INTRO APPWEB 2023

More CSS

Styling the Web

Luigi De Russis

Photo by Marcus Ganahl on Unsplash

Politecnico di Torino

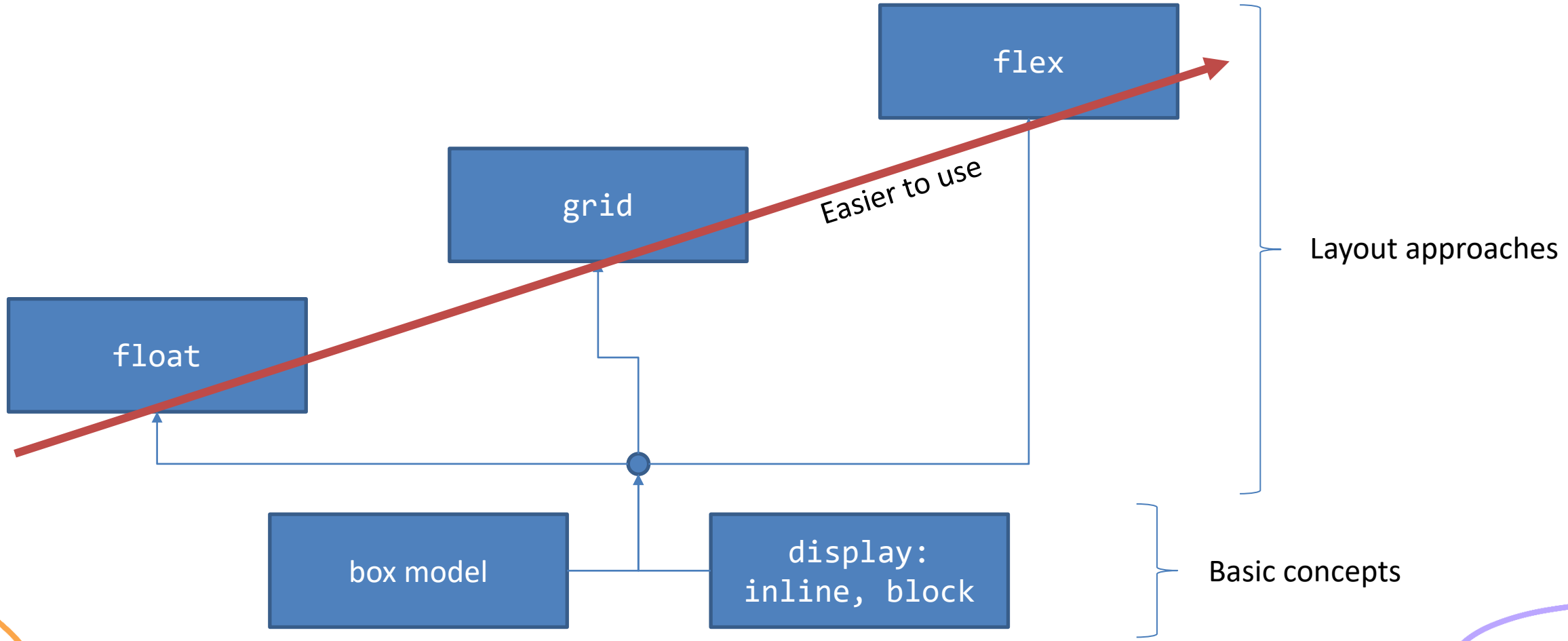
Goal

- Advanced layout in web pages
- Responsive layouts
- Libraries

Outline

- Page layout with
  - Grid
  - Flex
- Other flow layouts
- Responsive layout
- CSS Framework: Bootstrap

Page Layout Methods



Page Layout

GRIDS

Advanced Layout: Grid

Maki-zushi

The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.

Nigiri-zushi

The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.

Temaki-zushi

Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.

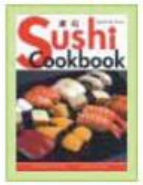
WHAT IS SUSHI?

Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.

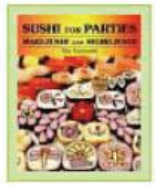
Technically, the word `sushi' refers to the rice, but colloquially, the term is used to describe a finger-size piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption or raw fish in the Japanese style (while sushi is not solely a Japanese invention, these days, the Japanese style is considered the de facto serving standard).

Sashimi

Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..



QUICK & EASY SUSHI COOKBOOK
This book has great pictures, however it is not as complete as Sushi Made Easy.



SUSHI FOR PARTIES: MAKI-ZUSHI AND NIGIRI-ZUSHI
This book also has great pictures, with advanced maki (cut roll) making techniques.

SUSHI MADE EASY
A very decent all-around book for the money.

Advanced Layout: Grid

Maki-zushi

a

The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.

a

Nigiri-zushi

c

The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.

Temaki-zushi

d

Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.

WHAT IS SUSHI?

e

Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.

e

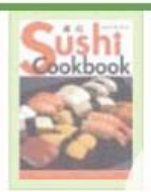
Technically, the word "sushi" refers to the rice, but colloquially, the term is used to describe a finger-sized piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption of raw fish in the Japanese style (while sushi is not solely a Japanese invention, these days, the Japanese style is considered the de facto serving standard).

g

Sashimi

i

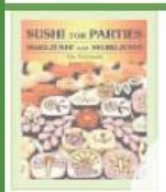
Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..



j

QUICK & EASY SUSHI COOKBOOK

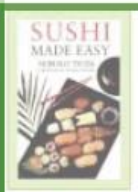
This book has great pictures, however it is not as complete as Sushi Made Easy.



k

SUSHI FOR PARTIES, MAKI-ZUSHI AND NIGIRI-ZUSHI

This book also has great pictures, with advanced maki (cut roll) making techniques.



l

SUSHI MADE EASY

A very decent all-around book for the money.

