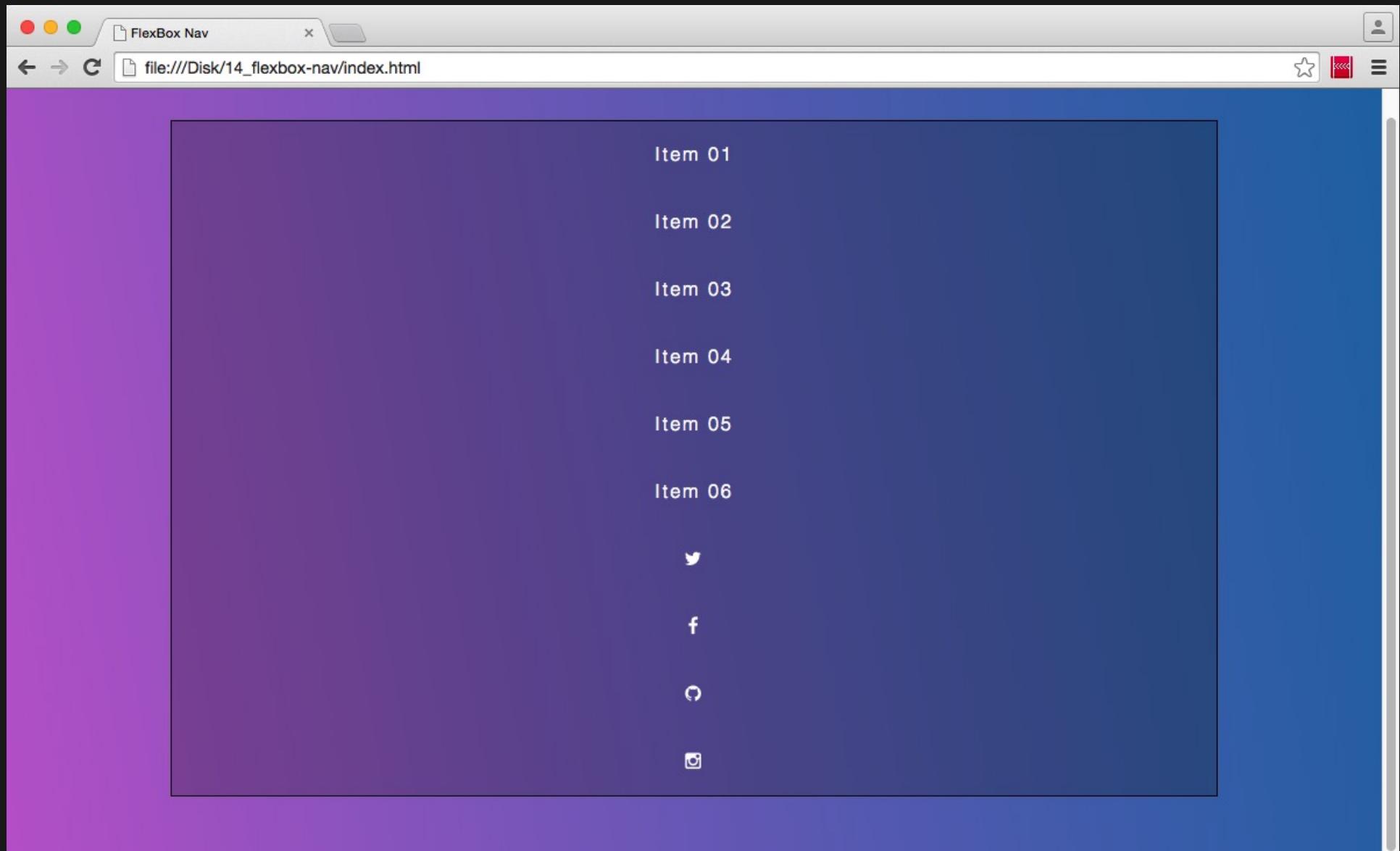
EXAMPLE: NAVIGATION

HTML

```
<nav class="flex-nav">
  <ul>
    <li><a href="#">Item 01</a></li>
    <li><a href="#">Item 02</a></li>
    ...
    <li class="social">
      <a href="http://twitter.com/wesbos"><i class="fa fa-twitter"></i></a>
    </li>
    ...
  </ul>
</nav>
```

