

Front-end



Development

Lesson 6: RWD & Bootstrap



University of Tehran



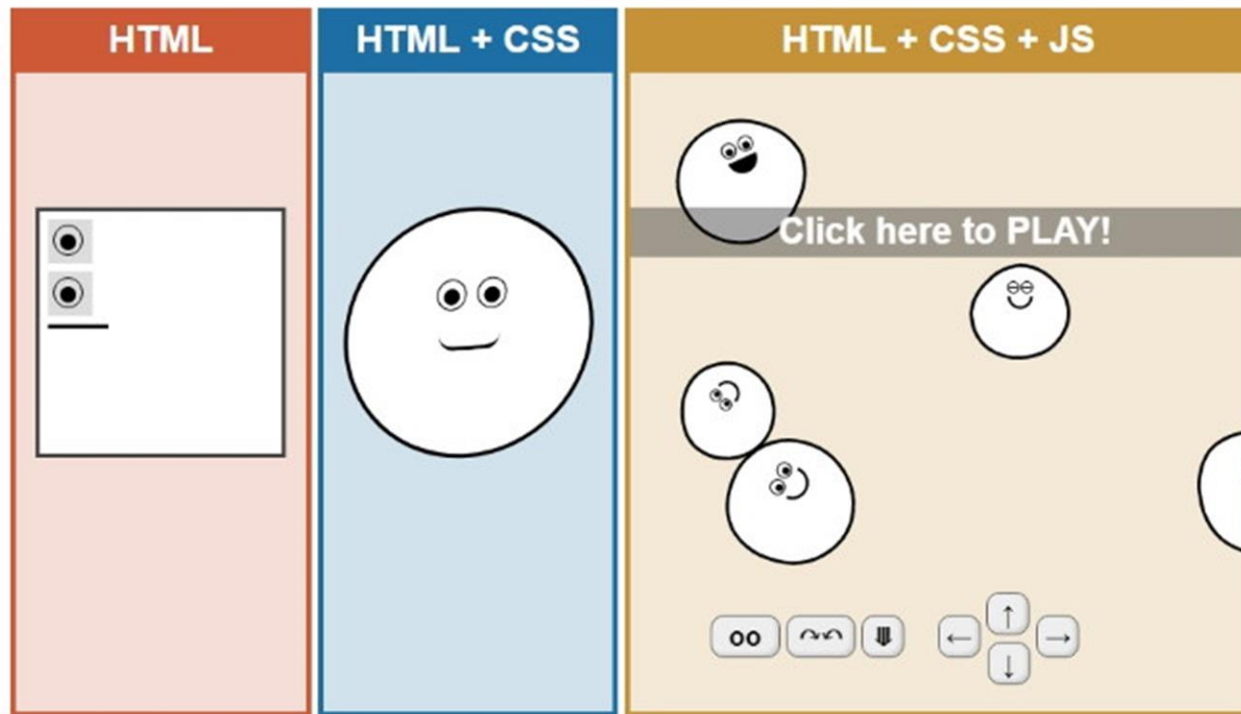
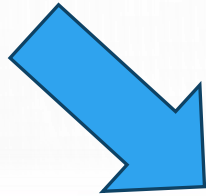
ACM
Summer of Code 2024

Lecturer:
Misagh Mohaghegh

A Short Review...

- CSS Position (static, relative, absolute, fixed, sticky)
- Flexbox container (direction, wrap, justify-content, align-items, align-content, gap)
- Flex items (align-self, flex-grow, flex-shrink, flex-basis, order)
- Variables (--var-name, var() function)
- Transform (translate, rotate, scale, skew)
- Transition (smooth transitions)

Continue...



Cascading Style Sheets





Responsive Design



Responsive Web Design (RWD)

Responsive design is when our page adapts to different screen sizes and displays content correctly on all. This often means that our CSS is used to resize, hide, or move content to make it good on any screen.