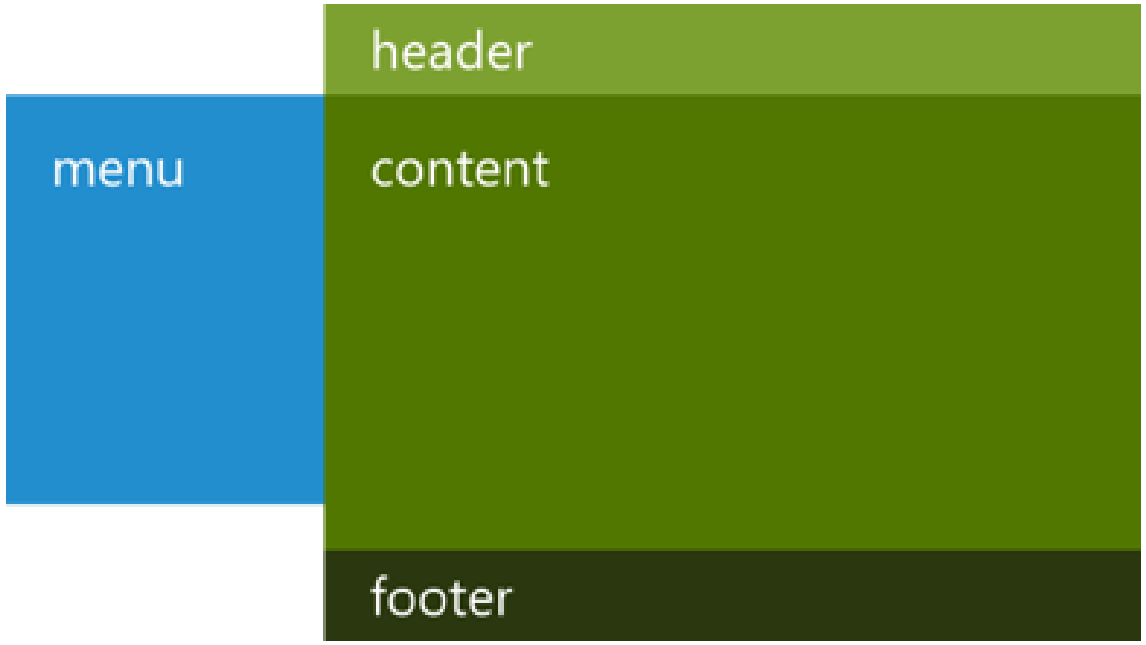
Advanced Layout: Grid

- It is possible to define a grid in which content can flow or be placed, or that remain empty
- There are 3 ways to define a grid
  - Explicit grid: defined with '`grid-template-columns`' and '`grid-template-rows`' properties
  - Natural grid: automatically created by elements with a natural grid structure (multi-column elements and tables)
  - Default grid: all other block elements define a single-cell grid

Example

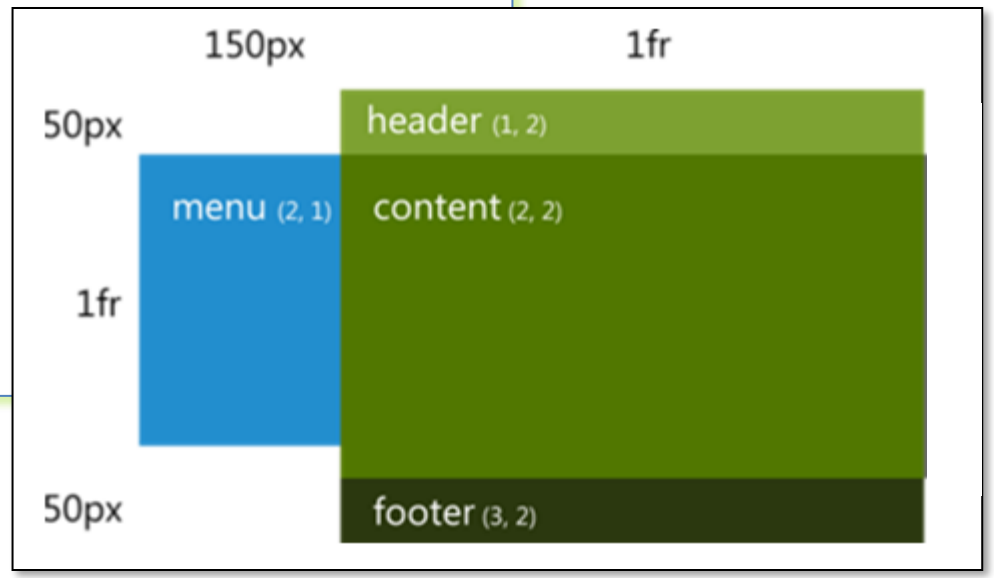
- Classic two-column layout

```
<section>
  <header>Title</header>
  <aside>Menu</aside>
  <article>Content</article>
  <footer>Footer</footer>
</section>
```



Example

```
section {
  display: grid;
  grid-template-columns: 150px 1fr;
  grid-template-rows: 50px 1fr 50px; }
section header {
  grid-column: 2;
  grid-row: 1; }
section aside {
  grid-column: 1;
  grid-row: 2; }
section article {
  grid-column: 2;
  grid-row: 2; }
section footer {
  grid-column: 2;
  grid-row: 3; }
```