↓ with a little bit of styling not shown here ↓

EXAMPLE: NAVIGATION

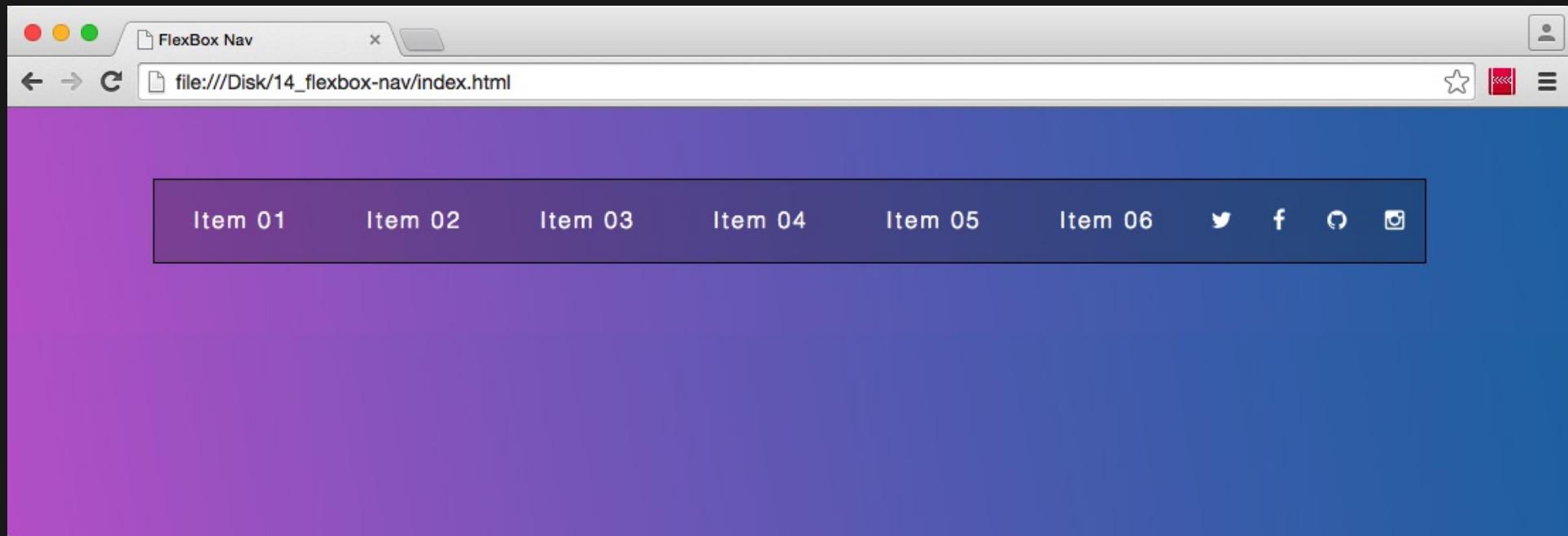


EXAMPLE: NAVIGATION

```
.flex-nav ul {  
    border: 1px solid black;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    display: flex;  
}  
.flex-nav li {  
    flex: 3;  
}  
.flex-nav .social {  
    flex: 1;  
}
```

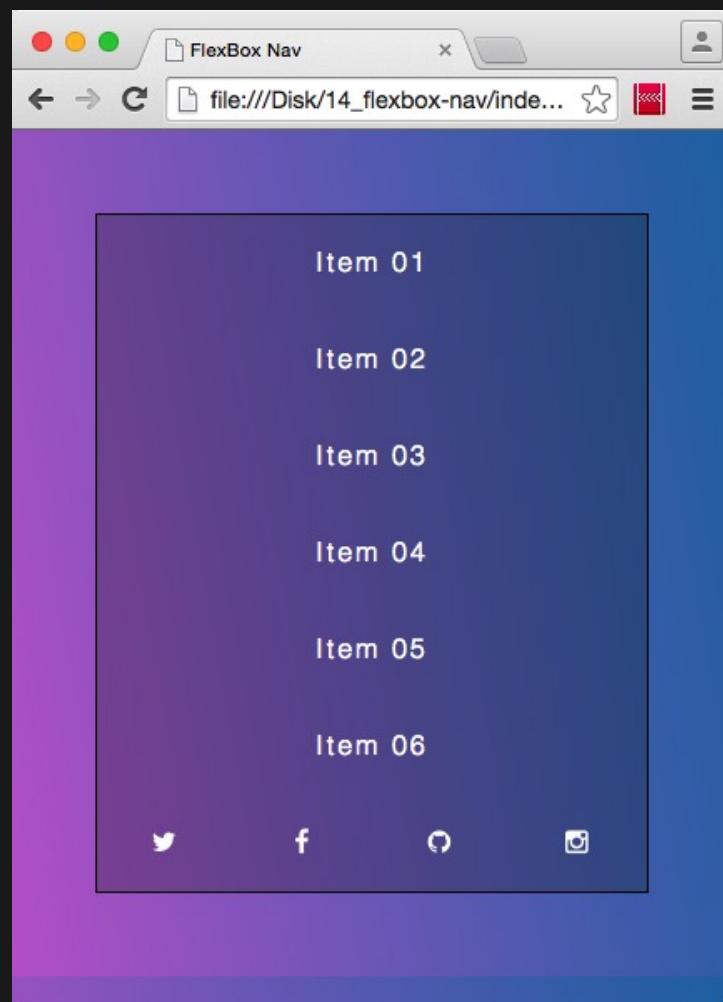
↓ preview ↓

EXAMPLE: NAVIGATION



↓ media queries for smaller screens ↓

EXAMPLE: NAVIGATION



EXAMPLE: MOBILE REORDERING

```
<body>
  <div class="wrapper">
    <header class="top">...</header>
    <nav class="flex-nav">...</nav>
    <section class="hero">...</section>
    <section class="details">...</section>
    <section class="signup">...</section>
    <footer>...</footer>
  </div>
  <script>...</script>
</body>
```

EXAMPLE: MOBILE REORDERING

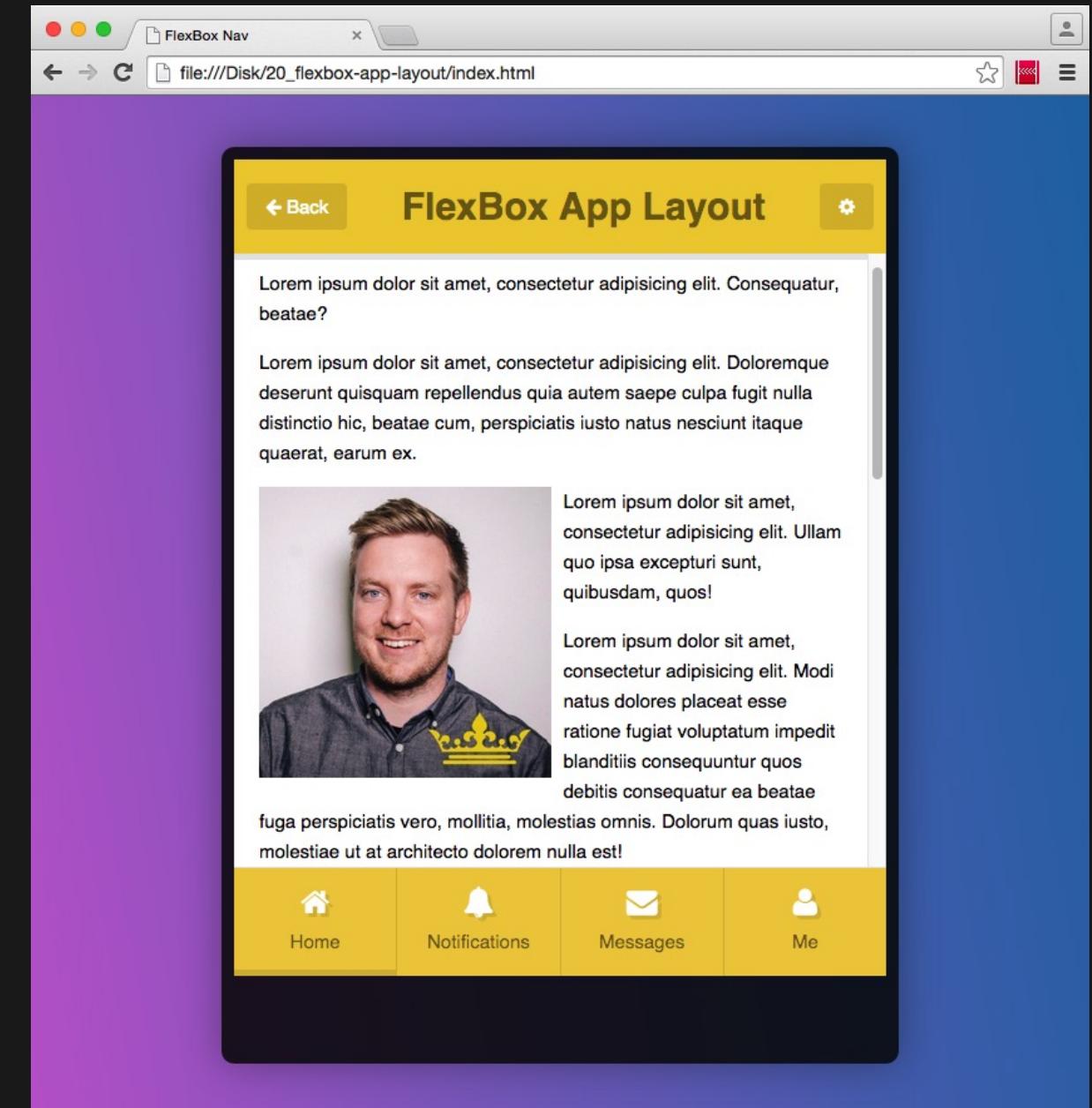
- On small mobile devices
 - the navigation should be on top of the screen
 - it could also be replaced by a toggle button
 - the details and signup areas should be moved up
- All this is easy with flexbox
- For re-ordering, the outer div must be *flex*

