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

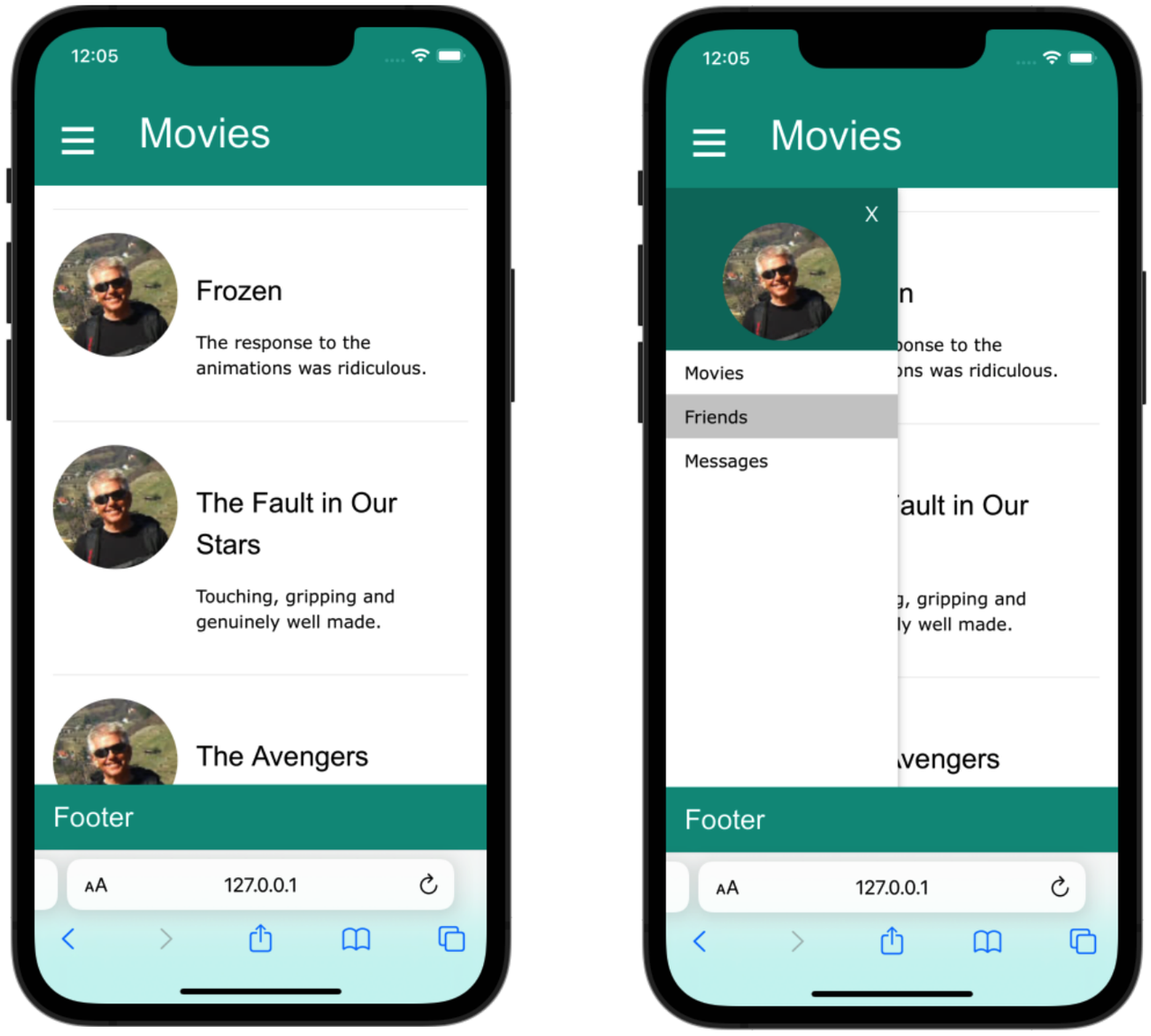
- If necessary consult:
  - WEB1 slides and related material
  - WBE preparation course
    https://moodle.zhaw.ch/course/view.php?id=1007
  - Learn to Code HTML & CSS
    https://learn.shayhowe.com/html-css/
- You probably also know:
  - Front-end frameworks like Foundation or Bootstrap
  - CSS preprocessors 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 MacOS X
- BlinkMacSystemFont is the equivalent for Chrome on MacOS 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

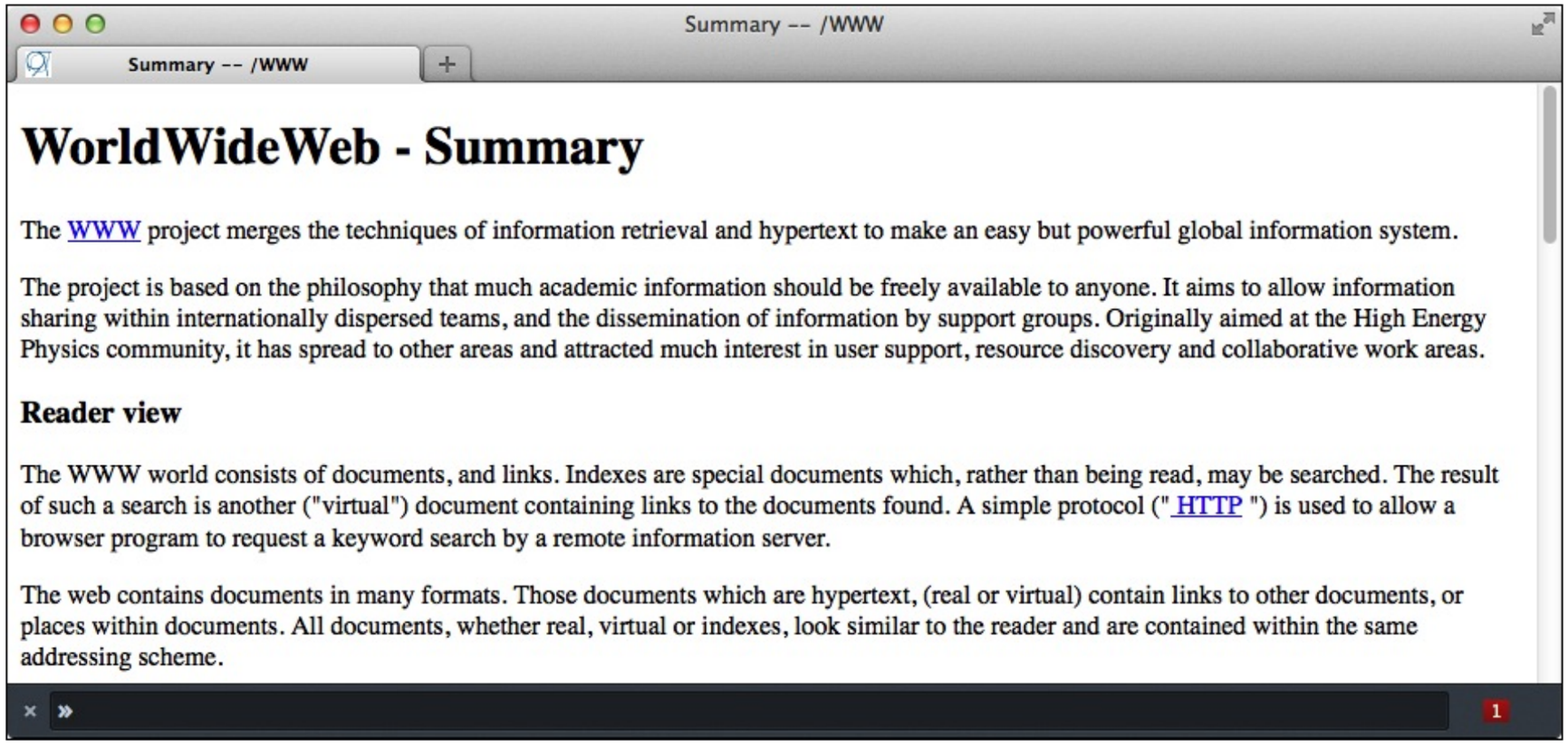
- Viewport-relative CSS units `vw`, `vh`, `vmin` and `vmax`
- CSS Media Queries, responsive design
- Responsive Images: `picture`, `srcset` (or: SVG)
- JS-based: Web Components

Technologies in progress:

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

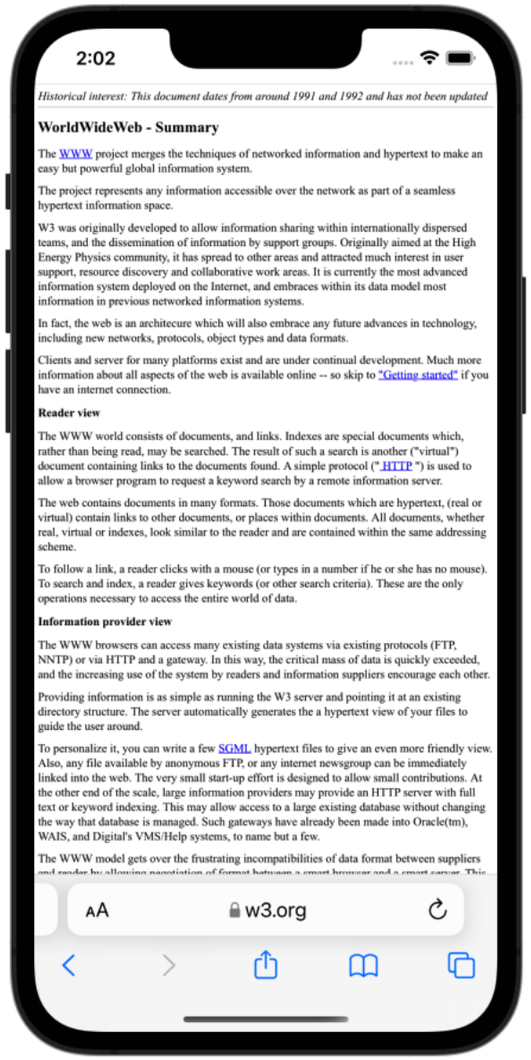
https://w3c.github.io/web-roadmaps/mobile/adaptation.html

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…

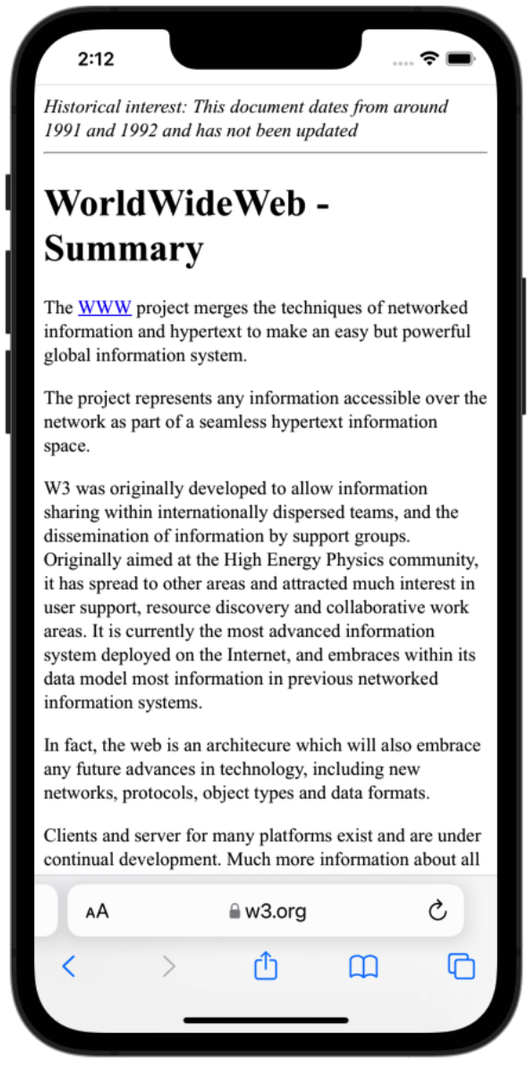
```
<meta name="viewport" content="width=320">
<meta name="viewport" content="width=device-width">
```

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?

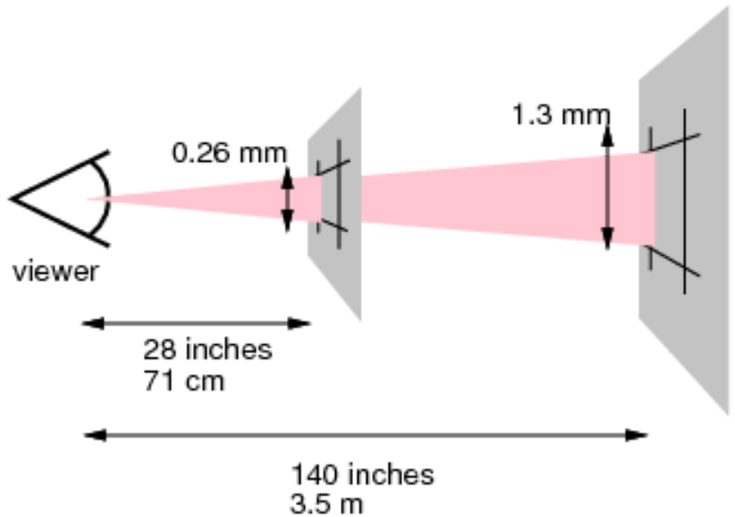
- According to the CSS 2.1 specification px is an absolute unit:

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

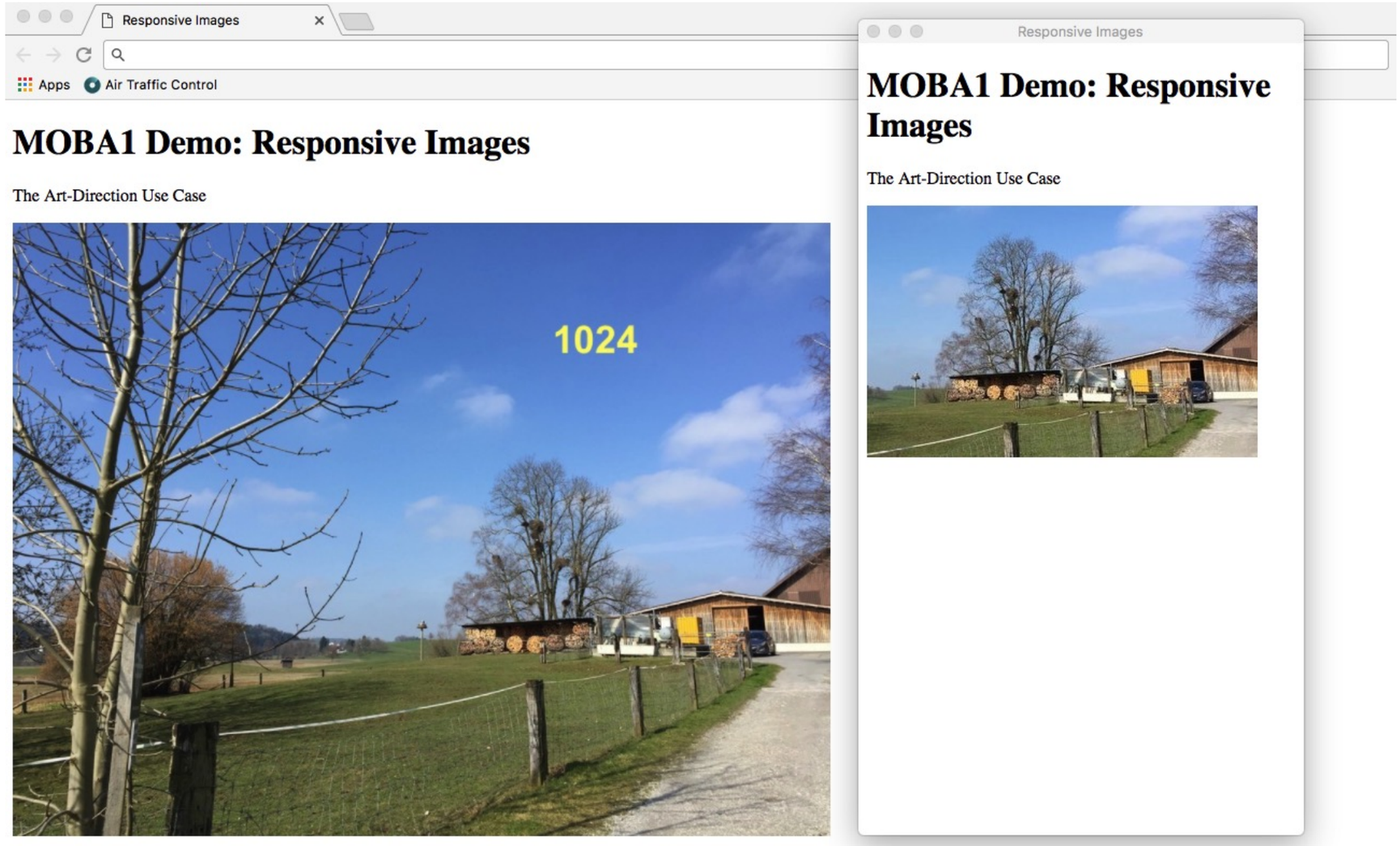
IMAGE RESOLUTION

- 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" />
  <img src="small.jpg" alt="Landscape with farm house" />
</picture>
```

RESPONSIVE IMAGES



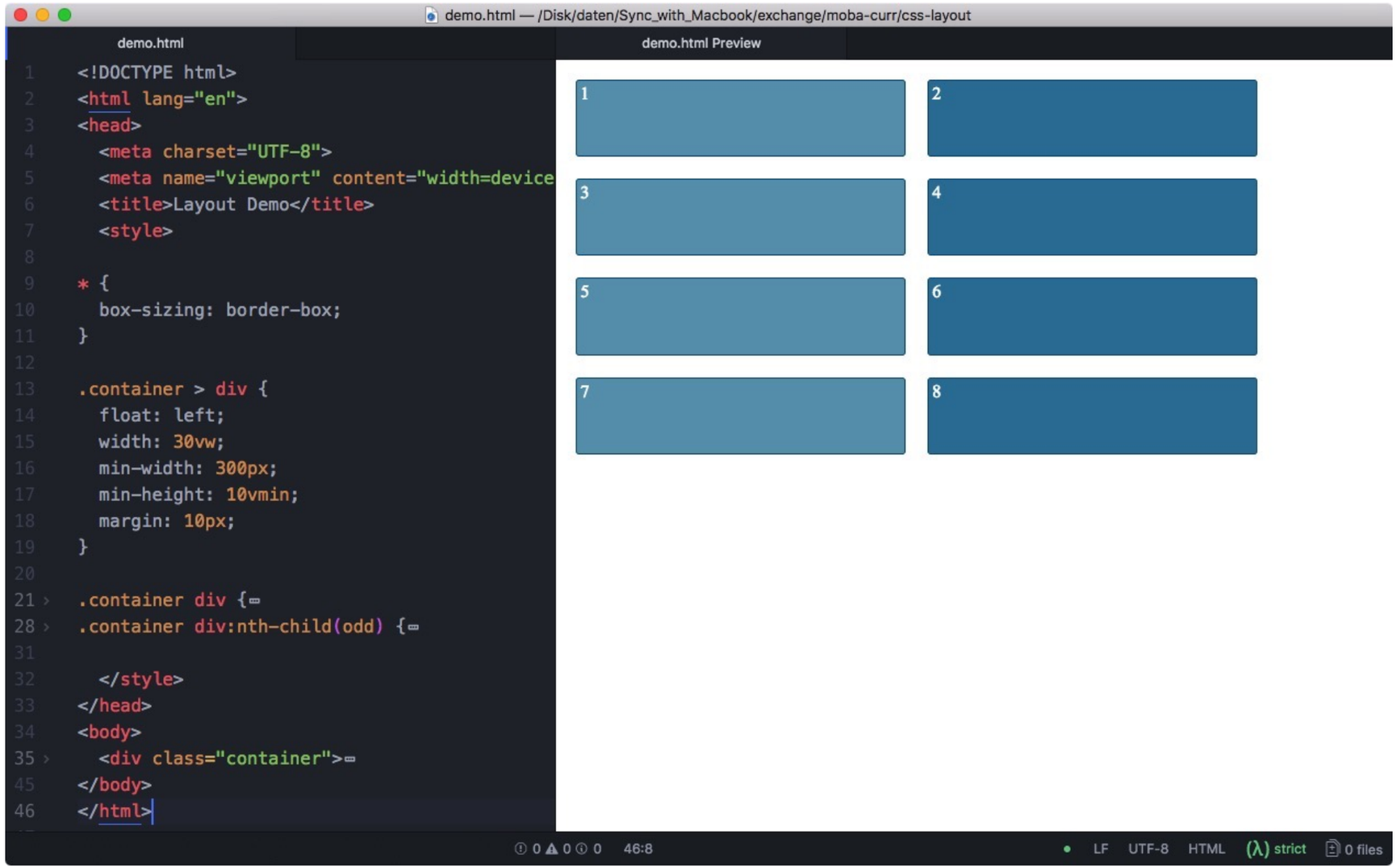
OVERVIEW

- 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



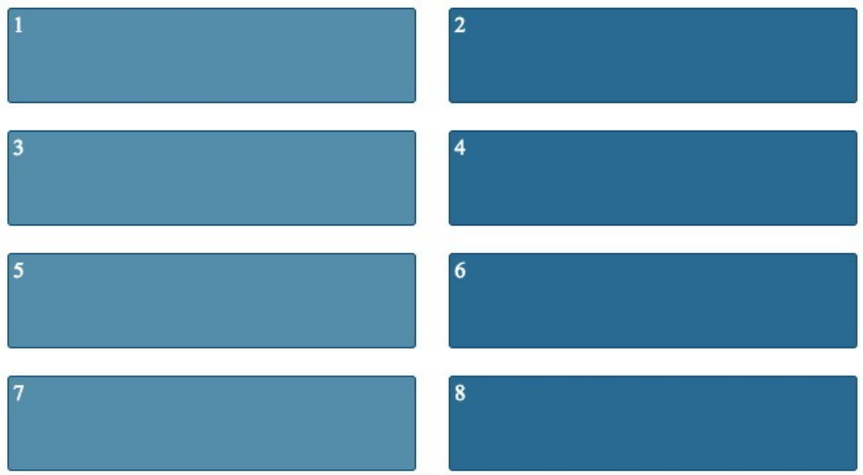
CSS LAYOUT: INLINE-BLOCK

demo.html — /Disk/daten/Sync_with_Macbook/exchange/moba-curr/css-layout

demo.html

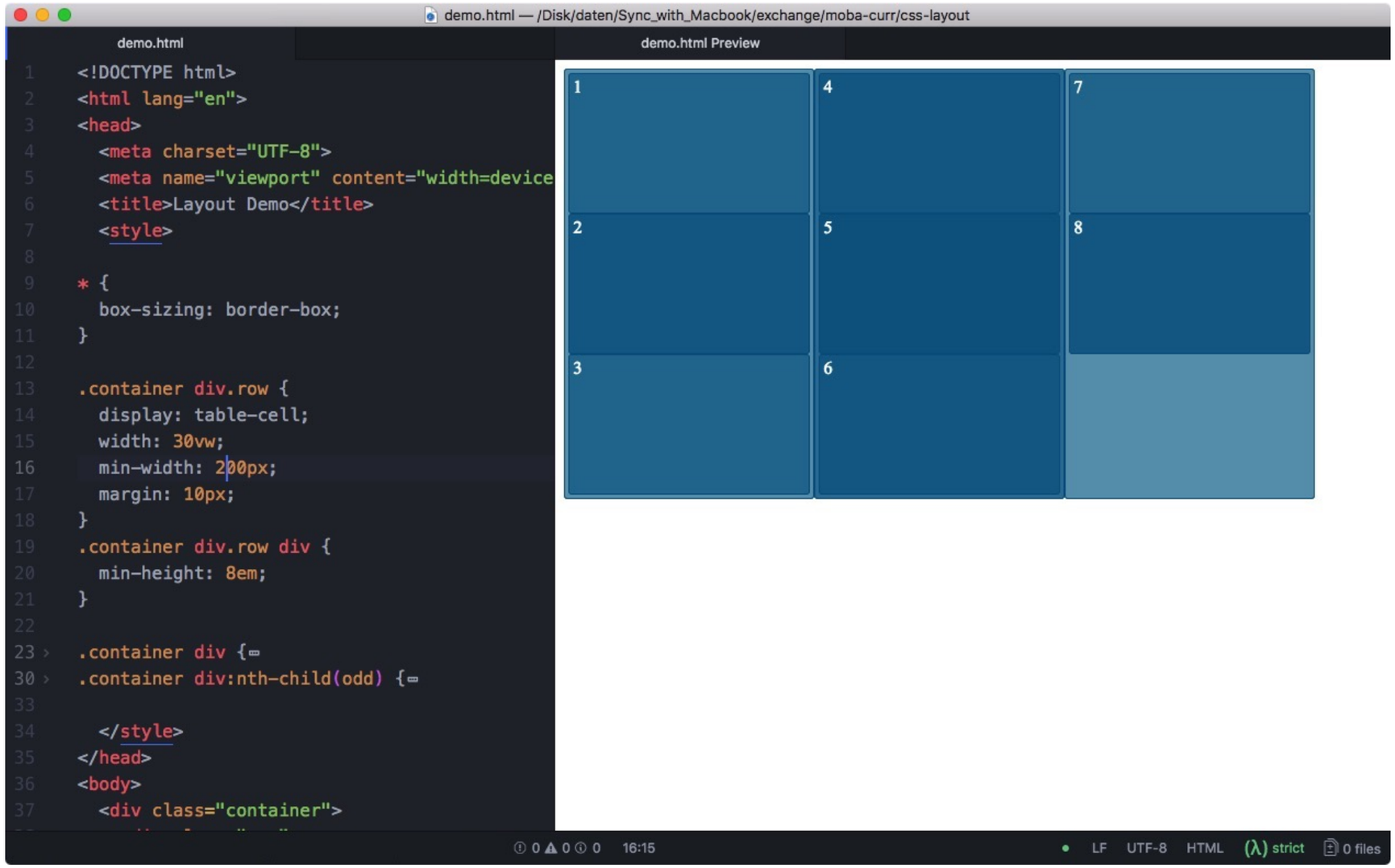
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device
  <title>Layout Demo</title>
  <style>

* {
  box-sizing: border-box;
}

.container > div {
  display: inline-block;
  width: 30vw;
  min-width: 300px;
  min-height: 10vmin;
  margin: 10px;
}

.container div {…
.container div:nth-child(odd) {…

  </style>
</head>
<body>
  <div class="container">…
</body>
</html>
```