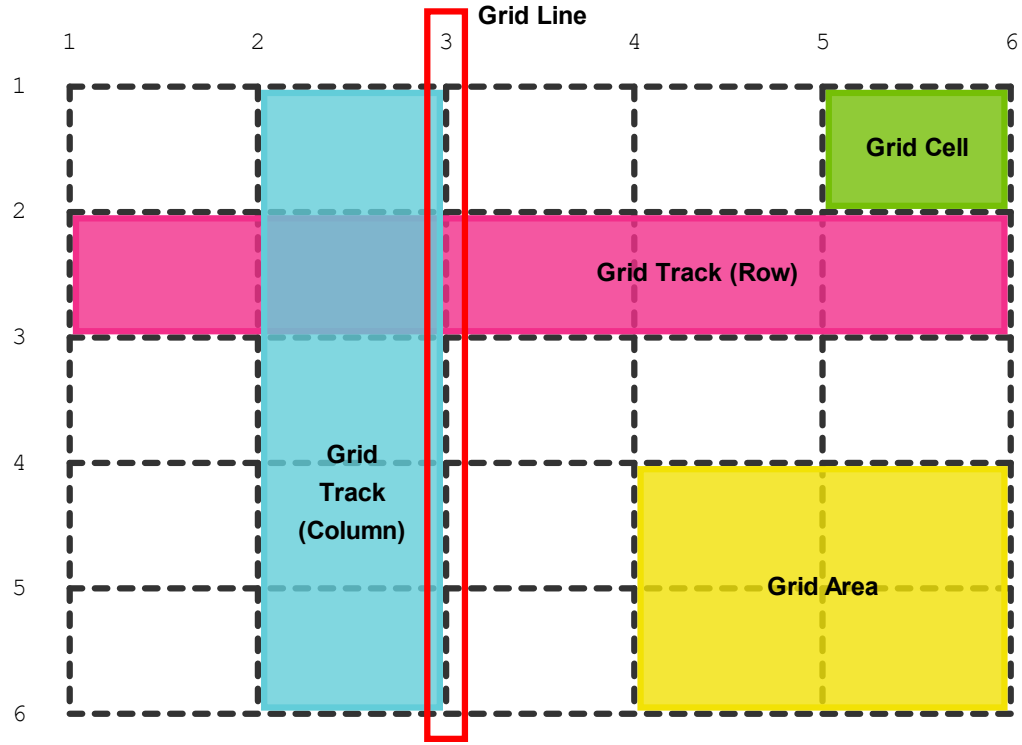


fr = fraction values

- new unit applicable to grid-template-rows and grid-template-columns properties

Suggested Reference for Grid Layout

- https://webkit.org/blog/7434/css-grid-layout-a-new-layout-module-for-the-web/



Page Layout

FLEX

Flexbox

- Alternative to floats/grids for defining the overall appearance of a web page
- Flexbox gives complete control over the alignment, direction, order, and size of boxes

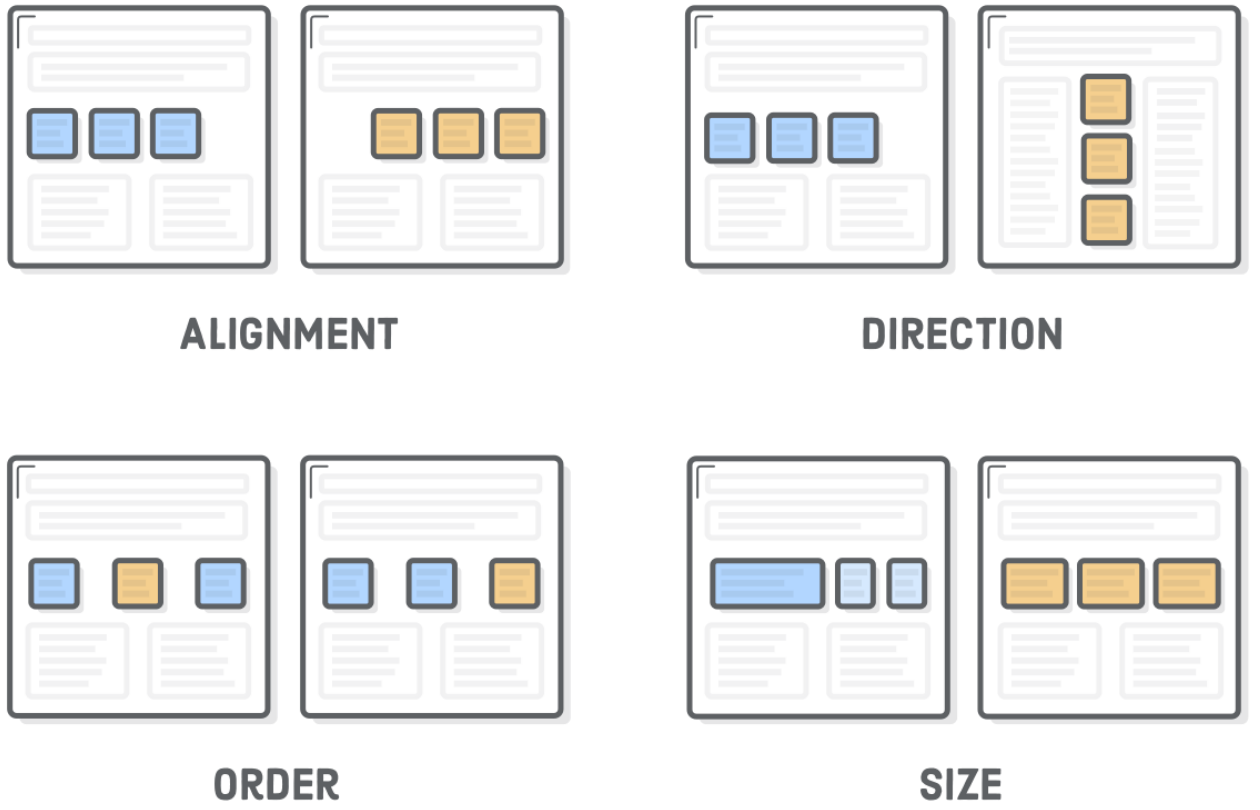
FLOATS

(MAGAZINE-STYLE LAYOUTS)

FLEXBOX

(OVERALL PAGE STRUCTURE)

Flexbox



https://internetingishard.com/html-and-css/flexbox/

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

Flexbox

- Flexbox uses two types of boxes
  - Flex containers: group a set of flex items and define how they're positioned
  - Flex items
- Every HTML element that is a direct child of a flex container is an item

"FLEX CONTAINER"