MOBILE APP LAYOUT

```
<div class="app-wrap">
  <header class="app-header">
    <a class="button">...</a>
    <h1>FlexBox App Layout</h1>
    <a class="button">...</a>
  </header>
  <div class="content">
    <p>...</p><p>...</p><img><p>...</p>
  </div>
  <div class="icon-bar">
    <a><i class="fa fa-home"></i>Home</a>
    <a><i class="fa fa-bell"></i>Notifications</a>
    <a><i class="fa fa-envelope"></i>Messages</a>
    <a><i class="fa fa-user"></i>Me</a>
  </div>
</div>
```

MOBILE APP LAYOUT

```
.app-wrap {  
  display: flex;  
  flex-direction: column;  
}  
.app-wrap > * {  
  flex: 1 1 auto;  
}  
.app-header {  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
}  
.content {  
  overflow-y: scroll;  
  -webkit-overflow-scrolling: touch;  
}  
.icon-bar {  
  display: flex;  
}  
.icon-bar a {  
  flex: 1;  
}
```

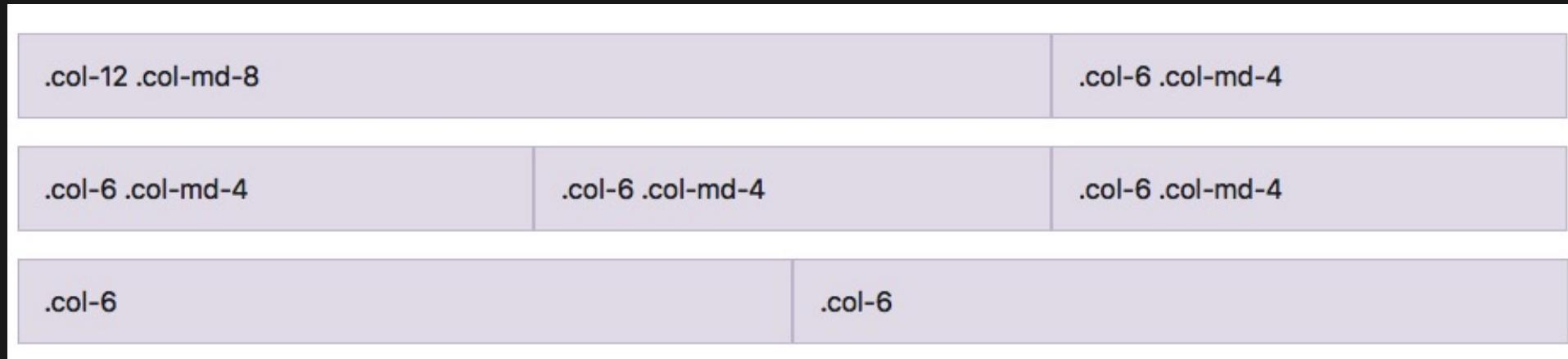


BOOTSTRAP

- Popular front-end component library
- Responsive grid system
- Prebuilt components and plugins
- Sass variables and mixins
- Grid system built with FlexBox

BOOTSTRAP GRID

```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-8">.col-12 .col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>
</div>
```