demo.html Preview



0 0 0 46:8 LF UTF-8 HTML (λ) strict 0 files

CSS LAYOUT: TABLE-CELL



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

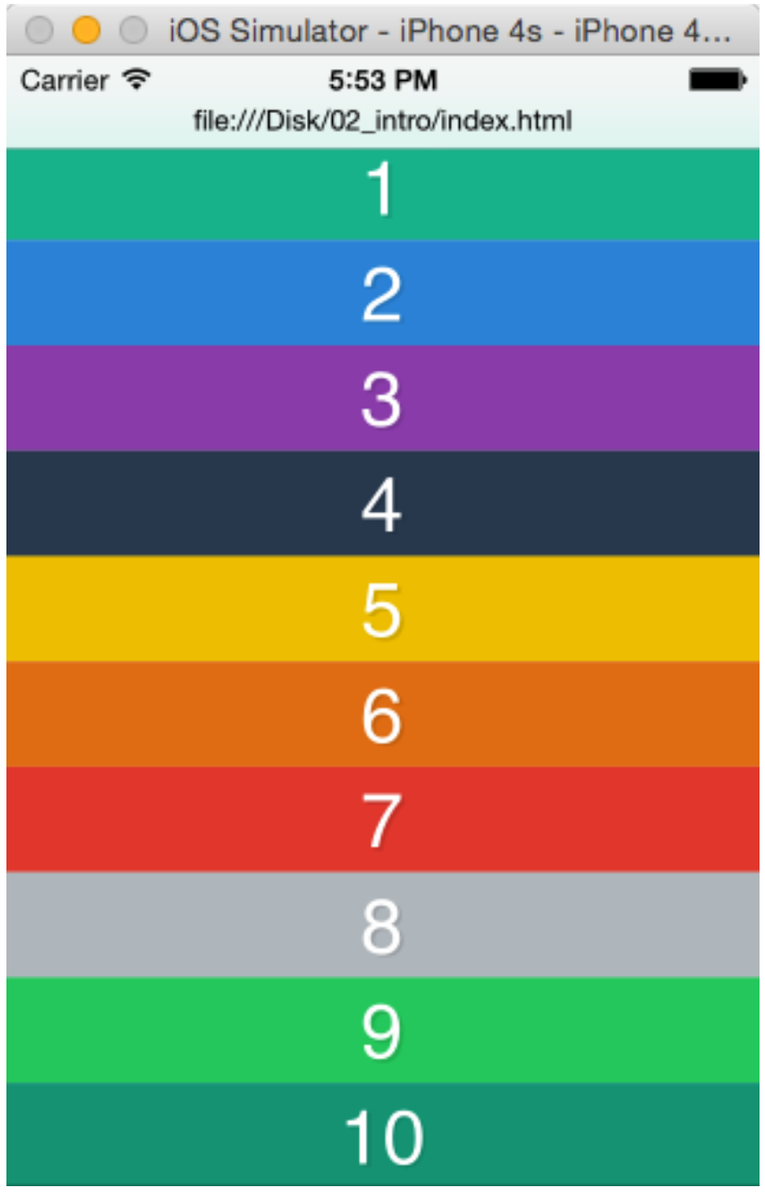
EXAMPLE

```
<!-- 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`

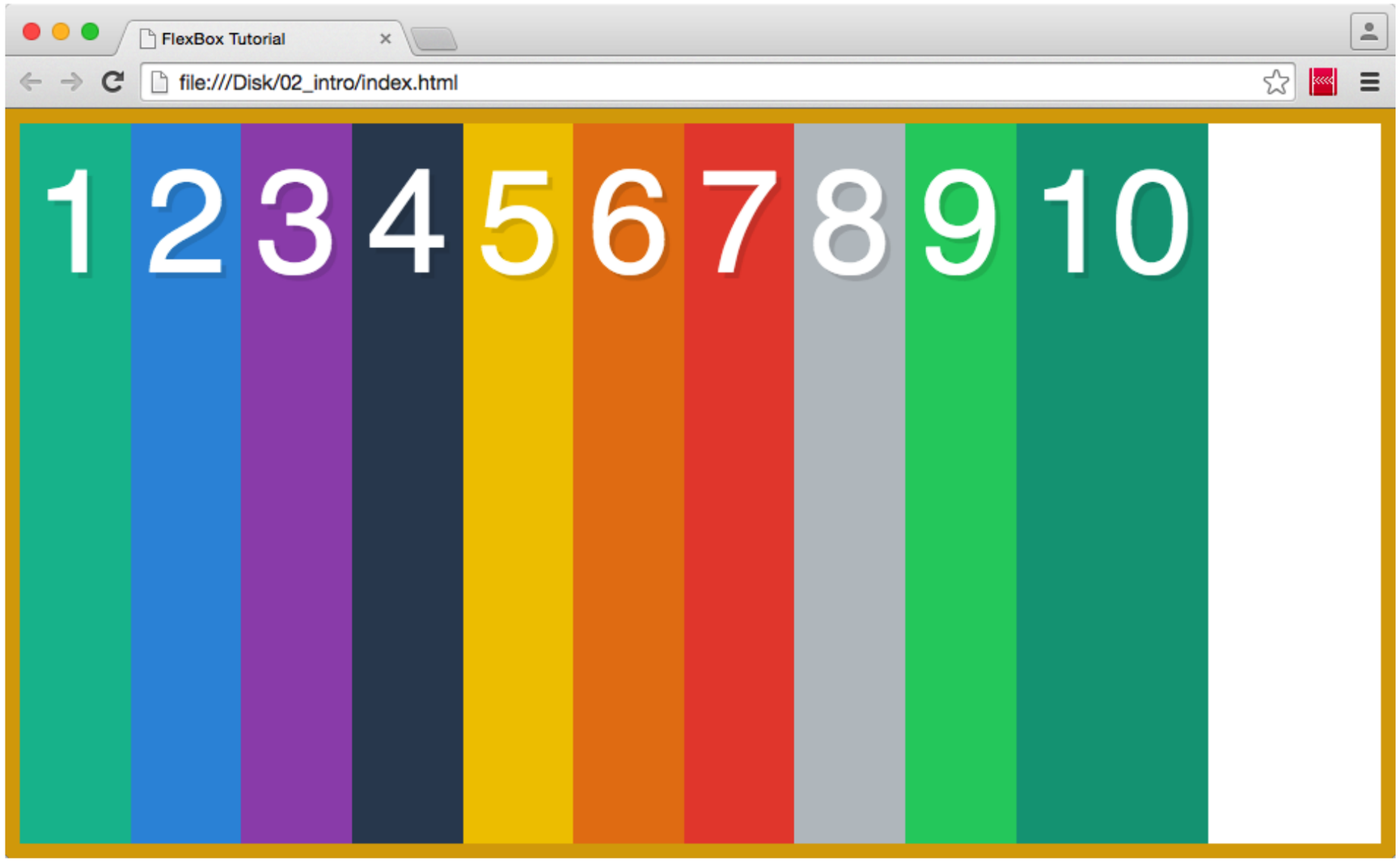
EXAMPLE: FLEX CONTAINER

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

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

↓ preview ↓

EXAMPLE: FLEX CONTAINER



FLEX ITEMS

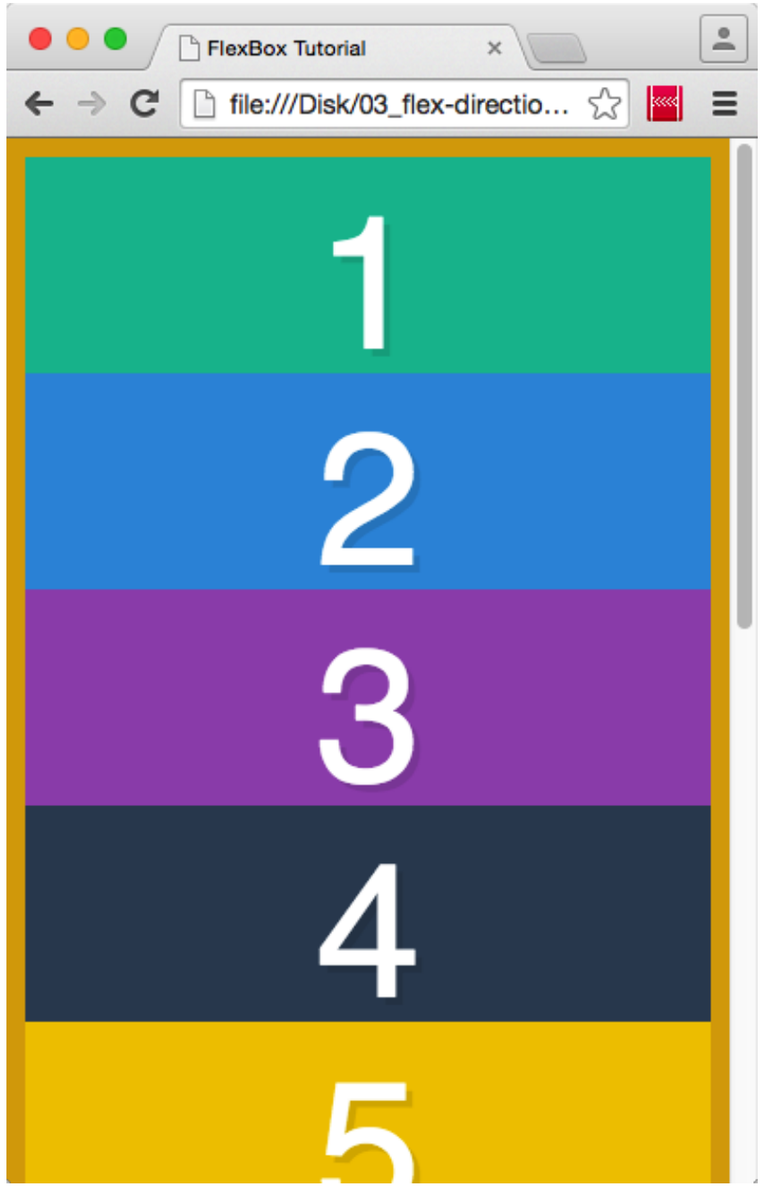
- Every child of a *flex container* is a *flex item*
- There can be any number of flex items
- Flexbox defines how flex items are laid out inside of flex containers

FLEX DIRECTION

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

↓ preview ↓

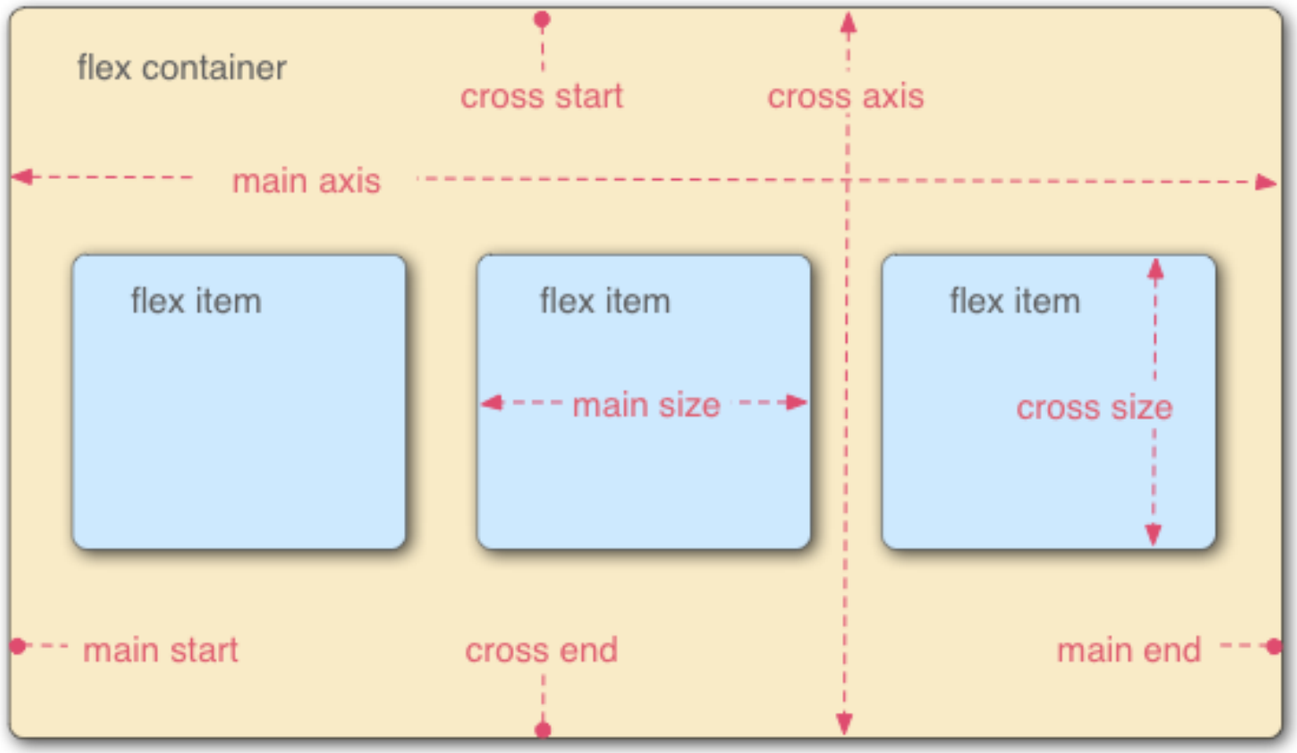
FLEX DIRECTION



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

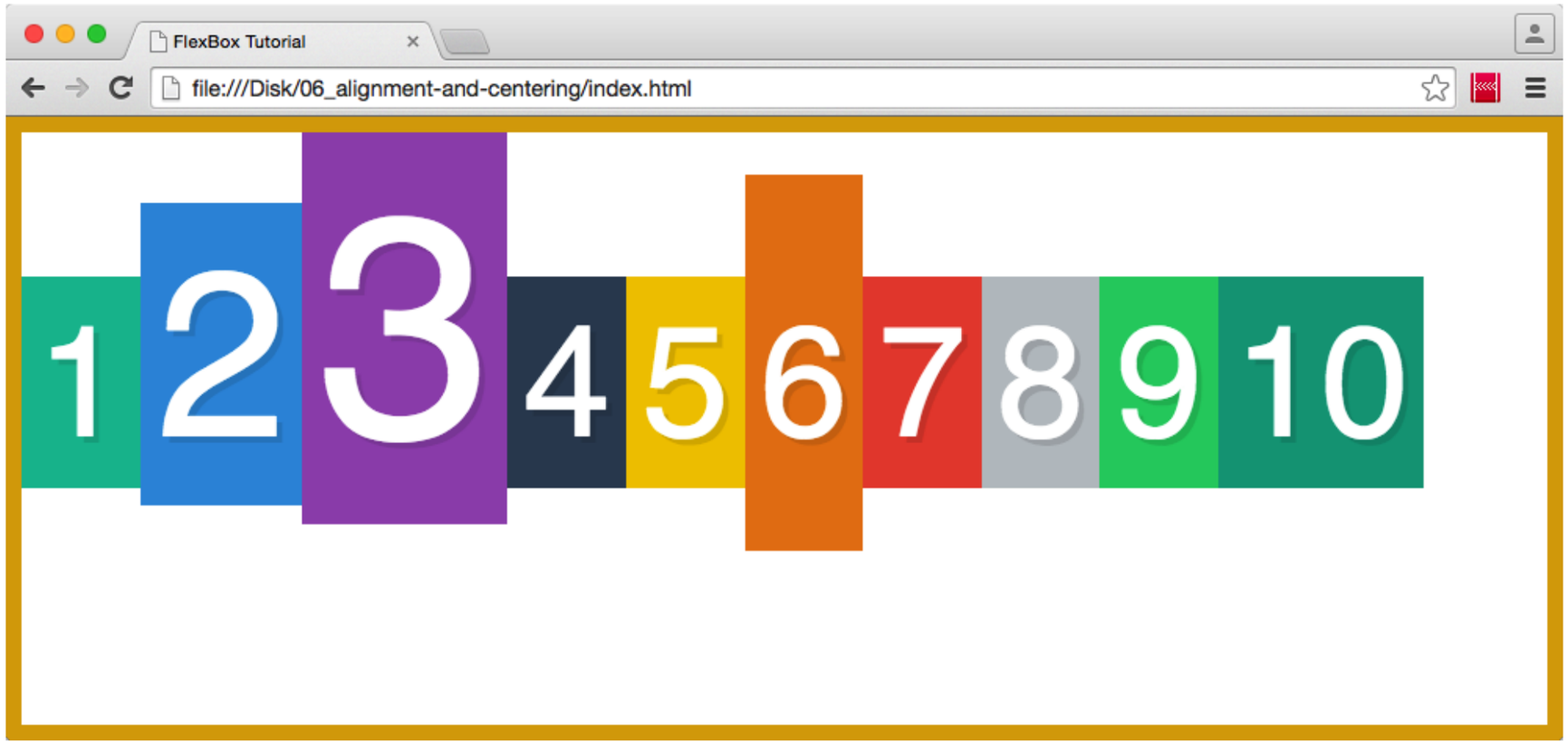
- Another property : `align-items`
- This one defines the alignment along the *cross axis*
- Possible values: `stretch` (Default)
  `flex-start` | `flex-end` | `center` | `baseline`

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 ↓

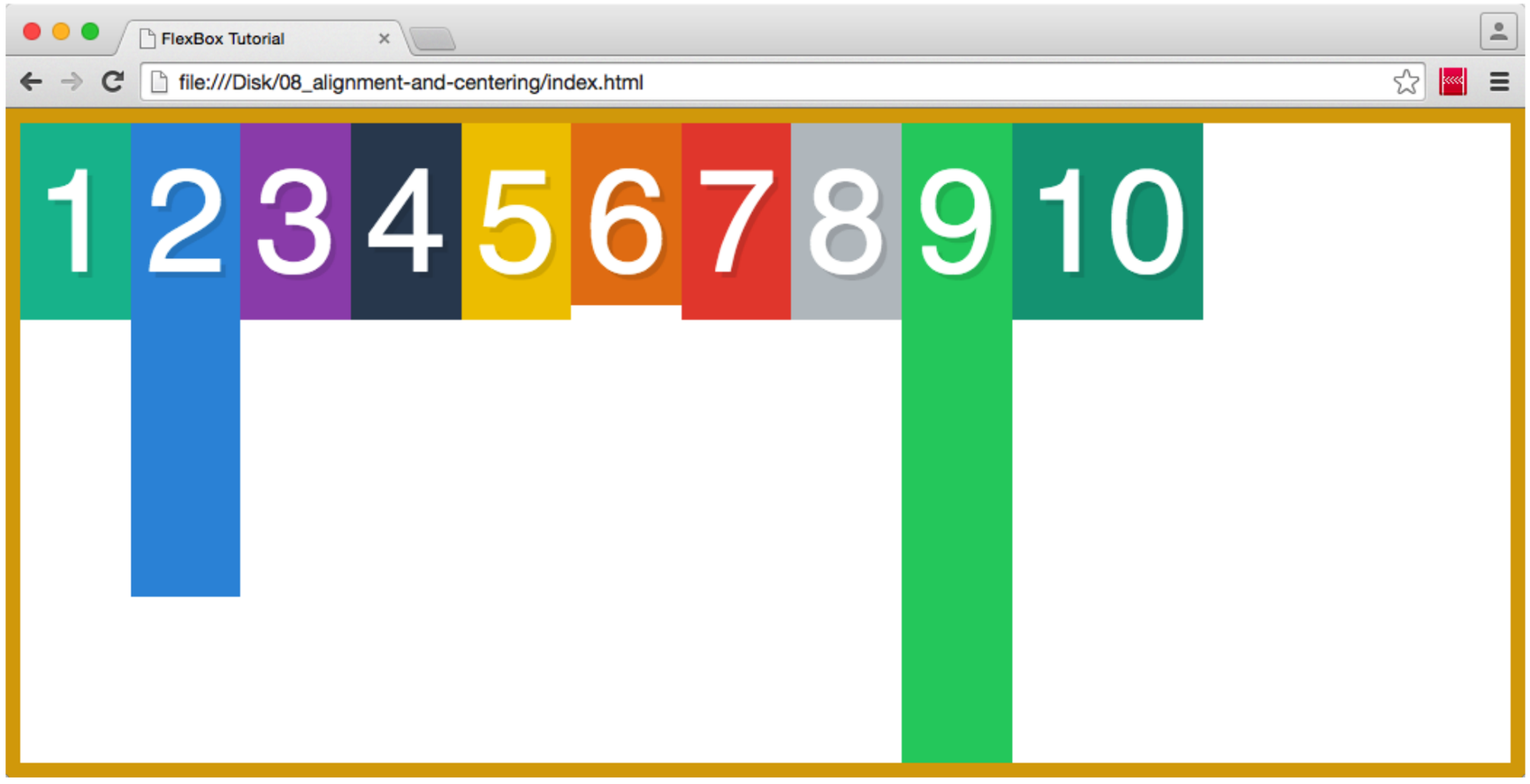
FLEXBOX ALIGNMENT

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



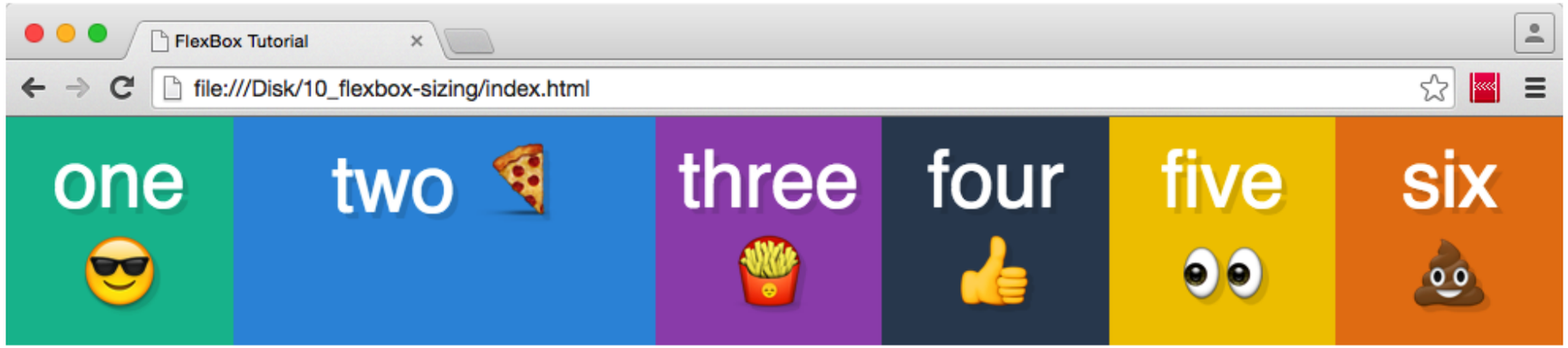
FLEXBOX SIZING