"FLEX ITEMS"

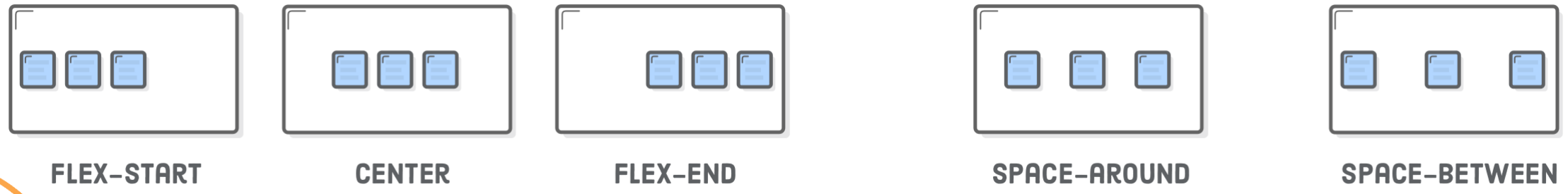
Horizontal Alignment

```
.menu-container {
  /* ... */
  display: flex;
  justify-content: center;
}
```

- To turn one HTML elements into a flex container:

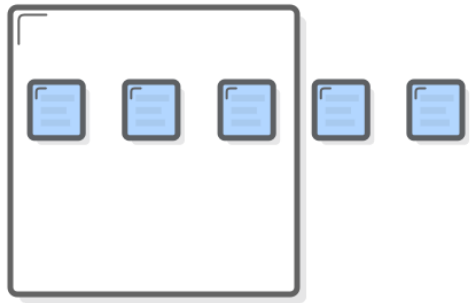
  `{ display:flex ; }`

- "`justify-content`" property defines the horizontal alignment of its items
  - center, flex-start, flex-end
  - space-around, space-between



Wrapping

- The `flex-wrap` property creates a grid
  - Then, you can change alignment, direction, order, and size of items



NO WRAPPING

FLEX-WRAP: NOWRAP;

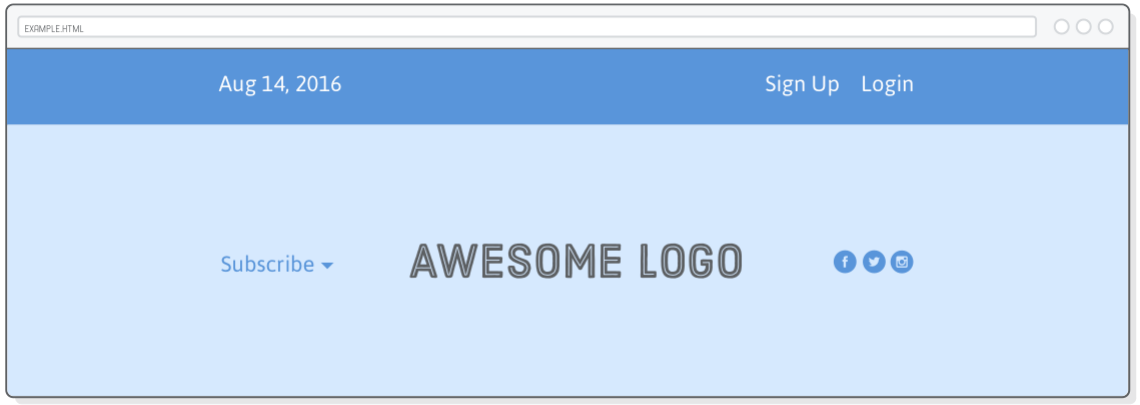
WITH WRAPPING

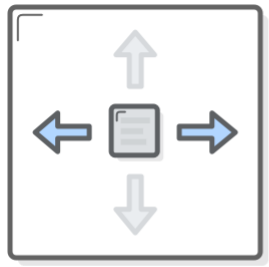
FLEX-WRAP: WRAP;

```
.photo-grid {
  width: 900px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}
```

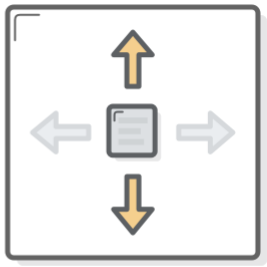
Vertical Alignment

- Flex containers can also define the vertical alignment of their items





JUSTIFY-CONTENT



ALIGN-ITEMS

```
.header {
  width: 900px;
  height: 300px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Direction

Refers to whether a container renders its items horizontally or vertically

ROW

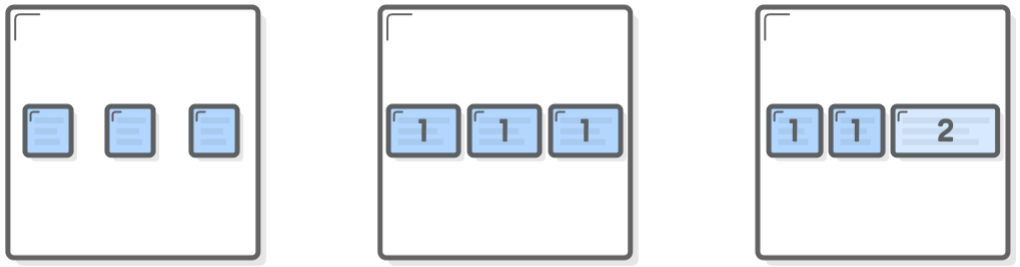
FLEX-DIRECTION: ROW;

COLUMN

FLEX-DIRECTION: COLUMN;

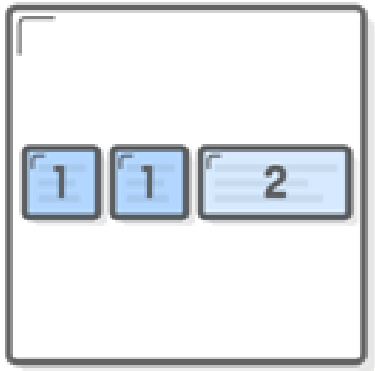
Flexible Items

- Flex items are flexible: they can shrink and stretch to match the width of their containers
- The `flex` property defines the width of individual items in a flex container
  - a *weight* that tells the flex container how to distribute extra space to each item
  - E.g., an item with a flex value of 2 will grow twice as fast as items with the default value of 1



Flexible Items: Example

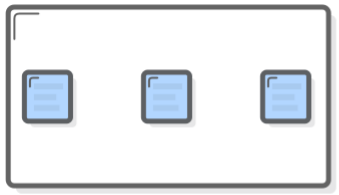
```
.footer {
  display: flex;
  justify-content: space-between;
}