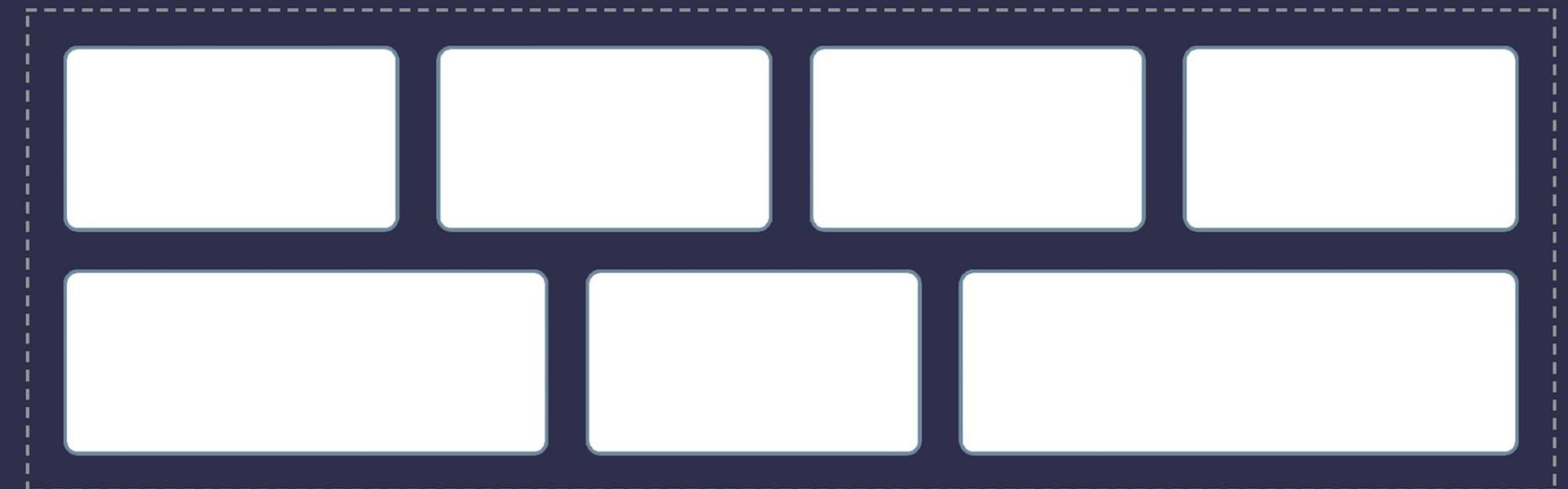


OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: CSS Flexible Box Layout Module
- Layouts: CSS Grid Layout
- Final Remarks