```
<!-- 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*

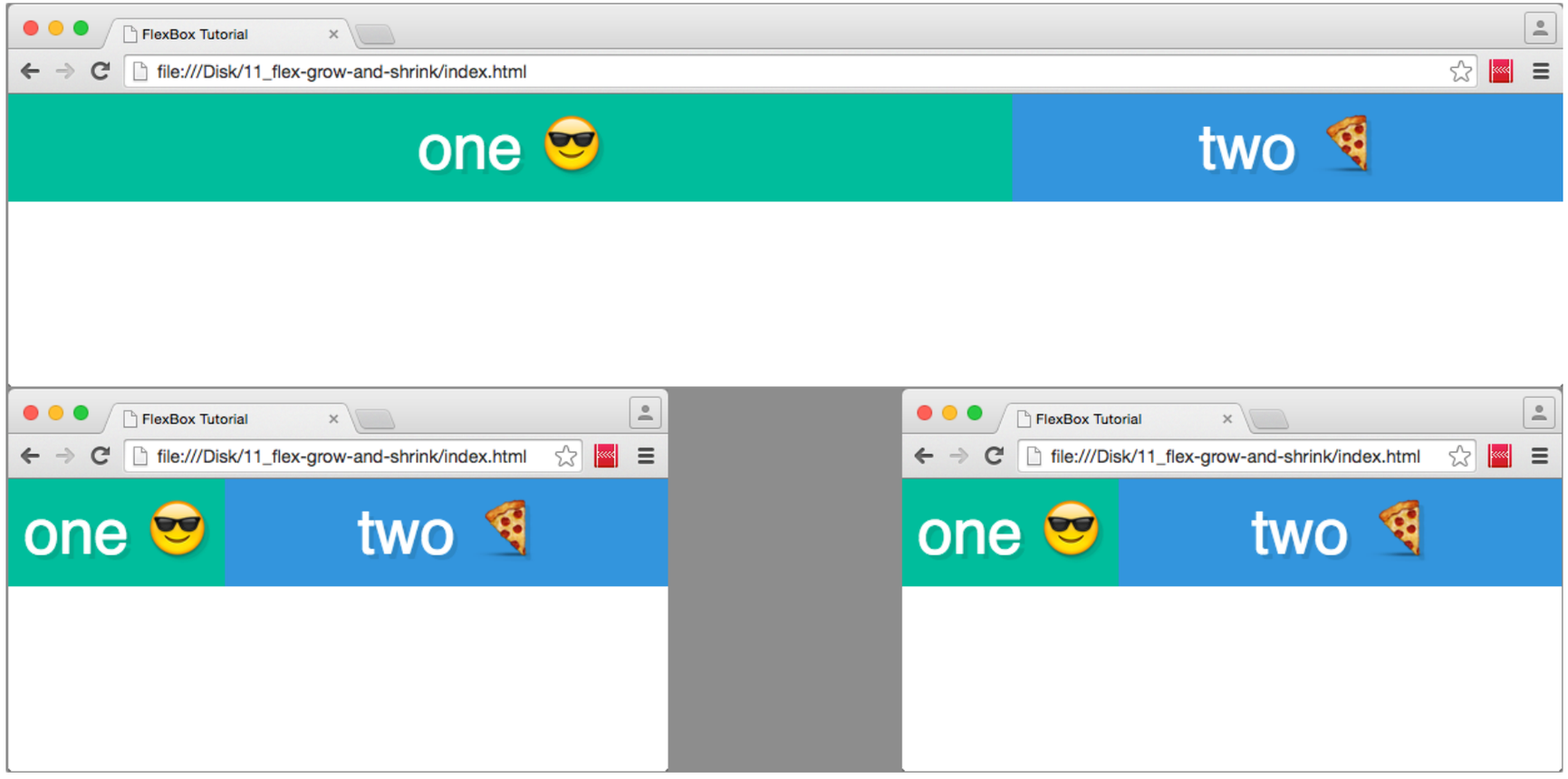
PROPERTY FLEX

```
.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

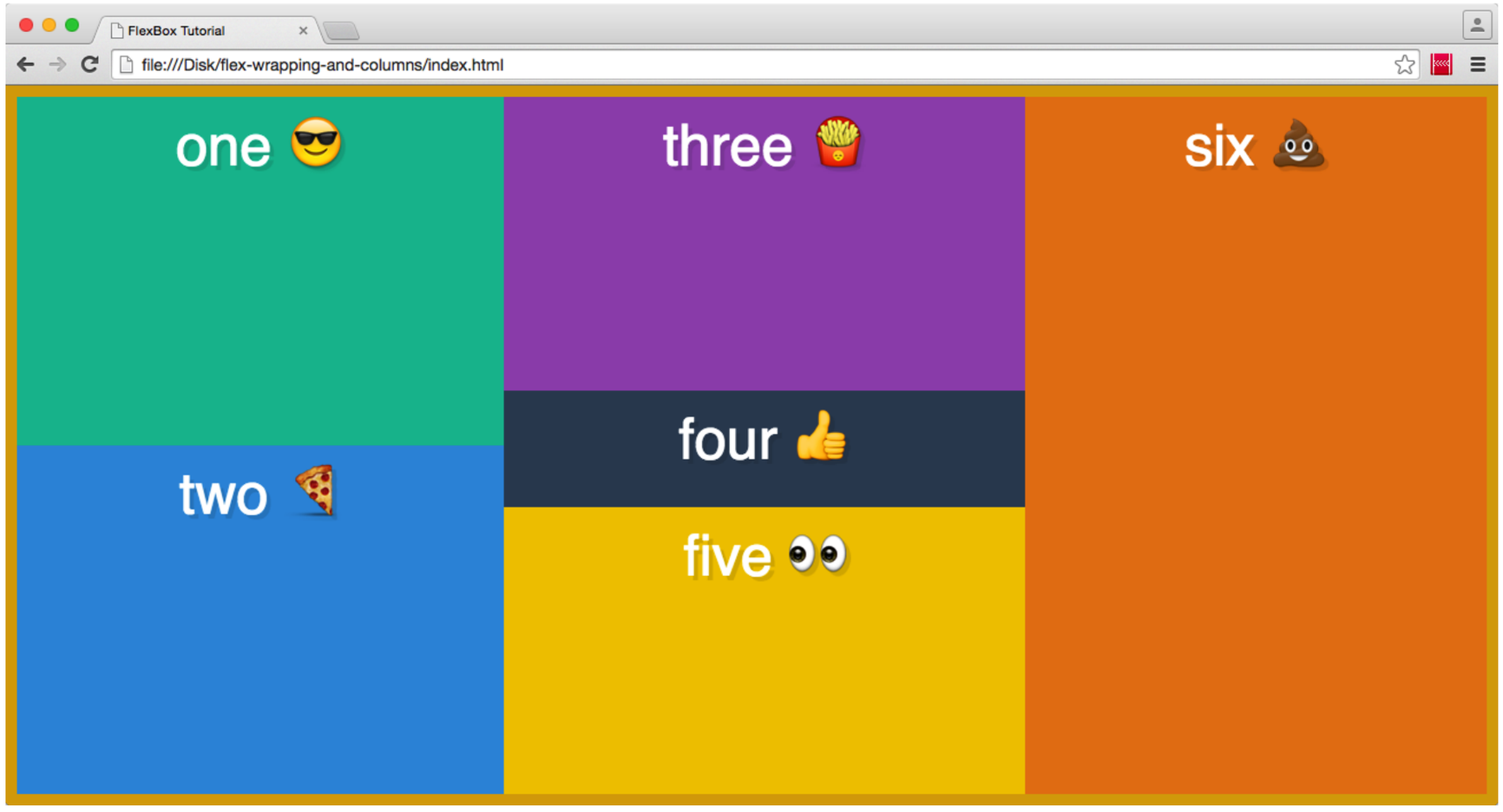
```
<div class="container">
    <div class="box box1">one 😎</div>
    ...
    <div class="box box6">six 💩</div>
</div>
```

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



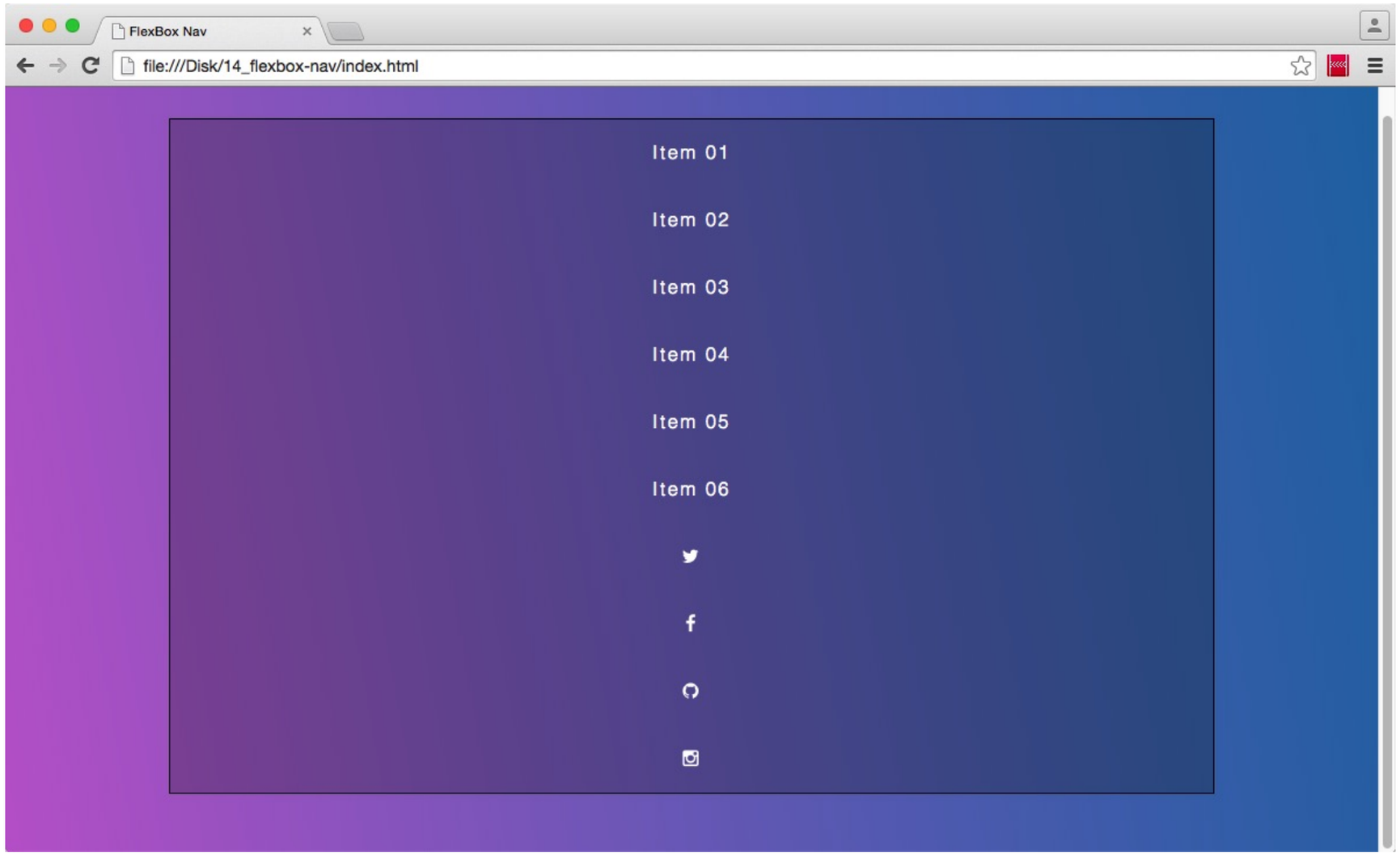
EXAMPLE: NAVIGATION

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

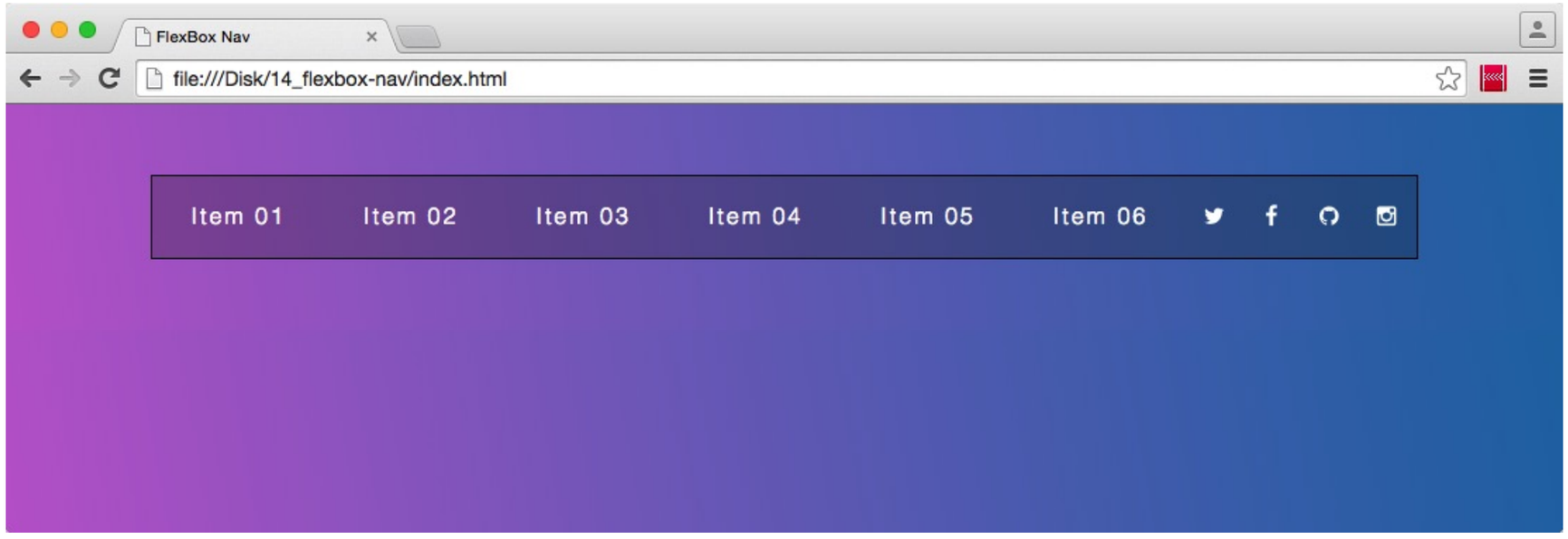


EXAMPLE: NAVIGATION