.footer-item {
  border: 1px solid #fff;
  background-color: #D6E9FE;
  height: 200px;
  flex: 1; }

.footer-three { flex: 2; }
```




```
<div class='footer'>
  <div class='footer-item footer-one'></div>
  <div class='footer-item footer-two'></div>
  <div class='footer-item footer-three'></div>
</div>
```

Grouping

- Flex containers only know how to position elements that are one level deep (i.e., their child elements)
  - You can group flex items using



NO GROUPING
(3 FLEX ITEMS)

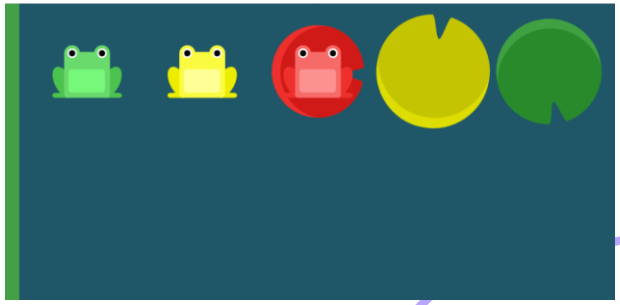
GROUPED ITEMS
(2 FLEX ITEMS)

Summary of Flexbox

- `display: flex` to create a flex container
- `justify-content` to define the horizontal alignment of items
- `align-items` to define the vertical alignment of items
- `flex-direction` if you need columns instead of rows
- `row-reverse` or `column-reverse` values to flip item order
- `order` to customize the order of individual elements
- `align-self` to vertically align individual items
- `flex` to create flexible boxes that can stretch and shrink

References for Flexbox

- Interneting is hard flexbox tutorial
  - https://internetingishard.com/html-and-css/flexbox/
- A complete guide to flexbox
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- W3schools
  - https://www.w3schools.com/css/css3_flexbox.asp
- Flexbox, guida pratica
  - http://www.html.it/guide/flexbox-guida-pratica/
- Flexbox Froggy (a game-like tutorial)
  - http://flexboxfroggy.com/

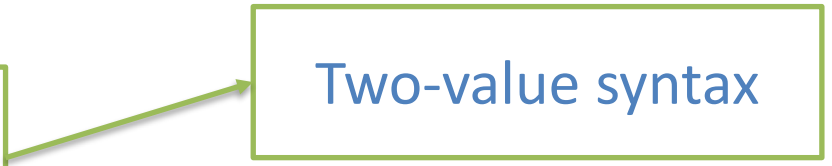


Advanced “display”

OTHER LAYOUTS

Property Keywords

- `flex`, `grid`, and `table` are inside keywords of the `display` property
- They specify the element's inner display type
  - The type of formatting context that its contents are laid out in
- `block` and `inline` are, instead, outside keywords
  - The element role in the flow layout
- The default inside keyword is `flow`
  - `display: block` == `display: block flow`
  - `display: flex` == `display: block flex`



Property Keyworks

- Browsers that support the two-value syntax, on finding the inner value only, will set their outer value to block

- Browsers that support the two-value syntax, on finding the outer value only, will set the inner value to flow

`flow-root`

- Another inside keyword for the display property
  - not experimental
- The element with this property generates a `block` element box that establishes a new block formatting context (BFC)
- BFC defines *where* the formatting root lies (typically: `<html>` and `floats`)
  - The region in which the layout of block boxes occurs and in which floats interact with other elements

`flow-root`

- Two-value syntax
  - `display: block flow-root`
  - `display: inline flow-root`

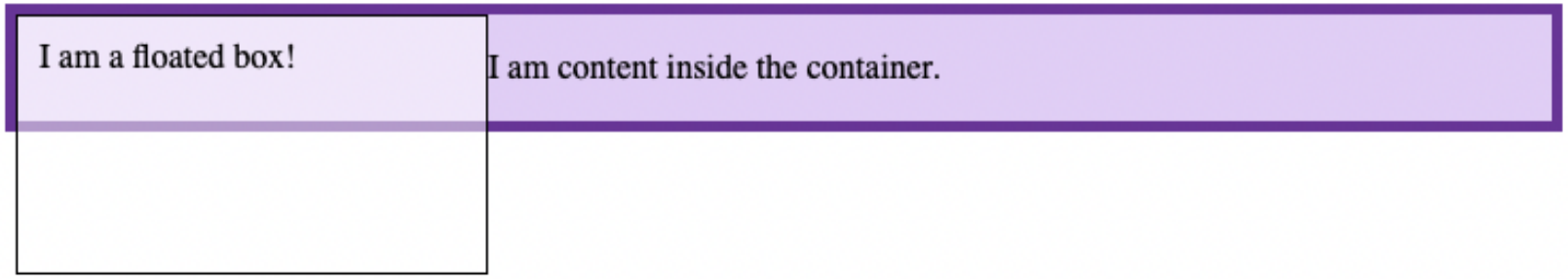
- Single-value syntax
  - `display: flow-root`
  - `display: inline-block`

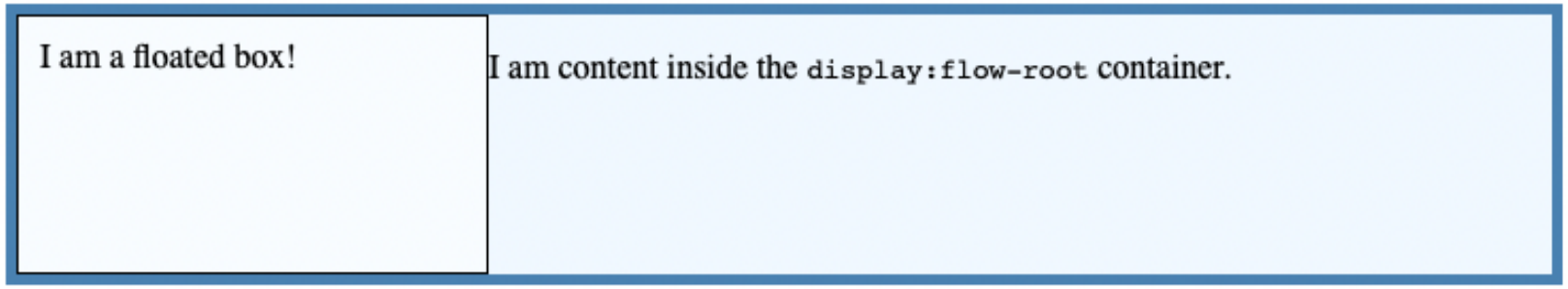
Example