Desktop



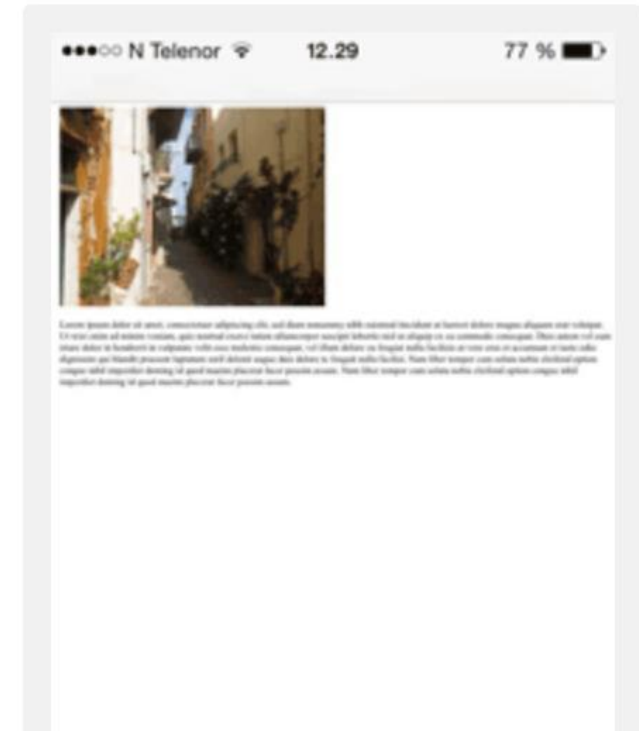
Tablet



Phone

Before Mobile

Before mobile phones, web pages were designed for computer screens. Computer viewport is too large to fit in a phone, so phone browsers scaled the entire page down. This means that the same computer page was zoomed out to fit on a phone screen.



Viewport Meta

The following meta tag should always be included for our responsive web page:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

width=device-width: this overrides the mobile browser default which is to render at a high viewport and zoom out to fit on screen.

Initial-scale=1.0: this sets the initial zoom level when the page is first loaded.

Basically, phones lie about their viewport width (for example, the device is 980px wide, but reports 360px) This is for CSS to understand that the device width is actually small, and use specific styles for it.

Flexible Web Pages

There are many CSS technologies that allow flexible web pages:

- Multi column layout (not used much)
- Flexbox
- Grid

We should also always use the following CSS reset property to avoid images overflowing their container:

```
img, picture, svg, canvas, video {  
  display: block;  
  max-width: 100%;  
}
```

Media Queries

Media queries allow us to check the user's situation and apply CSS for the cases. One usage is to check the user's screen width and change our CSS according to that.

Media queries are written using the **@media** at-rule.

An example media query is to check if the user has system dark mode enabled:

```
@media screen and (prefers-color-scheme: dark) {  
  body {  
    color: white;  
    background-color: black;  
  }  
}
```

Media Queries

After the `@media` at-rule, we specify the media type which can be **all**, **screen**, or **print**. If we don't specify it, it applies for all.

The rule we saw only changed the body colors for the screen, and not in printer view. This changes for all of them:

```
@media (prefers-color-scheme: dark) {  
  body {  
    color: white;  
    background-color: black;  
  }  
}
```

Viewport Media Query

We can use the **min-width** or **max-width** properties to change CSS based on the screen width:

```
@media (max-width: 768px) {  
  .flex-container {  
    gap: 2px;  
  }  
}
```

This media query states that if the screen size is below 768px, the flex gap should be lowered. min-width is the opposite and only applies if the screen size is above 768px.

Breakpoints

A **breakpoint** is a max or min-width size in which our website changes CSS.
We can specify many breakpoints with different media queries.

In the previous slide, a breakpoint was defined at 768px.

Desktop First Design

Desktop first design is when the default CSS that we write is written to work on desktop screens. This means that we are designing for a large screen width first, and using media queries for smaller screens. Media queries for desktop first design use the max-width property.

```
h1 {  
    font-size: 4rem;  
}  
  
@media (max-width: 768px) {  
    h1 {  
        font-size: 2rem;  
    }  
}
```

This means that the default font size is for large screens. If the screen size gets small (smaller than 768px), the font-size should get lowered.