```
.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;
}
```

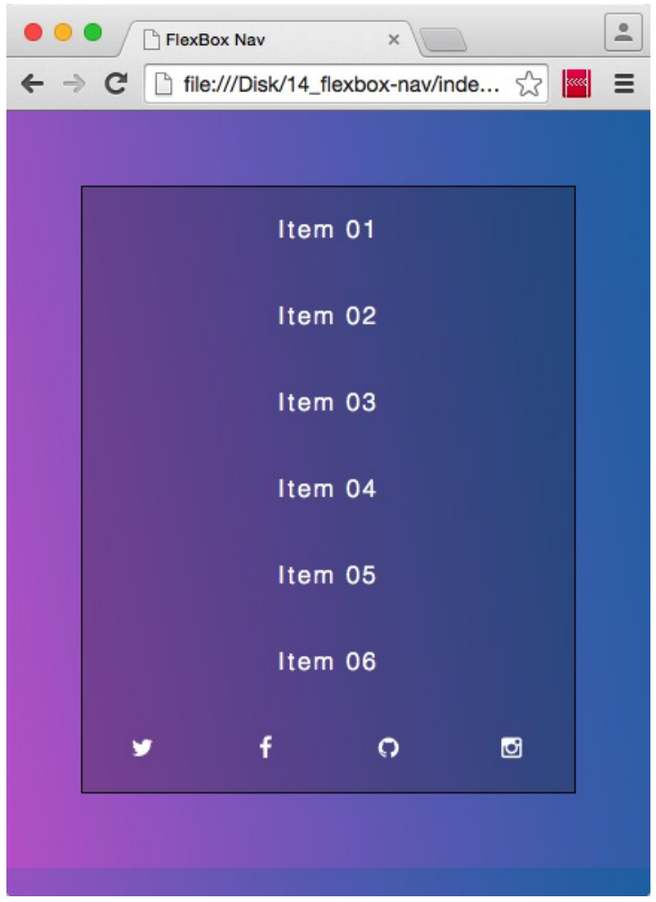
↓ preview ↓

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

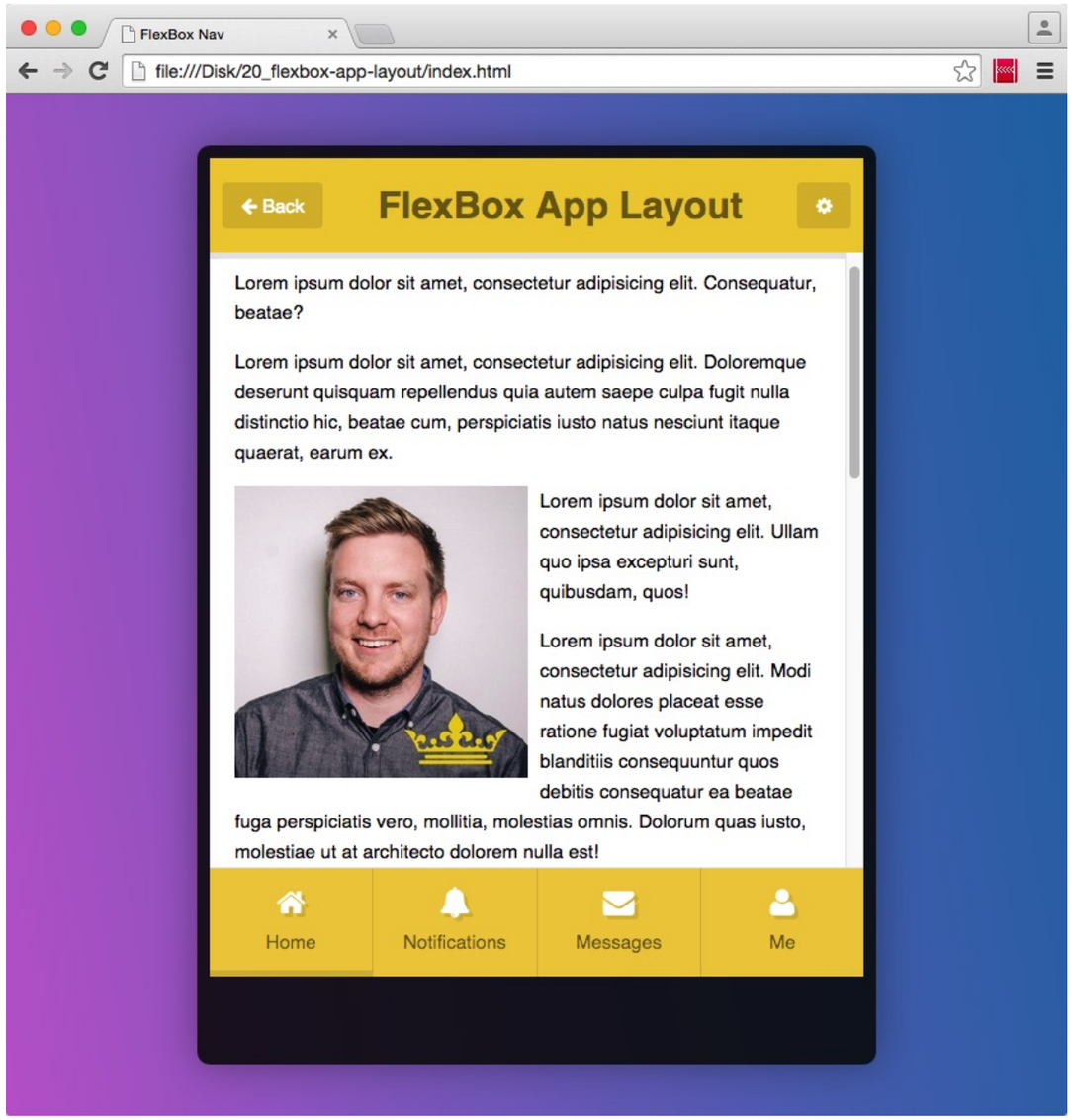
- On small mobile devices
    - the navigation should be on top of the screen
    - it could also be replaced by a toggle button
    - the details and signup areas should be moved up
- All this is easy with flexbox
- For re-ordering, the outer div must be *flex*

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;
}
```

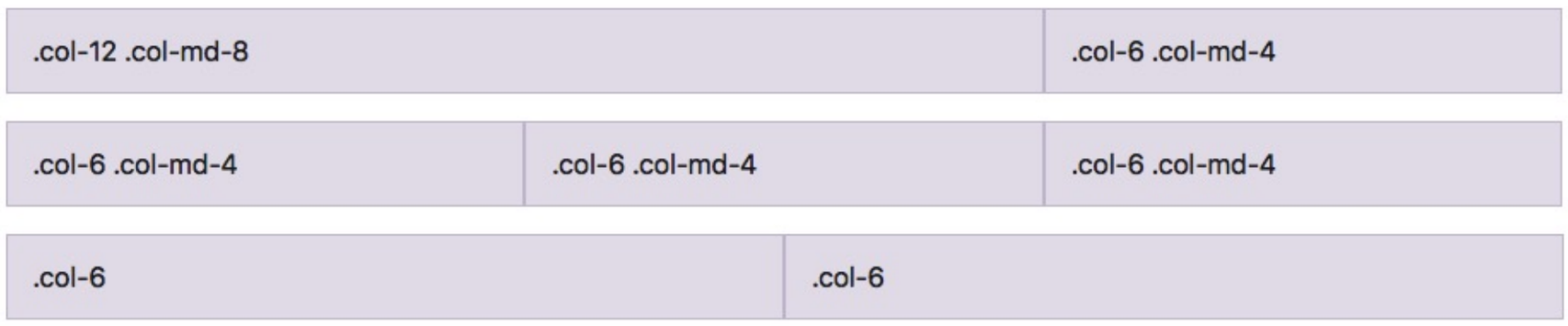


BOOTSTRAP

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

BOOTSTRAP GRID

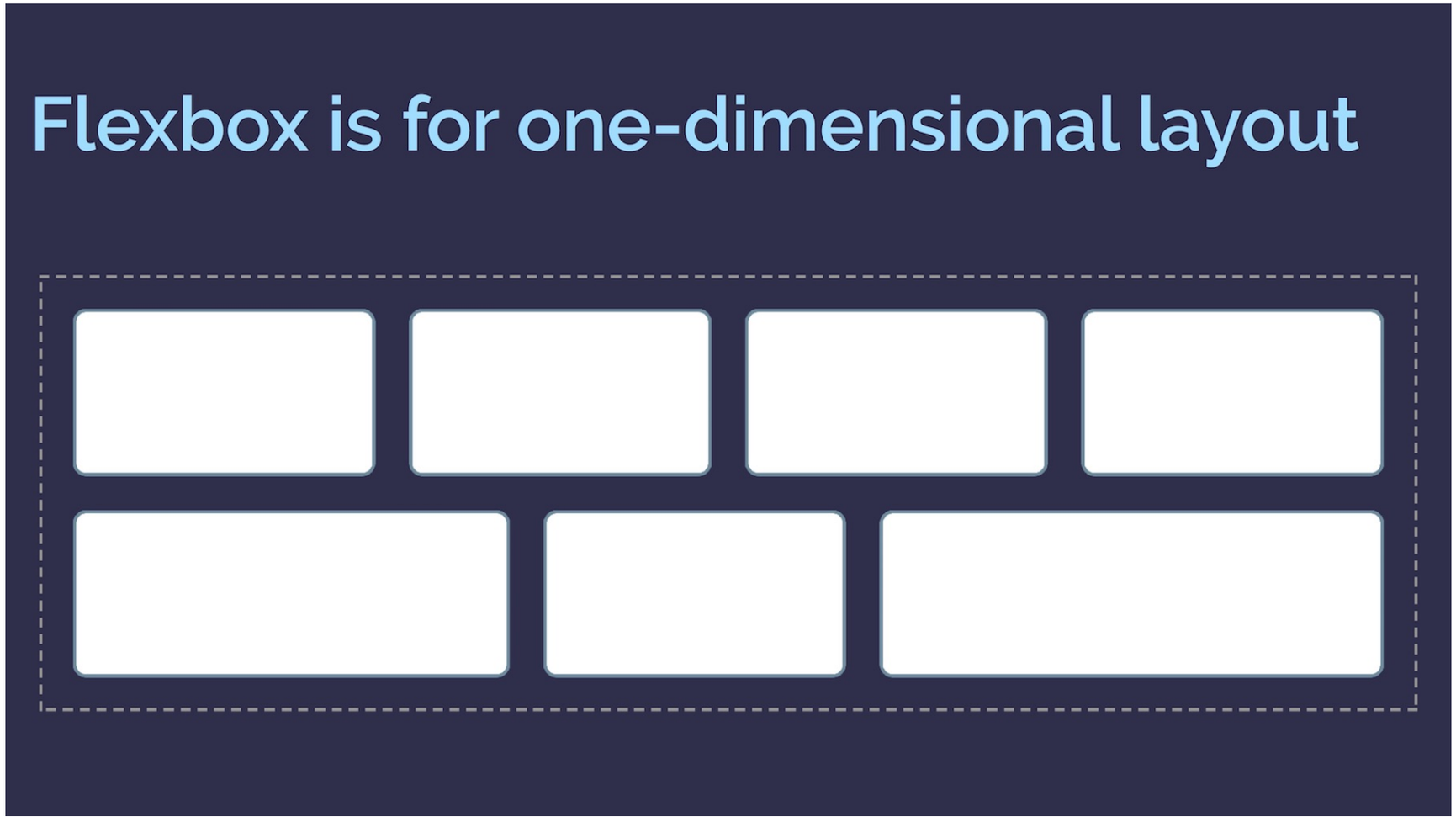
```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-8">.col-12 .col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>
</div>
```