```
<section>
  <div class="box">
    <div class="float">I am a floated box!</div>
    <p>I am content inside the container.</p>
  </div>
</section>
<section>
  <div class="box" id="flow-root">
    <div class="float">I am a floated box!</div>
    <p>I am content inside the <code>display:flow-root</code>
container.</p>
  </div>
</section>
```

```
section {
  height: 150px;
}
.box {
  background-color: rgb(224, 206, 247);
  border: 5px solid rebeccapurple;
}
#flow-root {
  display: flow-root;
  background-color: aliceblue;
  border: 5px solid steelblue;
}
.float {
  float: left;
  width: 200px;
  height: 100px;
  background-color: rgba(255, 255, 255, 0.5);
  border: 1px solid black;
  padding: 10px;
}
```

Example


I am a floated box!
I am content inside the container.


I am a floated box!
I am content inside the display:flow-root container.

Remember This?

With `inline-block`

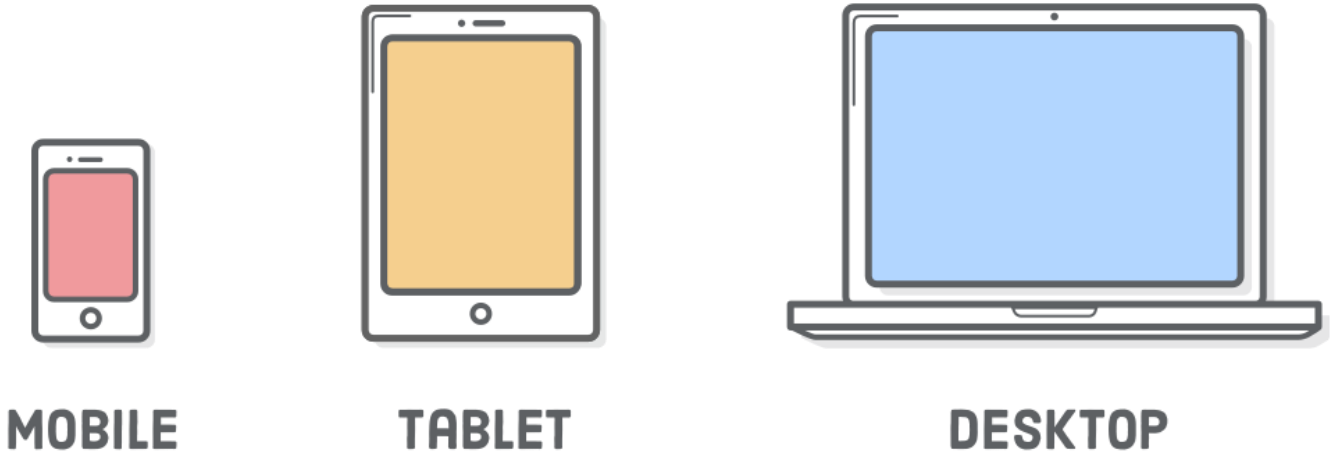
```
header > div {
  display: inline-block;
}