Mobile First Design

Mobile first design is when our default CSS is meant for mobile devices. So we design for a small width, and use media queries for bigger screens. Mobile first design queries make use of the min-width property:

```
h1 {  
    font-size: 2rem;  
}  
  
@media (min-width: 768px) {  
    h1 {  
        font-size: 4rem;  
    }  
}
```

Mobile first design is the new trend and more websites are getting designed this way.

Fluid Container

A **fluid (full width) container** is a container that expands the whole screen width. To create a fluid container, we simply set the width of the container element to 100%. This makes it take the entire parent width. We should not change this 100% width in our breakpoints.

Fixed Width Container

A **boxed (fixed width) container** is a container that has a max-width that changes with the screen width. This means that no matter how large a screen gets, the content will stay in a 900px box for example. When the screen gets smaller, the box gets smaller in each breakpoint.

```
.fixed-width-container {  
  width: 900px;  
  margin: 0 auto;  
}  
  
@media (max-width: 1024px) {  
  .fixed-width-container {  
    width: 760px;  
  }  
}
```

Bootstrap





Bootstrap



Bootstrap

Bootstrap is a CSS framework that provides many premade classes and a grid architecture to make building web pages easier.

Some Bootstrap features require a JS script to be included.

Using Bootstrap, we write less CSS because there are already pre-defined classes that we can use. But there are still times when we need to write CSS.

The current version of Bootstrap is 5 (v5.3.3 specifically).

Bootstrap 3 and 4 are still supported but not recommended anymore.

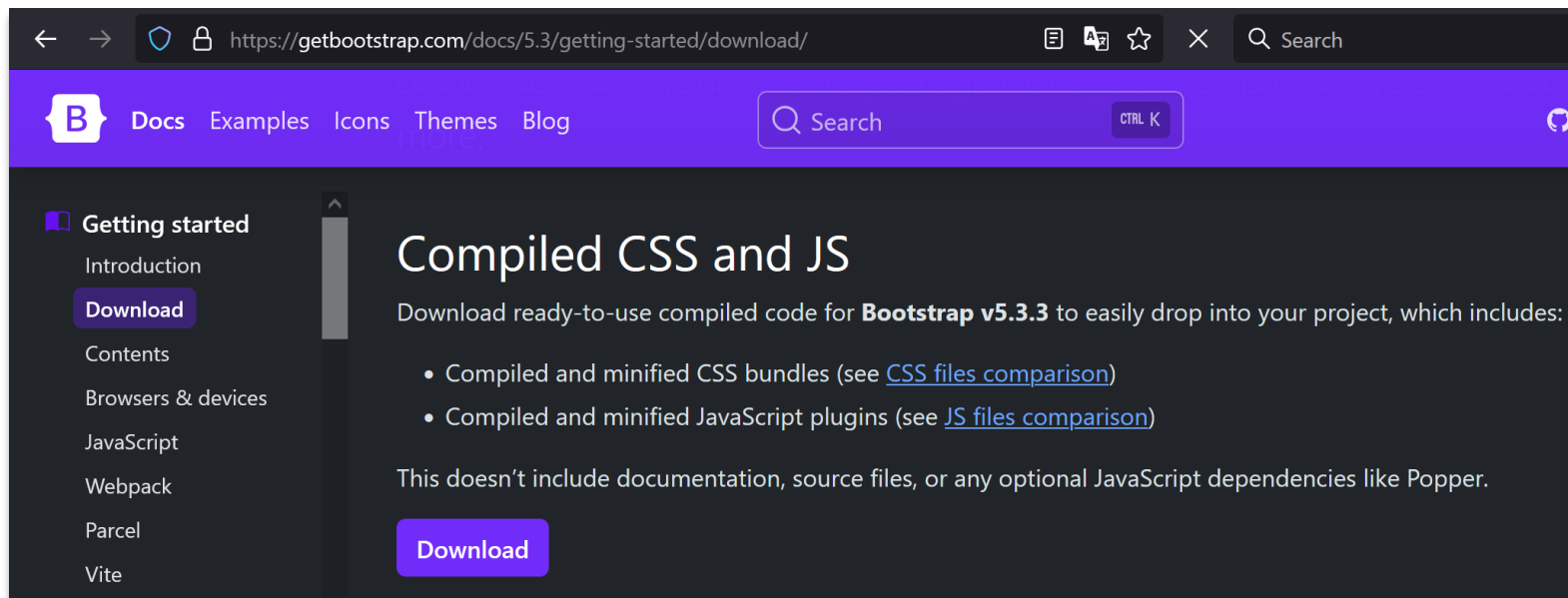
Bootstrap 5 removed the jQuery dependency and uses vanilla JS.