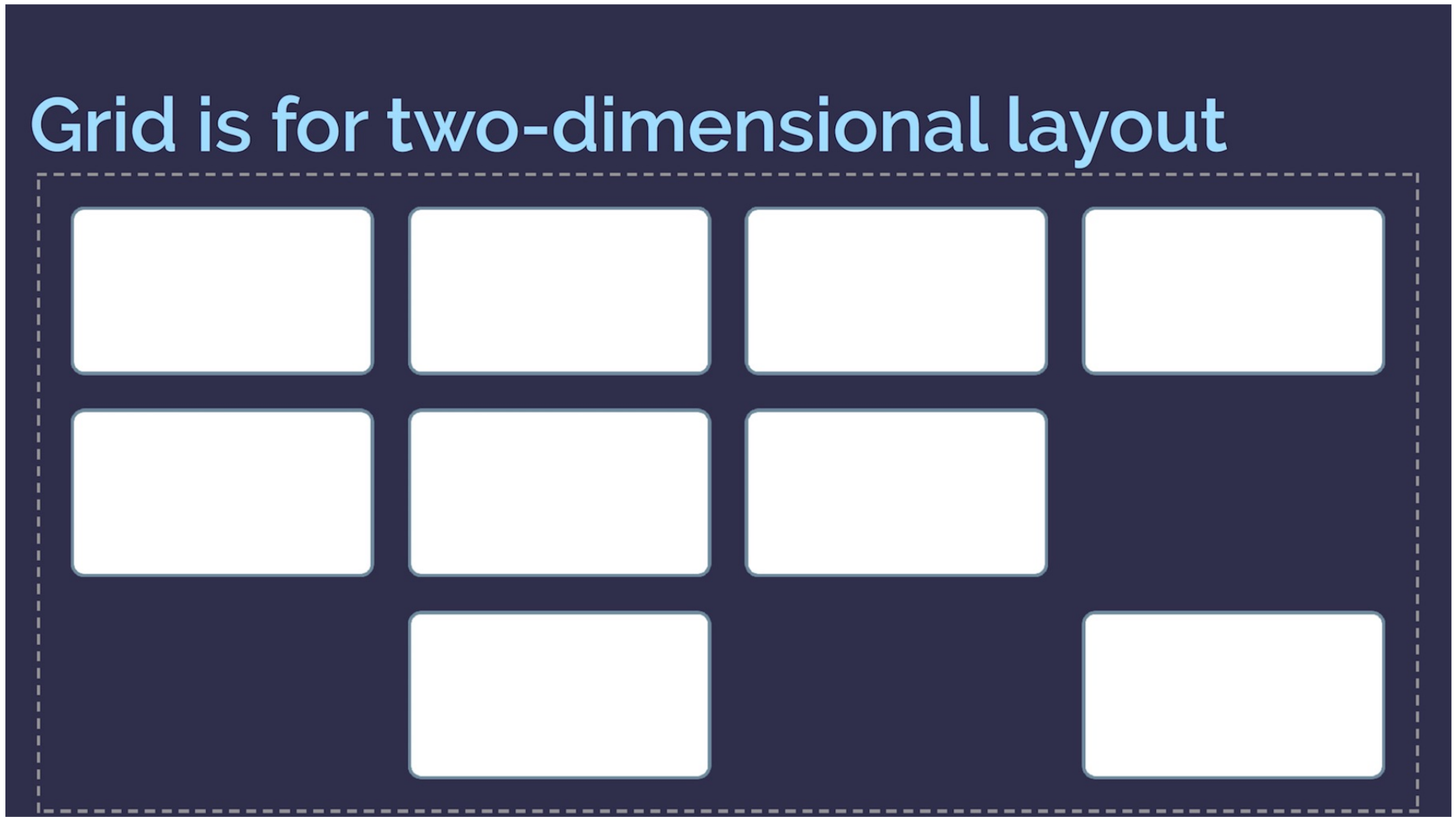
OVERVIEW

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

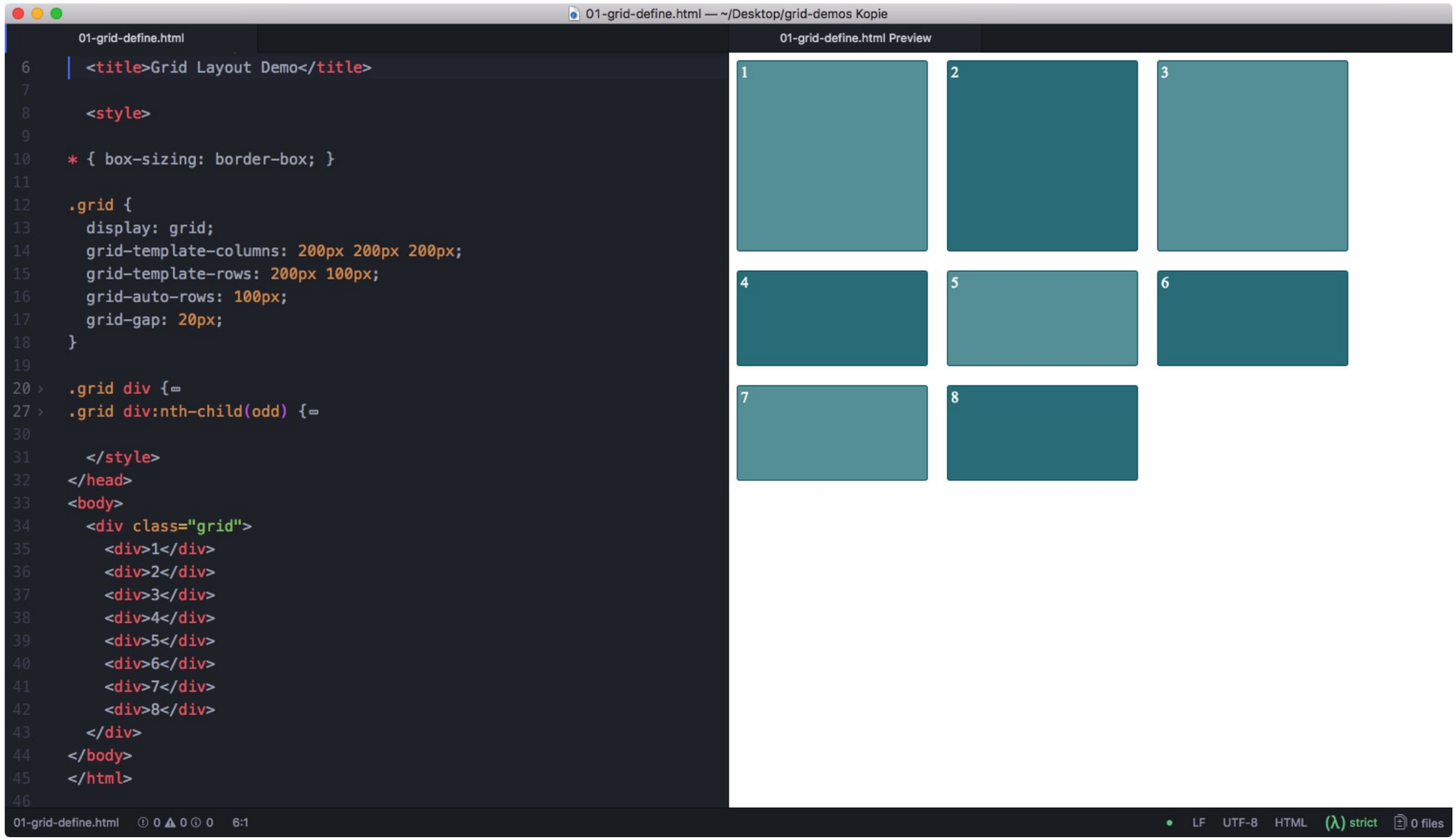
FLEXBOX



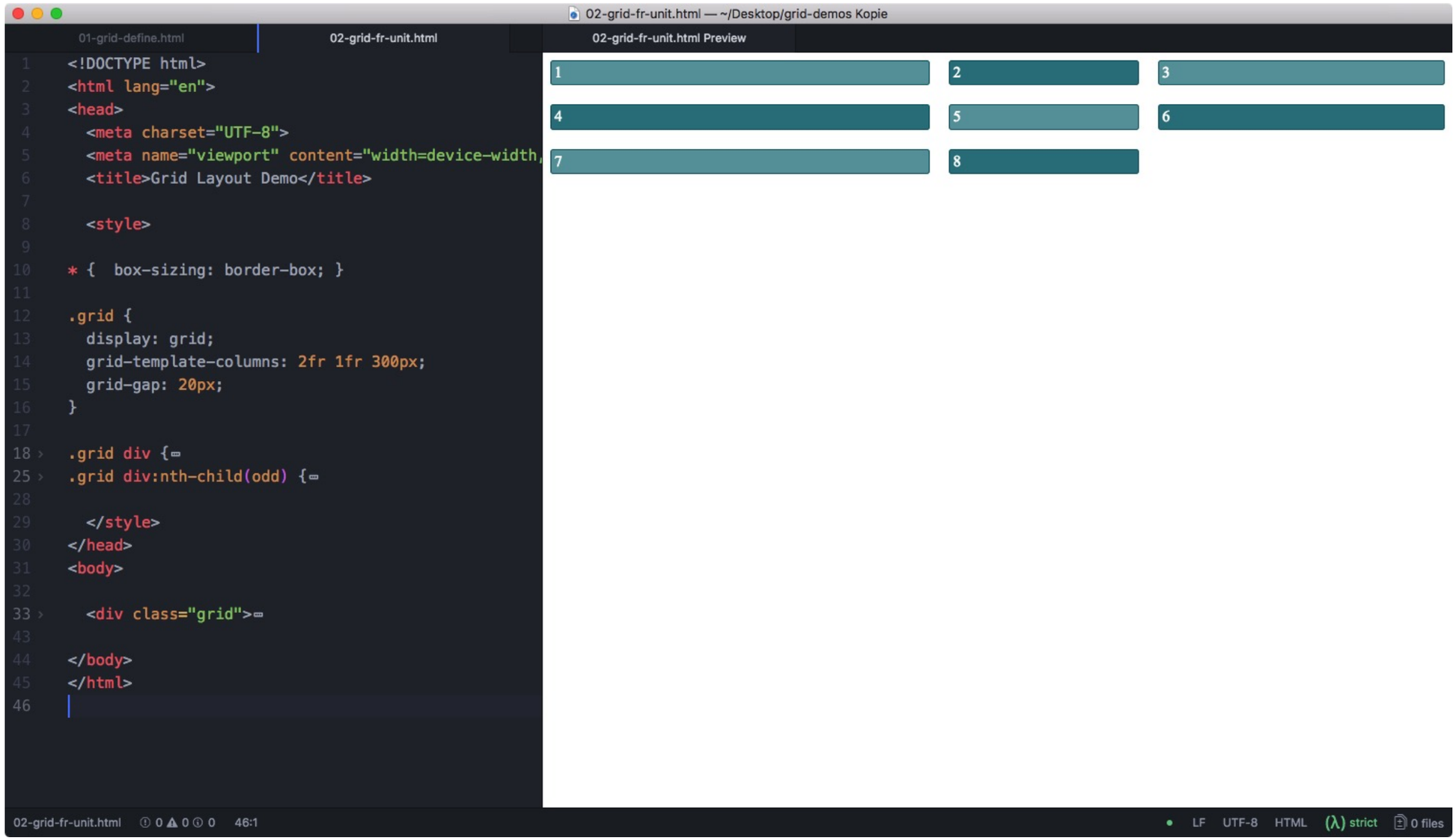
GRID



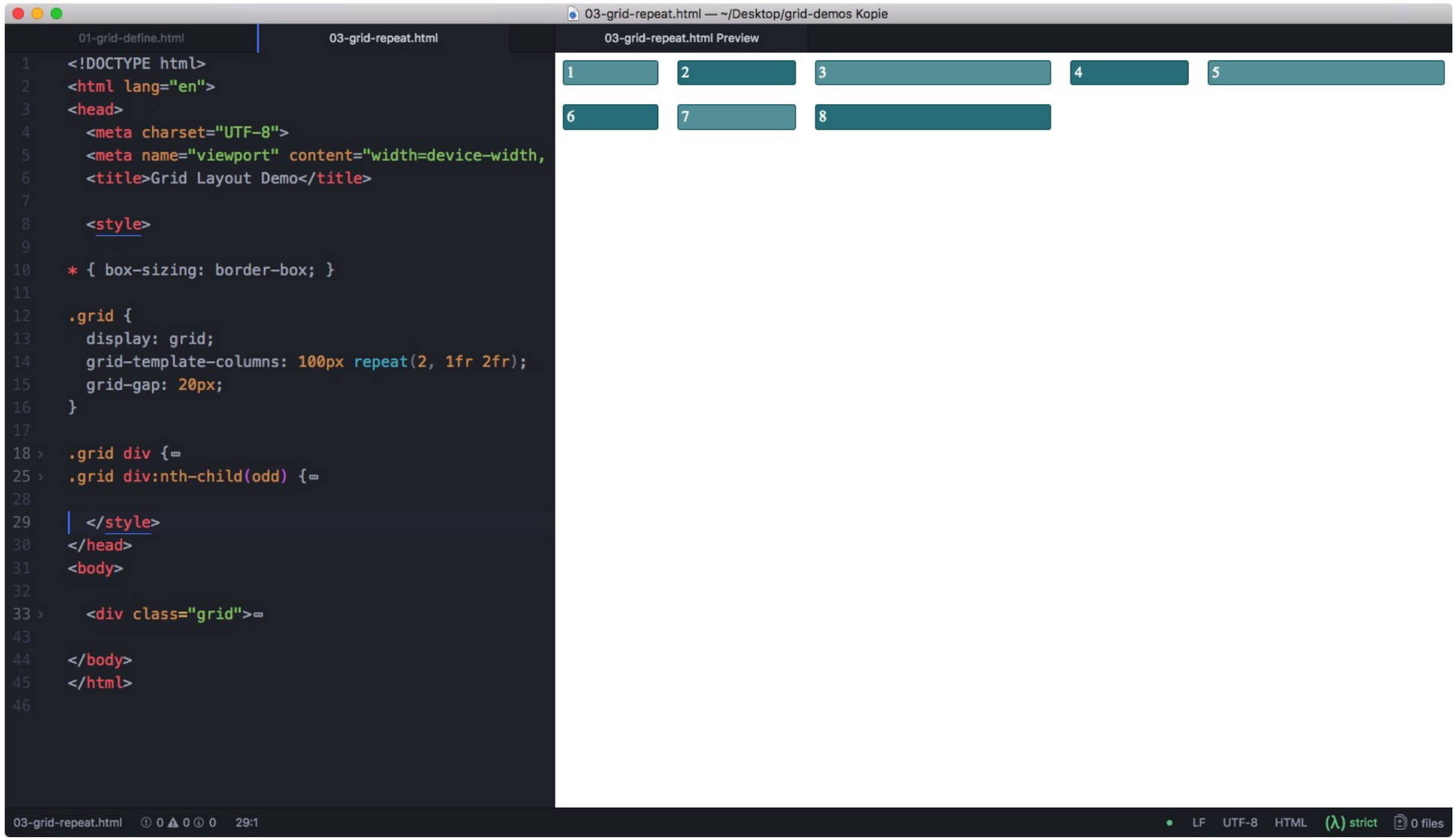
DEFINING A GRID