header > nav {
  display: inline-block;
}
```

```
<header>
  <div>
    <h1>IAW Blog</h1>
    <h2>Tutto su Introduzione alle Applicazioni Web</h2>
  </div>
  <nav>
    <ul>
      <li><a href="home.html" title="Homepage">Home</a></li>
      <li><a href="presentazione.html" title="Di cosa si
parla">Presentazione</a></li>
      <li><a href="contatti.html" title="Chi sono">Contatti</a></li>
    </ul>
    <a href="login.html" title="Entra nel sito">Login</a>
  </nav>
</header>
```

RESPONSIVE LAYOUT

Responsive Design

- Display well in everything from widescreen monitors to mobile phones
- Approach to web design to eliminate the distinction between the mobile-friendly version of your website and its desktop counterpart



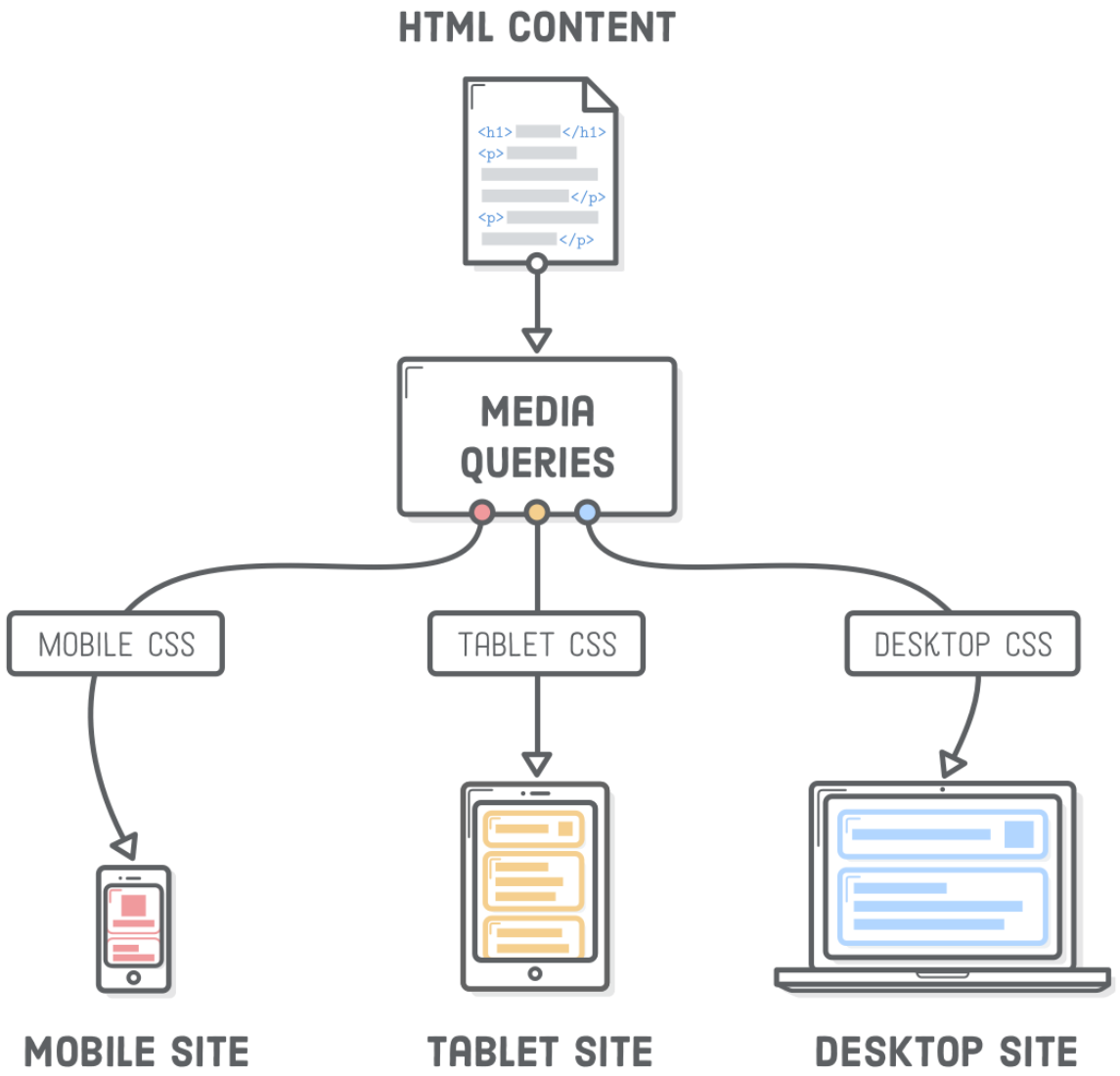
https://internetingishard.com/html-and-css/responsive-design/

Responsive Design

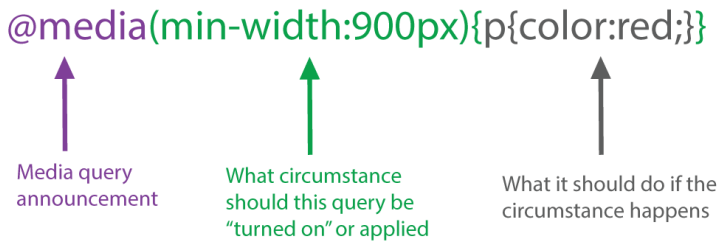
- Responsive design is accomplished through CSS “media queries”
- A way to *conditionally* apply CSS rules

```
@media(min-width:900px){p{color:red;}}
```

- Media query announcement
- What circumstance should this query be “turned on” or applied
- What it should do if the circumstance happens



Media Queries



- Composed of an (optional) media type and any number of media feature expressions
- Media type
  - category of device for which the query applies
  - `all` (default), `print`, `screen`
- Media feature
  - specific characteristic of the user agent or output device
  - height, width, orientation, aspect ratio, …

Combining Media Queries

Mostly use Boolean operators

```
@media (min-width: 30em) and (orientation: landscape) {
  /* Restrict styles to landscape-oriented devices with
a width of at least 30 ems  */
}
```

```
@media (min-height: 680px), screen and (orientation: portrait) {
 /* Apply the style if the user's device has either a minimum
height of 680px or is a screen device in portrait mode  */
}
```