How to Include

To use Bootstrap, first we create a HTML page that correctly defines the **<!doctype html>** and has the **viewport meta** tag.

Now we need to link the Bootstrap CSS file.

This can be done using a CDN link or by downloading the Bootstrap files.



How to Include

Online linking:

```
<link  
  rel="stylesheet"  
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  
  crossorigin="anonymous">
```

How to Include

Bootstrap is made of different components.

We can clearly see this in the downloaded version:
































This allows us to only include what we need from Bootstrap.

The **bootstrap.min.css** file includes everything and is minified.

Minified files (CSS or JS) have all spacing removed to minimize the size.

This makes them less readable and only useful for production use.

The **bootstrap.css** file is the actual written code.

Name	Size
 bootstrap-grid.css	70 329
 bootstrap-grid.css.map	203 221
 bootstrap-grid.min.css	51 795
 bootstrap-grid.min.css.map	115 986
 bootstrap-grid.rtl.css	70 403
 bootstrap-grid.rtl.css.map	203 225
 bootstrap-grid.rtl.min.css	51 870
 bootstrap-grid.rtl.min.css.map	116 063
 bootstrap-reboot.css	12 065
 bootstrap-reboot.css.map	129 371
 bootstrap-reboot.min.css	10 126
 bootstrap-reboot.min.css.map	51 369
 bootstrap-reboot.rtl.css	12 058
 bootstrap-reboot.rtl.css.map	129 386
 bootstrap-reboot.rtl.min.css	10 198
 bootstrap-reboot.rtl.min.css.map	63 943
 bootstrap-utilities.css	107 823
 bootstrap-utilities.css.map	267 535
 bootstrap-utilities.min.css	85 352
 bootstrap-utilities.min.css.map	180 381
 bootstrap-utilities.rtl.css	107 691
 bootstrap-utilities.rtl.css.map	267 476
 bootstrap-utilities.rtl.min.css	85 281
 bootstrap-utilities.rtl.min.css.map	180 217
 bootstrap.css	281 046
 bootstrap.css.map	679 755
 bootstrap.min.css	232 803
 bootstrap.min.css.map	589 892
 bootstrap.rtl.css	280 259
 bootstrap.rtl.css.map	679 615
 bootstrap.rtl.min.css	232 911
bootstrap.rtl.min.css.map	589 087

How to Include

To use some Bootstrap components that require JavaScript, the following should be included:

```
<script  
  src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"  
  crossorigin="anonymous">  
  
<script  
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"  
  crossorigin="anonymous">
```

<script> elements are included at the end of the <body> element.
Popper.js is required for dropdowns, popovers, and tooltips.
We can also use bootstrap.bundle.min.js which includes Popper.

How to Include

```
1  <!doctype html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Bootstrap</title>
8    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9          rel="stylesheet" crossorigin="anonymous">
10 </head>
11
12 <body>
13   <h1>Hello World!</h1>
14   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
15         crossorigin="anonymous"></script>
16 </body>
17
18 </html>
```

Reboot

Reboot is a Bootstrap component that does normalization. The default box-sizing of all elements is set to border-box.

```
*, *::before, *::after {  
  box-sizing: border-box;  
}
```

Breakpoints

Bootstrap (since version 3) is a mobile first design framework. This means that we design for mobile first and apply changes for desktop screens. The breakpoints are specified using the **min-width** property. Bootstrap defines 6 breakpoints:

| Breakpoint | Class infix | Dimensions |
|-------------------|-------------|------------|
| Extra small | <i>None</i> | <576px |
| Small | <i>sm</i> | ≥576px |
| Medium | <i>md</i> | ≥768px |
| Large | <i>lg</i> | ≥992px |
| Extra large | <i>xl</i> | ≥1200px |
| Extra extra large | <i>xxl</i> | ≥1400px |

Containers

There are some classes for fluid and fixed width containers.