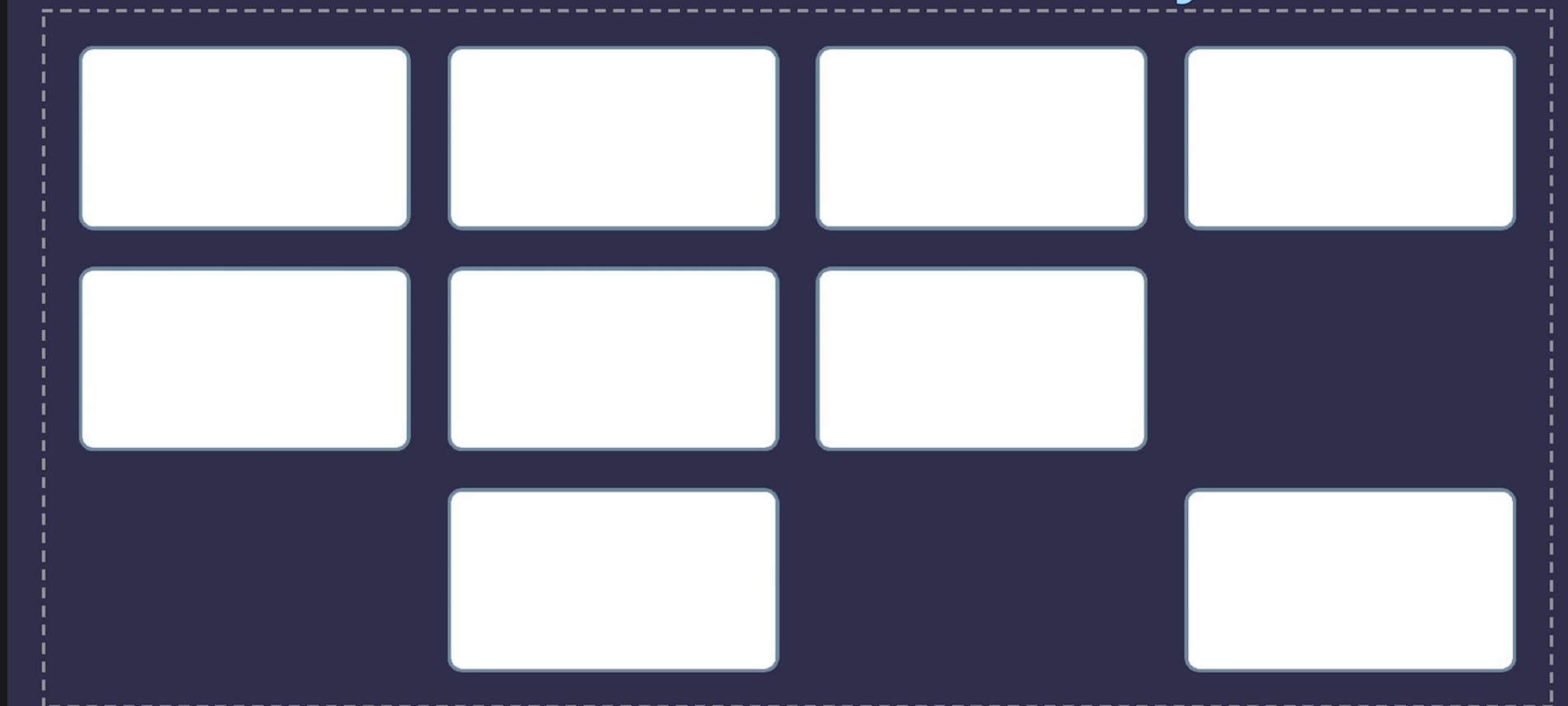
FLEXBOX

Flexbox is for one-dimensional layout



GRID

Grid is for two-dimensional layout



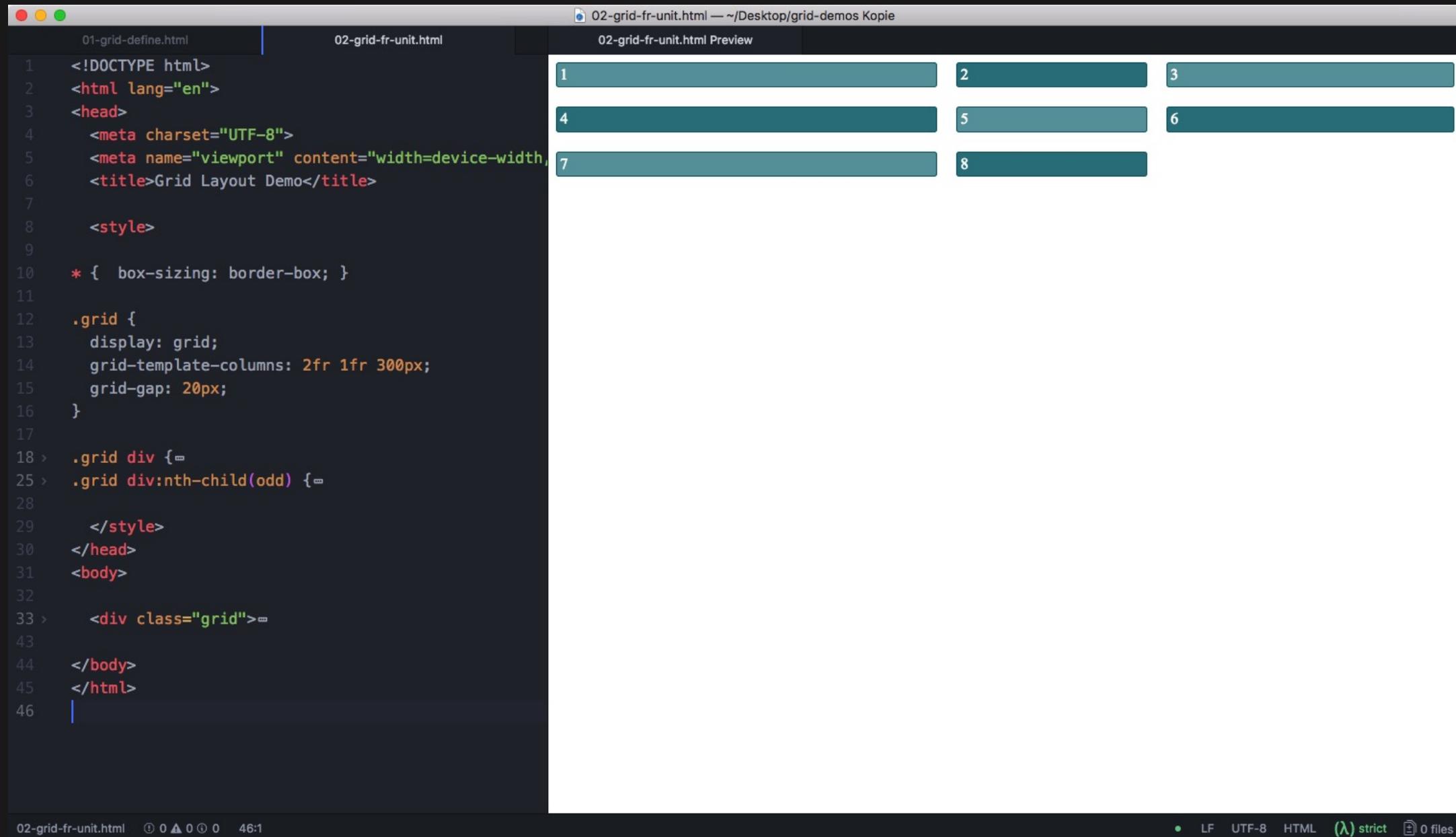
DEFINING A GRID

The screenshot shows a code editor with two tabs: "01-grid-define.html" and "01-grid-define.html Preview". The "01-grid-define.html" tab displays the following code:

```
01-grid-define.html
1 <title>Grid Layout Demo</title>
2
3 <style>
4 * { box-sizing: border-box; }
5
6 .grid {
7   display: grid;
8   grid-template-columns: 200px 200px 200px;
9   grid-template-rows: 200px 100px;
10  grid-auto-rows: 100px;
11  grid-gap: 20px;
12}
13
14 .grid div {
15   background-color: #4CAF50;
16   color: white;
17   padding: 10px;
18   text-align: center;
19 }
20
21 .grid div:nth-child(odd) {
22   background-color: #FF9800;
23   color: black;
24   text-align: right;
25 }
26
27 </style>
28
29 </head>
30
31 <body>
32
33   <div class="grid">
34     <div>1</div>
35     <div>2</div>
36     <div>3</div>
37     <div>4</div>
38     <div>5</div>
39     <div>6</div>
40     <div>7</div>
41     <div>8</div>
42   </div>
43
44 </body>
45
46 </html>
```

The "01-grid-define.html Preview" tab shows a 2x4 grid of eight teal-colored boxes, each containing a number from 1 to 8. The first three columns have a height of 200px, and the last column has a height of 100px. The boxes are separated by a 20px gap.

THE FR UNIT



The screenshot shows a code editor with two tabs open: "01-grid-define.html" and "02-grid-fr-unit.html". The "02-grid-fr-unit.html" tab is active, displaying the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Grid Layout Demo</title>
7
8     <style>
9
10    * { box-sizing: border-box; }
11
12    .grid {
13      display: grid;
14      grid-template-columns: 2fr 1fr 300px;
15      grid-gap: 20px;
16    }
17
18    .grid div {
19      background-color: #2e3436;
20      color: white;
21      padding: 10px;
22      text-align: center;
23    }
24
25    .grid div:nth-child(odd) {
26      background-color: #4c566a;
27    }
28
29    </style>
30  </head>
31  <body>
32
33    <div class="grid">
34
35      <div>1</div>
36      <div>2</div>
37      <div>3</div>
38
39      <div>4</div>
40      <div>5</div>
41      <div>6</div>
42
43      <div>7</div>
44      <div>8</div>
45
46    </div>

```

The preview window shows the resulting grid layout with 8 items arranged in a 2x4 grid:

1	2	3	
4	5	6	
7	8		

REPEAT NOTATION

The screenshot shows a code editor with three tabs: '01-grid-define.html', '03-grid-repeat.html', and '03-grid-repeat.html Preview'. The '03-grid-repeat.html Preview' tab is active, displaying a grid layout with 8 teal-colored boxes labeled 1 through 8. The layout consists of two rows: the top row contains boxes 1, 2, 3, 4, and 5; the bottom row contains boxes 6, 7, and 8. The '03-grid-repeat.html' tab shows the source code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6   <title>Grid Layout Demo</title>
7
8   <style>
9
10  * { box-sizing: border-box; }
11
12  .grid {
13    display: grid;
14    grid-template-columns: 100px repeat(2, 1fr 2fr);
15    grid-gap: 20px;
16  }
17
18 > .grid div {-
25 > .grid div:nth-child(odd) {-
28
29 |   </style>
30 </head>
31 <body>
32
33 >   <div class="grid">-
43
44 </body>
45 </html>
46
```

The status bar at the bottom indicates the file is '03-grid-repeat.html', has 0 changes, 0 errors, 0 warnings, and 29:1 lines. It also shows encoding as 'UTF-8', mode as 'HTML', strict mode as '(λ) strict', and 0 files.