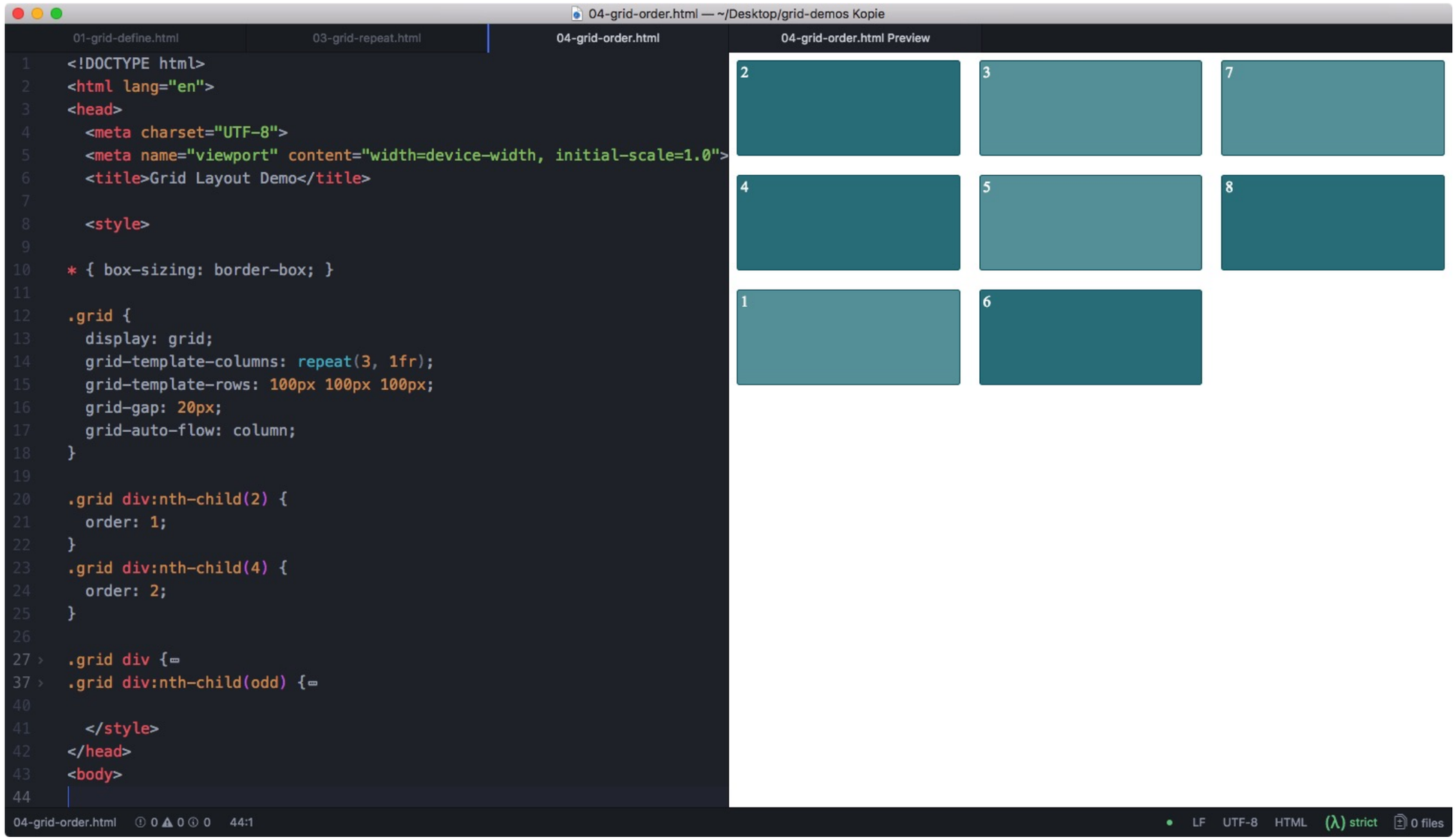
THE FR UNIT



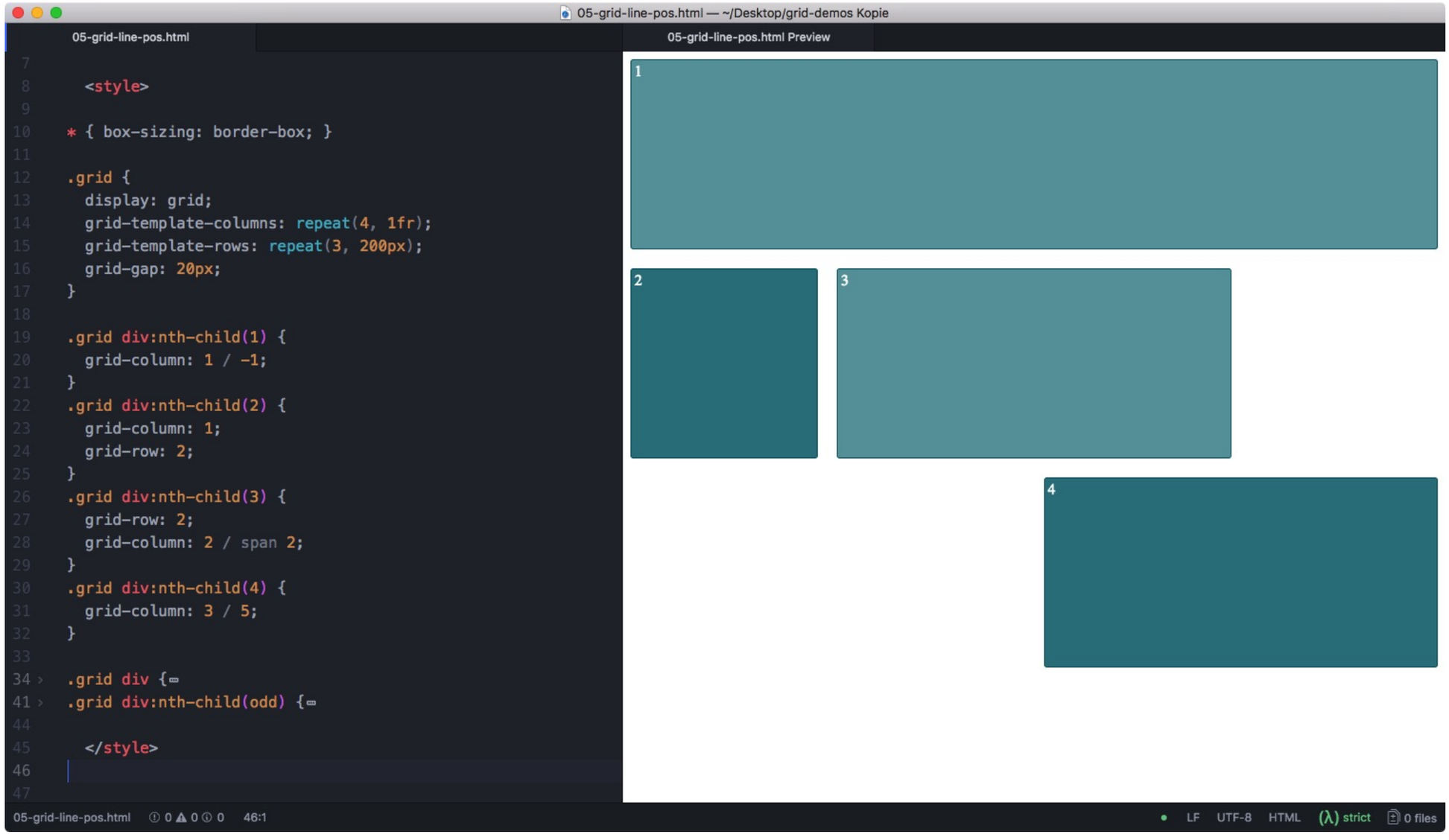
REPEAT NOTATION



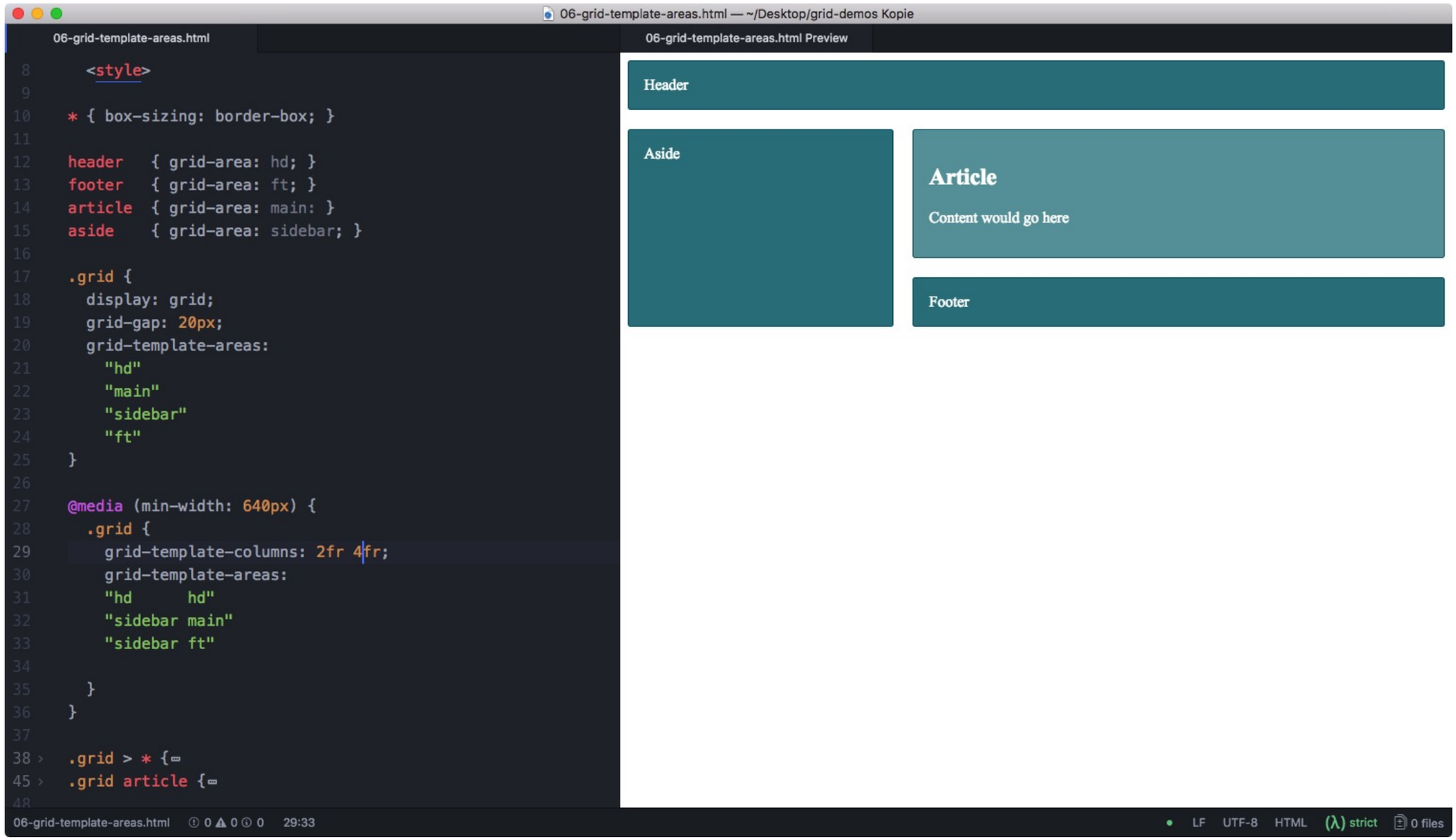
AUTO-PLACEMENT AND ORDER



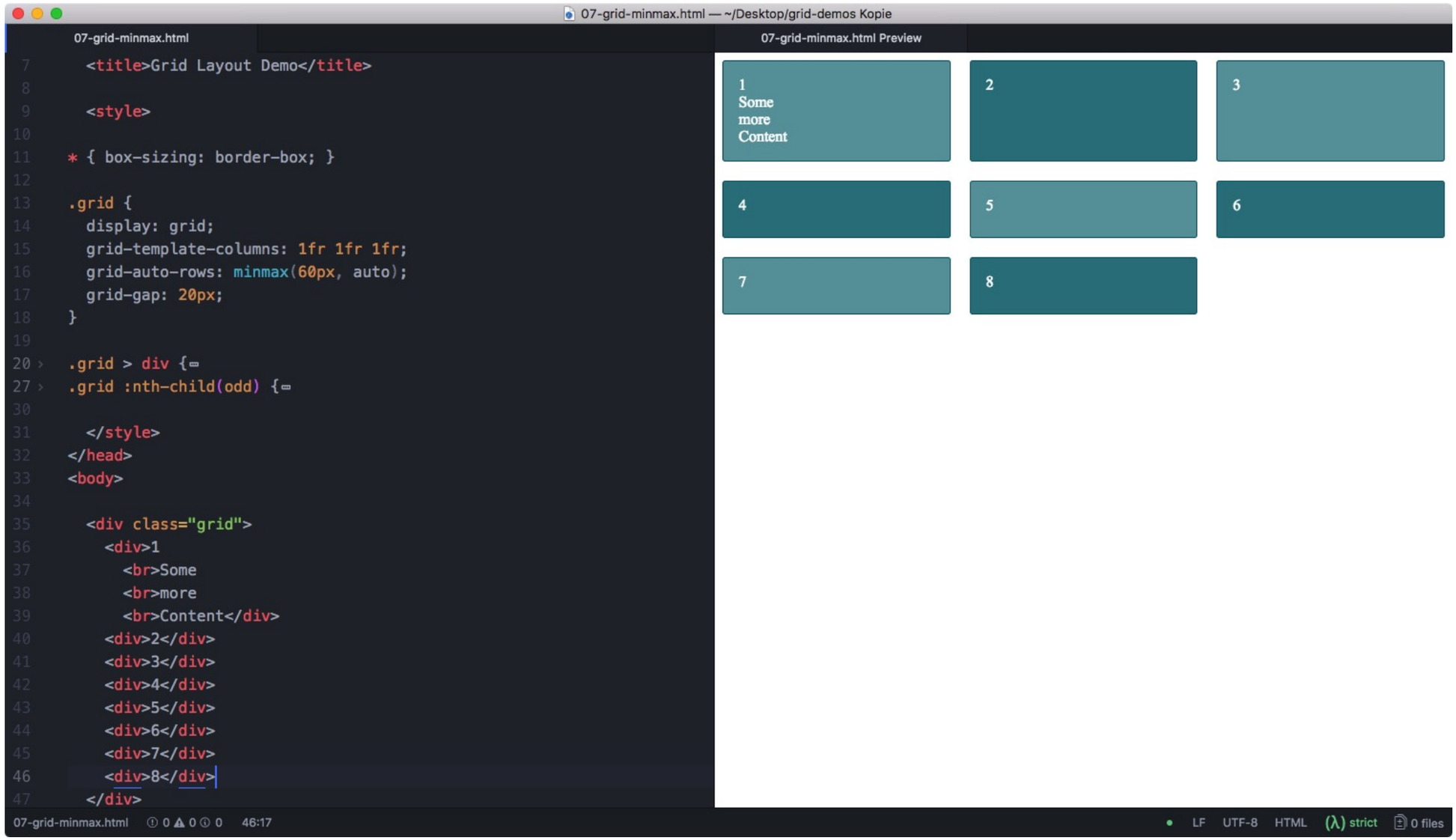
LINE-BASED PLACEMENT



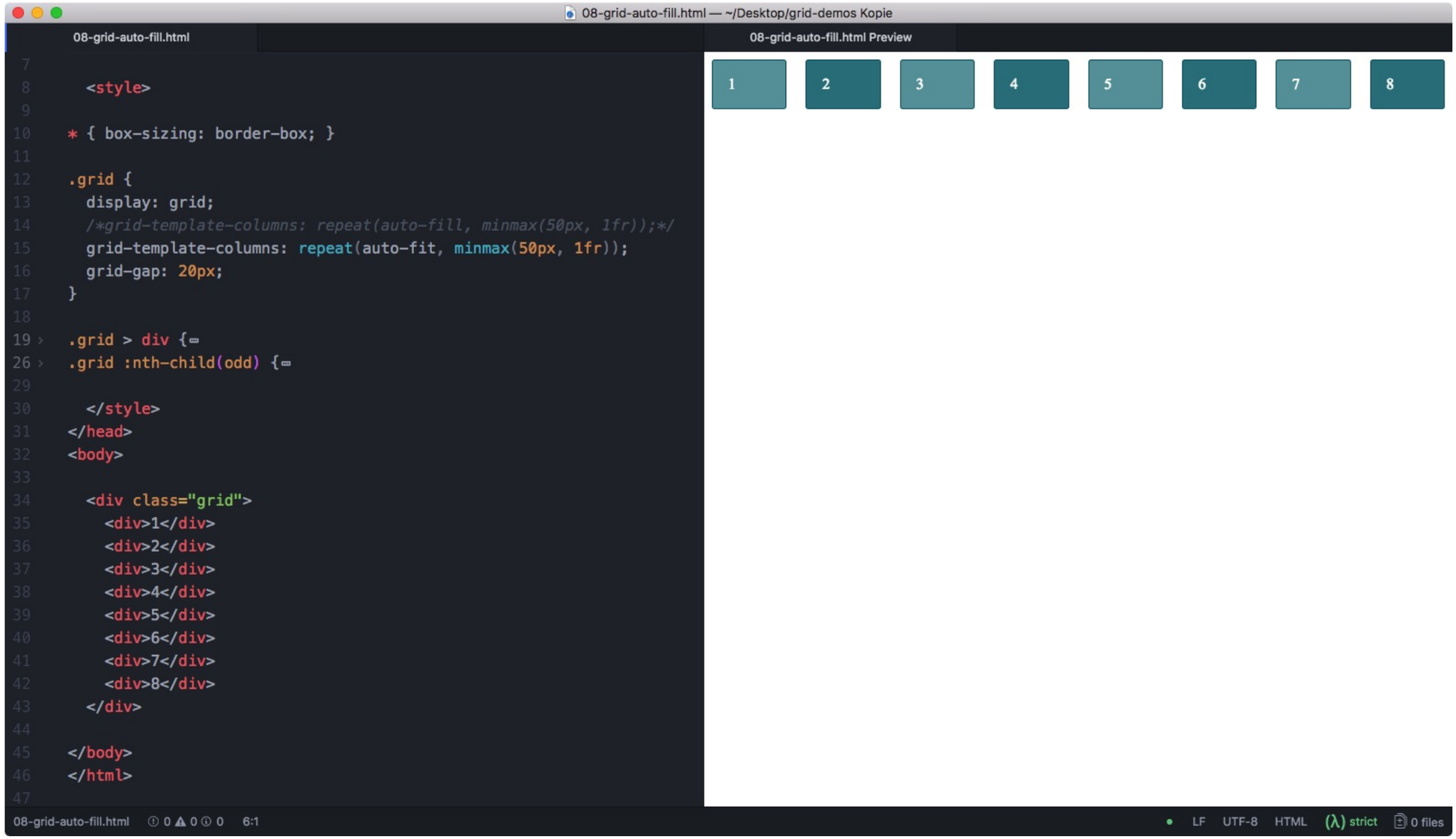