```
@media not screen and (color), print and (color) {
  /* Apply the style if it’s not a colored screen nor a colored printer */
}
```

Range Syntax for Media Queries (NEW)

You can now use <, <=, >, and >= to specify width and height

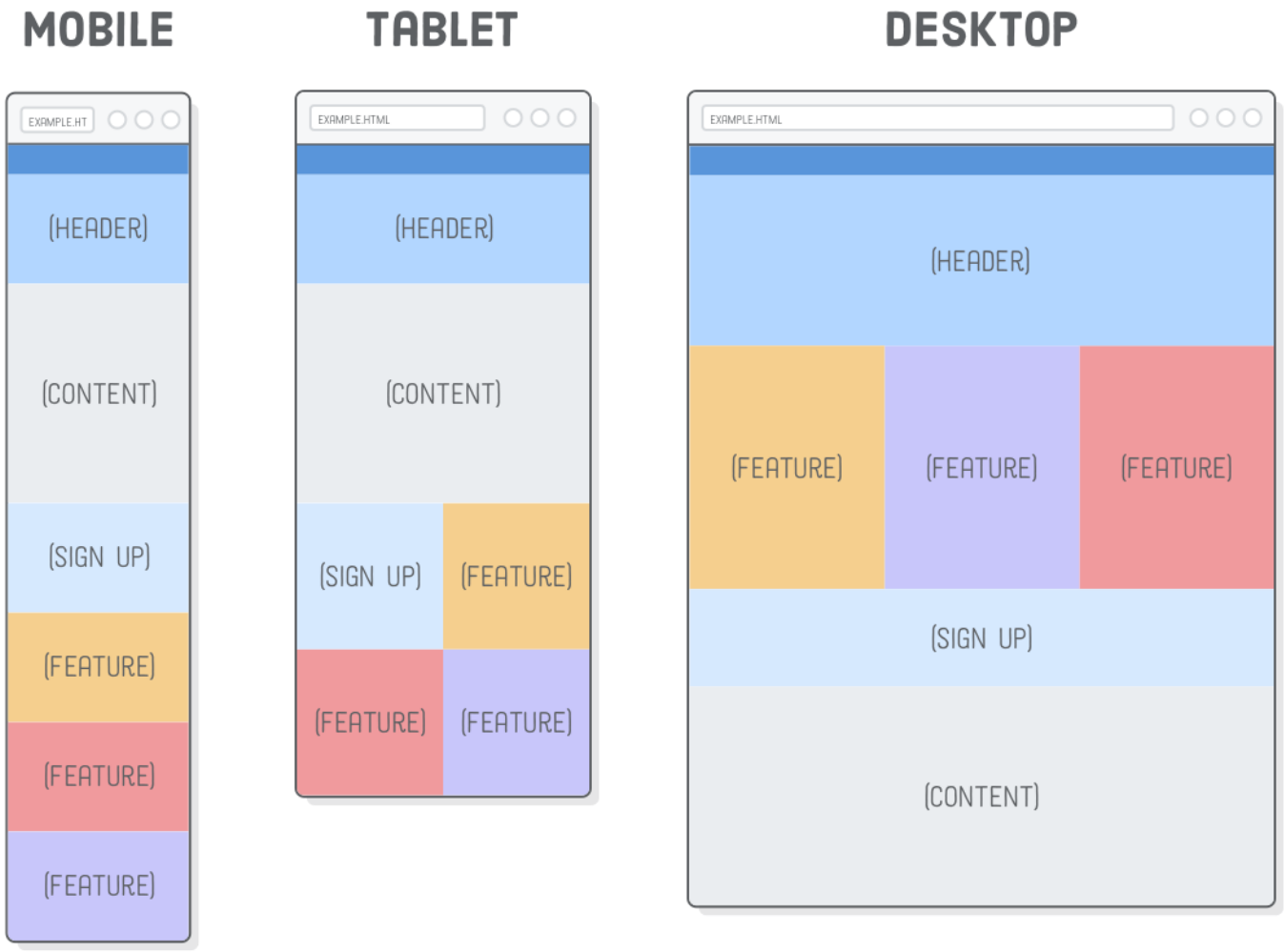
```
@media (min-width: 30em) and (max-width: 50em) {
  /* … */
}
```

```
@media (30em <= width <= 50em) {
  /* … */
}
```

Beware: compatibility! https://caniuse.com/

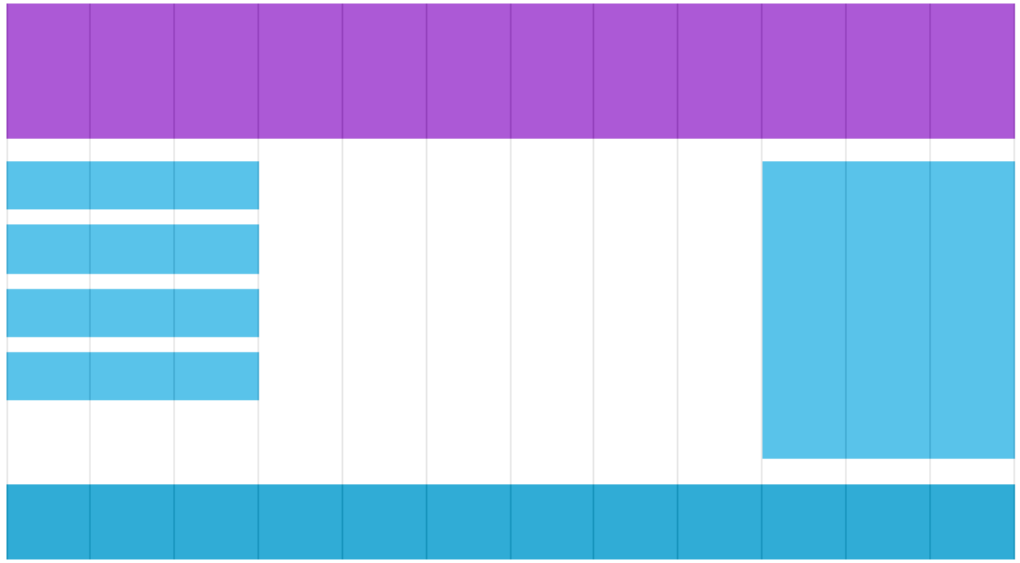
Chrome	Edge *	Safari	Firefox	Opera	IE
4-103	12-103		2-62		
104-105	104-105	3.1-15.6	63-104	10-90	6-10
106	106	16.0	105	91	11
107-109		16.1-TP	106-107		

Layout for Responsive Design



Grid-view

- Many web pages are based on a grid-view, i.e., the page is divided into columns
- A responsive grid-view often has 12 columns, a total width of 100%, and will shrink and expand as you resize the browser window



CSS Frameworks

- Set of templates to simplify web development
- Example: Bootstrap
  - Open-Source CSS (and JavaScript) framework
  - Allows applying “modern” styles with sensible and nice-looking defaults
    - Many ready-to-use UI elements (e.g., buttons, menus, tabs, collapsible items, etc.)
  - Takes care of cross-browser issues
  - Simplified layout model
  - Developed by Twitter
    - https://getbootstrap.com

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/