AUTO-PLACEMENT AND ORDER

The screenshot shows a code editor with several tabs and a preview window.

Code Editor Tabs:

- 01-grid-define.html
- 03-grid-repeat.html
- 04-grid-order.html — ~/Desktop/grid-demos Kopie

Preview Window:

04-grid-order.html Preview

The preview shows a 2x4 grid of teal-colored boxes, each containing a number from 1 to 8. The layout is as follows:

Row	Column 1	Column 2	Column 3
1	2	3	7
2	4	5	8
3	1	6	

CSS Styles (from 04-grid-order.html):

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Grid Layout Demo</title>
7
8      <style>
9
10     * { box-sizing: border-box; }
11
12     .grid {
13         display: grid;
14         grid-template-columns: repeat(3, 1fr);
15         grid-template-rows: 100px 100px 100px;
16         grid-gap: 20px;
17         grid-auto-flow: column;
18     }
19
20     .grid div:nth-child(2) {
21         order: 1;
22     }
23     .grid div:nth-child(4) {
24         order: 2;
25     }
26
27     > .grid div { ...
28     > .grid div:nth-child(odd) { ...
29
30         </style>
31     </head>
32     <body>
33
34
35
36
37
38
39
39
40
41
42
43
44
```

File Status:

- 04-grid-order.html: ① 0 ▲ 0 ② 0 44:1
- LF UTF-8 HTML (λ) strict ④ 0 files

LINE-BASED PLACEMENT

The screenshot shows a code editor with two tabs: "05-grid-line-pos.html" and "05-grid-line-pos.html Preview". The code in "05-grid-line-pos.html" demonstrates line-based grid placement:

```
05-grid-line-pos.html
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
```

The "Preview" tab shows a 4x3 grid of four teal-colored boxes labeled 1 through 4. Box 1 is in the top-left position. Box 2 is in the second column of the first row. Box 3 is in the third column of the first row. Box 4 is in the third column of the second row, spanning two columns.

05-grid-line-pos.html — ~/Desktop/grid-demos Kopie

05-grid-line-pos.html Preview

1

2

3

4

05-grid-line-pos.html ① 0 ▲ 0 ① 0 46:1

• LF UTF-8 HTML (λ) strict ② 0 files

GRID TEMPLATE AREAS

The screenshot shows a code editor and a browser preview side-by-side. The code editor on the left displays the following CSS:

```
8  <style>
9
10 * { box-sizing: border-box; }
11
12 header { grid-area: hd; }
13 footer { grid-area: ft; }
14 article { grid-area: main; }
15 aside { grid-area: sidebar; }
16
17 .grid {
18   display: grid;
19   grid-gap: 20px;
20   grid-template-areas:
21     "hd"
22     "main"
23     "sidebar"
24     "ft"
25 }
26
27 @media (min-width: 640px) {
28   .grid {
29     grid-template-columns: 2fr 4fr;
30     grid-template-areas:
31       "hd    hd"
32       "sidebar main"
33       "sidebar ft"
34     }
35   }
36 }
37
38 > .grid > * { }
39 > .grid article { }
40
41
42
```

The browser preview on the right shows a layout with four main sections: Header (top), Aside (left), Article (center), and Footer (bottom). The Article section contains the placeholder text "Content would go here".

INTRODUCING MINMAX()

The screenshot shows a code editor window with two panes. The left pane displays the HTML and CSS code for a grid layout, while the right pane shows the resulting visual preview.

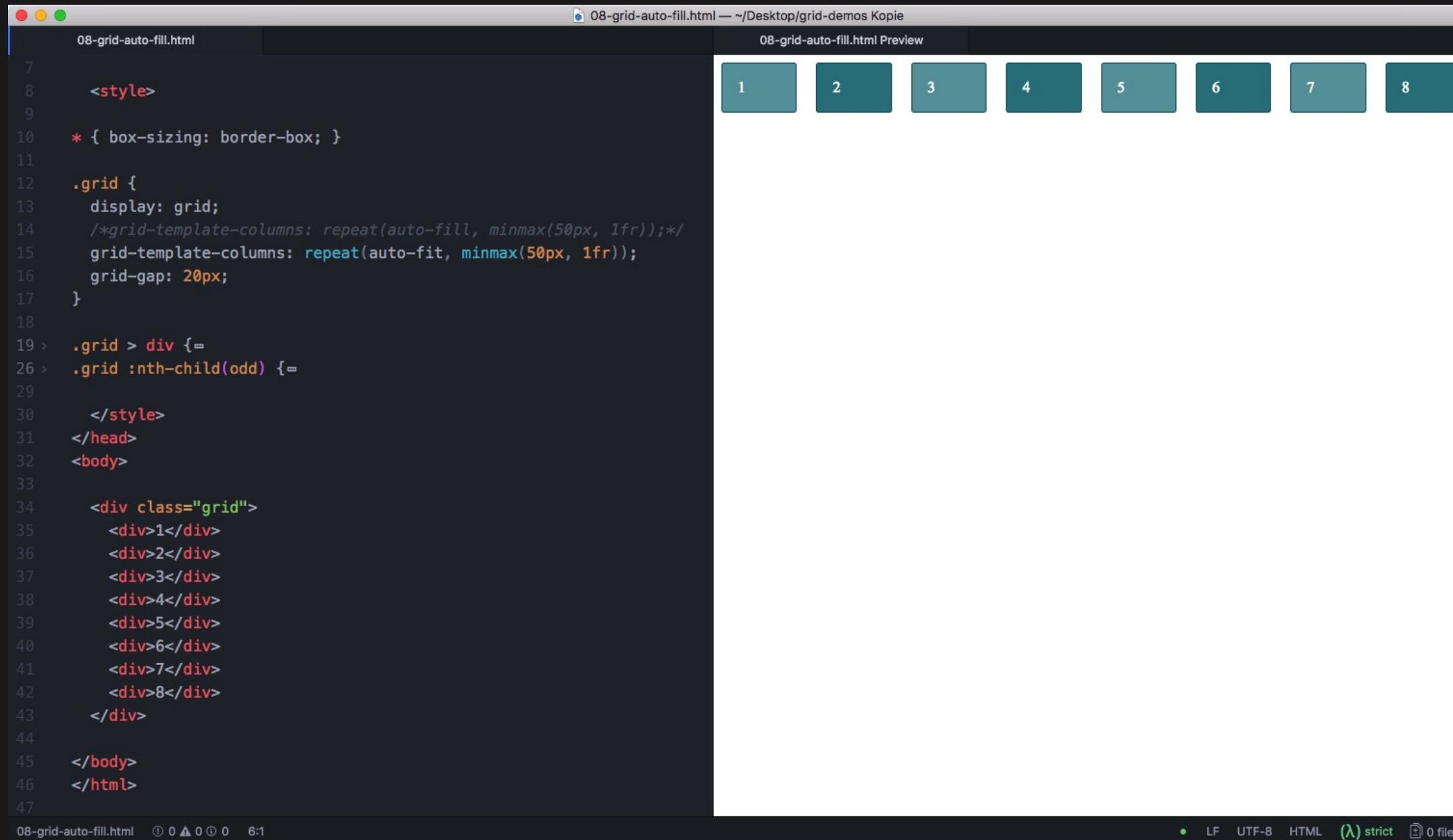
Code Editor (Left):