GRID TEMPLATE AREAS



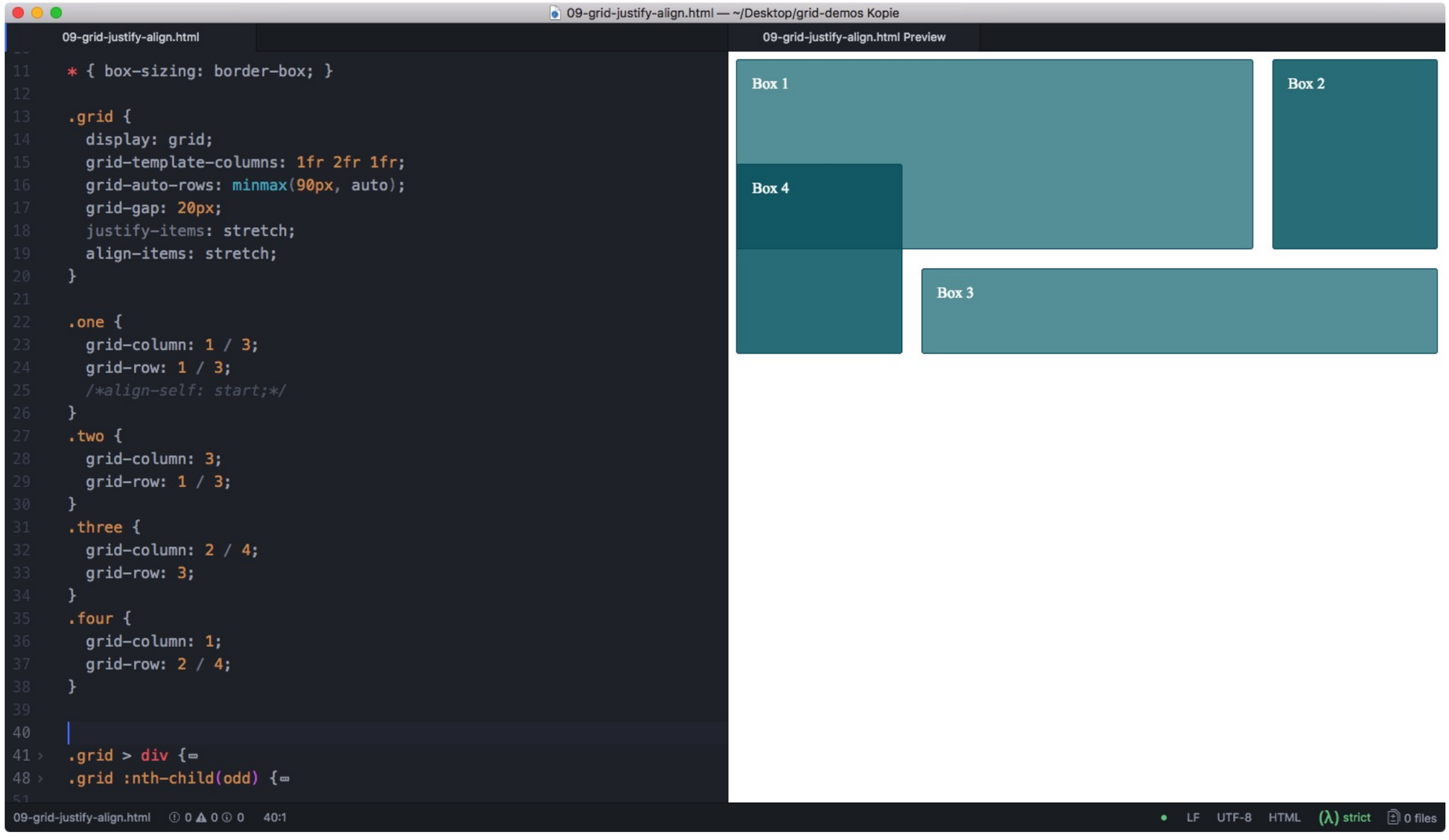
INTRODUCING MINMAX()



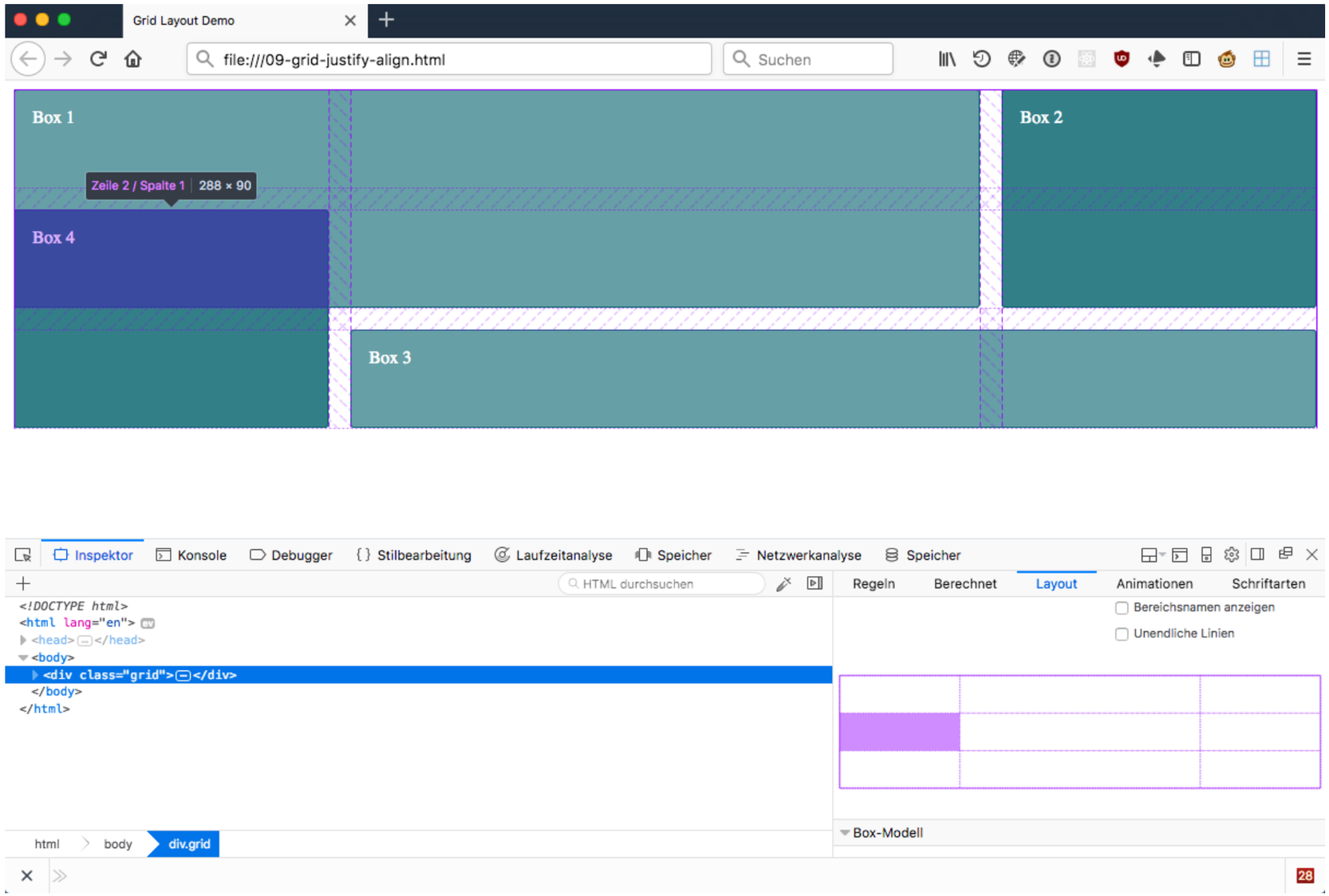
INTRODUCING AUTO-FILL AND AUTO-FIT



ALIGNING AND JUSTIFYING GRID ITEMS



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