```
07-grid-minmax.html
1 <title>Grid Layout Demo</title>
2
3 <style>
4 * { box-sizing: border-box; }
5
6 .grid {
7   display: grid;
8   grid-template-columns: 1fr 1fr 1fr;
9   grid-auto-rows: minmax(60px, auto);
10  grid-gap: 20px;
11 }
12
13 .grid > div {
14   &:nth-child(odd) {
15     background-color: #2e3436;
16     color: white;
17     padding: 10px;
18   }
19
20 </head>
21 <body>
22
23   <div class="grid">
24     <div>1
25       <br>Some
26       <br>more
27       <br>Content</div>
28     <div>2</div>
29     <div>3</div>
30     <div>4</div>
31     <div>5</div>
32     <div>6</div>
33     <div>7</div>
34     <div>8</div>
35   </div>
36
37 0 0 0 0 46:17
```

Preview (Right):

The preview shows a 3x3 grid of teal-colored boxes with white text. The first row contains three boxes labeled 1, 2, and 3. The second row contains three boxes labeled 4, 5, and 6. The third row contains two boxes labeled 7 and 8. Box 1 contains the text "Some more Content". The grid has a gap of 20px between items and a total height of 180px (3 rows * 60px). The last row is shorter because it only contains two items.

INTRODUCING AUTO-FILL AND AUTO-FIT



The screenshot shows a code editor with two panes. The left pane displays the source code for '08-grid-auto-fill.html', and the right pane shows the visual output of the code.

Code (08-grid-auto-fill.html):

```
08-grid-auto-fill.html
08-grid-auto-fill.html — ~/Desktop/grid-demos Kopie
08-grid-auto-fill.html Preview

1 <style>
2   * { box-sizing: border-box; }
3
4   .grid {
5     display: grid;
6     /*grid-template-columns: repeat(auto-fit, minmax(50px, 1fr));*/
7     grid-template-columns: repeat(auto-fit, minmax(50px, 1fr));
8     grid-gap: 20px;
9   }
10
11   .grid > div {
12     &:nth-child(odd) {
13       background-color: #2e3436;
14       color: white;
15     }
16   }
17
18 </style>
19 </head>
20 <body>
21
22   <div class="grid">
23     <div>1</div>
24     <div>2</div>
25     <div>3</div>
26     <div>4</div>
27     <div>5</div>
28     <div>6</div>
29     <div>7</div>
30     <div>8</div>
31   </div>
32
33 </body>
34 </html>
```

Preview: The preview shows a horizontal grid of 8 teal-colored boxes, each containing a number from 1 to 8. The boxes are separated by a 20px gap.

ALIGNING AND JUSTIFYING GRID ITEMS

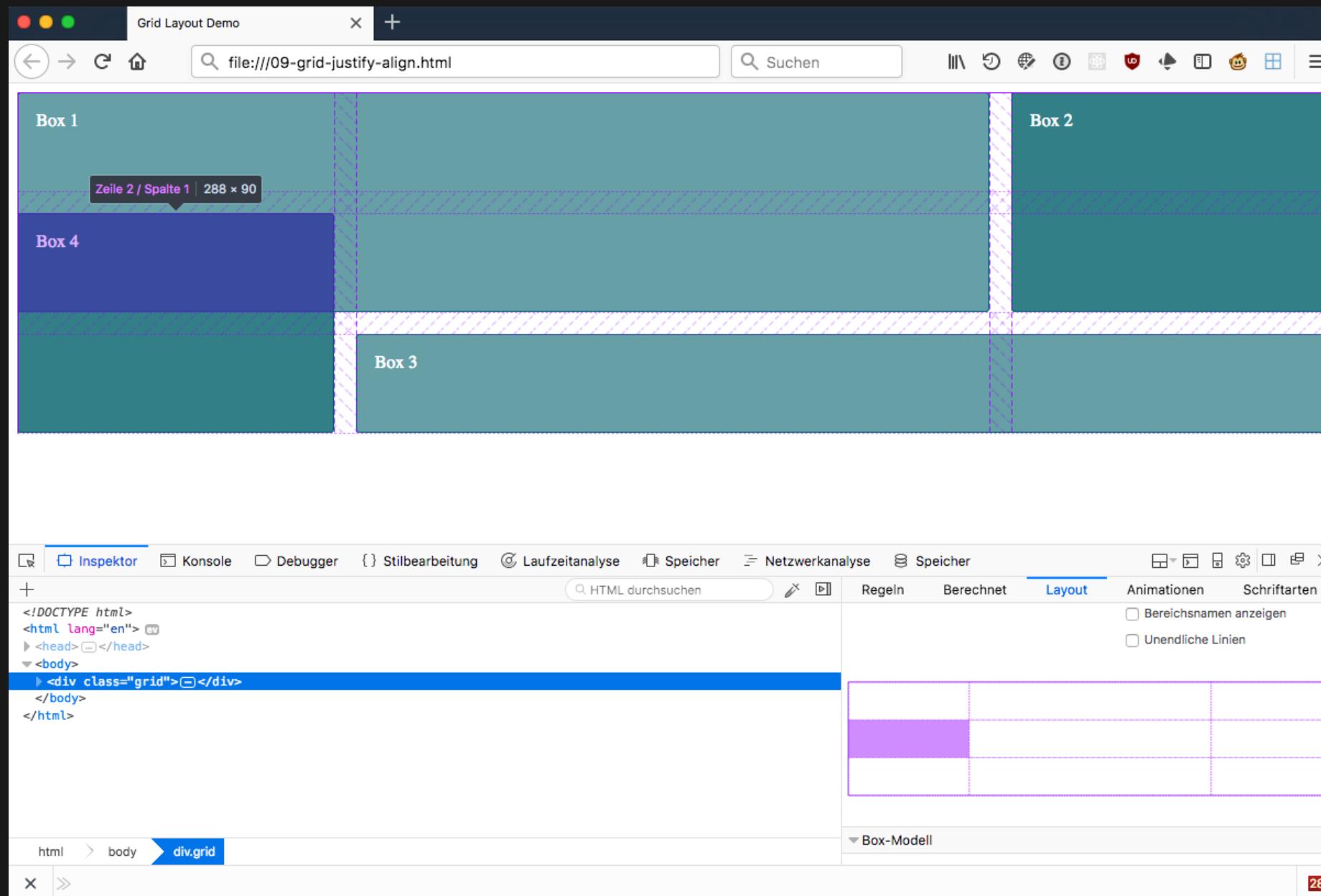
The screenshot shows a code editor and a browser preview side-by-side. The code editor on the left displays the following CSS code:

```
09-grid-justify-align.html
09-grid-justify-align.html — ~/Desktop/grid-demos Kopie
09-grid-justify-align.html Preview

11 * { box-sizing: border-box; }
12
13 .grid {
14   display: grid;
15   grid-template-columns: 1fr 2fr 1fr;
16   grid-auto-rows: minmax(90px, auto);
17   grid-gap: 20px;
18   justify-items: stretch;
19   align-items: stretch;
20 }
21
22 .one {
23   grid-column: 1 / 3;
24   grid-row: 1 / 3;
25   /*align-self: start;*/
26 }
27 .two {
28   grid-column: 3;
29   grid-row: 1 / 3;
30 }
31 .three {
32   grid-column: 2 / 4;
33   grid-row: 3;
34 }
35 .four {
36   grid-column: 1;
37   grid-row: 2 / 4;
38 }
39
40
41 > .grid > div {-
42 >   .grid :nth-child(odd) {-
43 >     background-color: #1a237e;
44 >     color: white;
45 >     padding: 10px;
46 >   }
47 > }
```

The browser preview on the right shows a grid layout with four items: Box 1, Box 2, Box 3, and Box 4. The grid has three columns with widths of 1fr, 2fr, and 1fr respectively. Box 1 is in the top-left cell, Box 2 is in the top-right cell, Box 3 is in the bottom-right cell, and Box 4 is in the bottom-left cell. All items have a height of 90px and a 20px gap between them. The items are aligned to stretch both horizontally and vertically.

FIREFOX GRID INSPECTOR



OVERVIEW

- Fonts and Graphical Effects
- Device Adaptation: Responsive Webdesign
- Layouts: CSS Flexible Box Layout Module
- Layouts: CSS Grid Layout
- Final Remarks

CONSIDER FLEXBOX

Flexbox or Grid?

Use Flexbox when ...

- ▶ Your content is a row OR a column
- ▶ You want the size of items to dictate their layout
- ▶ You want to distribute space

CONSIDER GRID

Flexbox or Grid?

Consider grid when ...

- ▶ You need to control rows and columns
- ▶ You are adding widths to a flex item in order to make it line up with rows above.
- ▶ You want control of the layout from the parent
- ▶ Items need to occupy the same space or overlap

GRID AND OLD BROWSERS

What about old browsers?

If using display: grid on a container, child items:

- ▶ Using float, lose their float behaviour
- ▶ The vertical-align property has no effect
- ▶ Flex items become grid items
- ▶ Items set to display: block or inline-block become grid items
- ▶ Items set to display: table-cell stop creating anonymous boxes

GRID FEATURE QUERIES

Using feature queries

Add a margin for flex layout, remove it if we are using grid layout.

```
.listing > * {  
  flex: 1 1 30%;  
  margin: 0 20px 20px 20px;  
}  
  
@supports(display: grid) {  
  .listing > * {  
    margin: 0;  
  }  
}
```

READING MATERIAL, SOURCES

READING MATERIAL

- A Complete Guide to Flexbox, CSS-Tricks
css-tricks.com/snippets/css/a-guide-to-flexbox/
- A Complete Guide to Grid, CSS-Tricks
css-tricks.com/snippets/css/complete-guide-grid/

SOURCES: FLEXBOX

- What The Flexbox?! Video Tutorial by Wes Bos
flexbox.io/
- Mozilla Developer Network: Using CSS flexible boxes
developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes
- Mozilla Developer Network: CSS Length Units
developer.mozilla.org/en-US/docs/Web/CSS/length
- Flexbox Froggy – A Game For Learning Flexbox
flexboxfroggy.com

SOURCES: GRID

- Get Started with Grid Layout (Videos)
gridbyexample.com/learn/
- SmashingConf San Francisco 2017 - Rachel Andrew on
Laying Out The Future With Grid And Flexbox
vimeo.com/215091807
- Grid Garden – A Game For Learning Grid
cssgridgarden.com