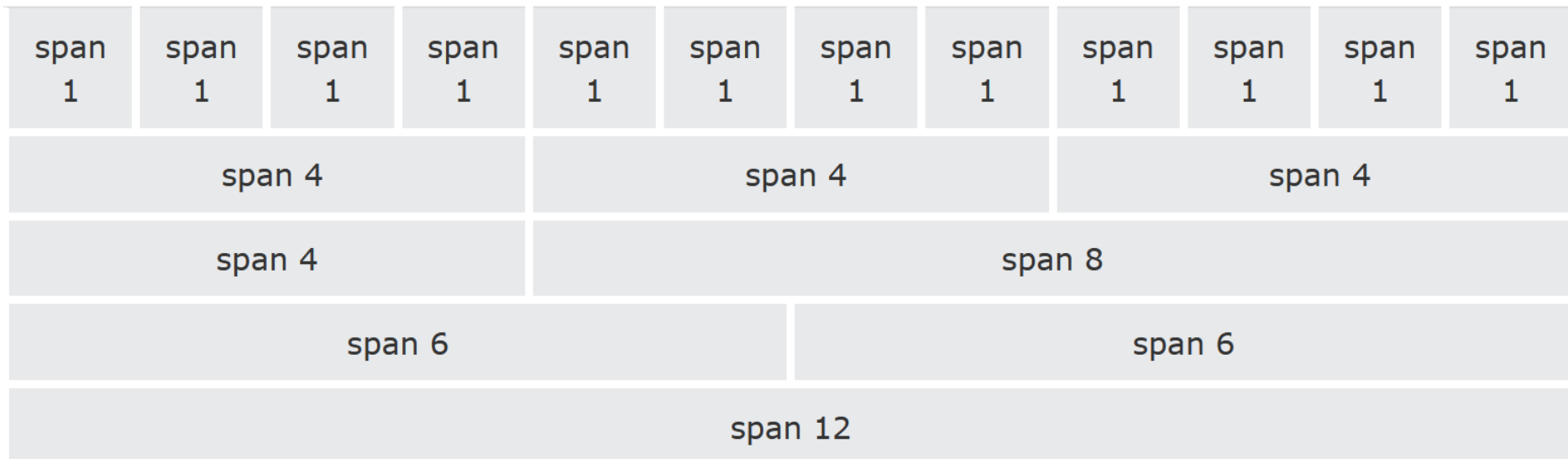
.container is the complete fixed width container which changes size at each breakpoint.

.container-fluid is a fluid container always taking 100% width.

| | Extra small
<576px | Small
≥576px | Medium
≥768px | Large
≥992px | X-Large
≥1200px | XX-Large
≥1400px |
|-------------------------------|-----------------------|-----------------|------------------|-----------------|--------------------|---------------------|
| <code>.container</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-sm</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-md</code> | 100% | 100% | 720px | 960px | 1140px | 1320px |
| <code>.container-lg</code> | 100% | 100% | 100% | 960px | 1140px | 1320px |
| <code>.container-xl</code> | 100% | 100% | 100% | 100% | 1140px | 1320px |
| <code>.container-xxl</code> | 100% | 100% | 100% | 100% | 100% | 1320px |
| <code>.container-fluid</code> | 100% | 100% | 100% | 100% | 100% | 100% |

Grid

Bootstrap uses a grid layout to place our items in it. (The grid is actually built using flexbox)
The grid is a **12 column** system which supports all 6 breakpoints.
The grid system should be used inside of a container (fixed-width or fluid).



Grid

There is a **row** and a **col** class.

The row class is a wrapper for columns which allow applying sizing and gaps for all columns.
The columns are equally distributed based on how many there are.

```
<div class="container">
  <div class="row">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```



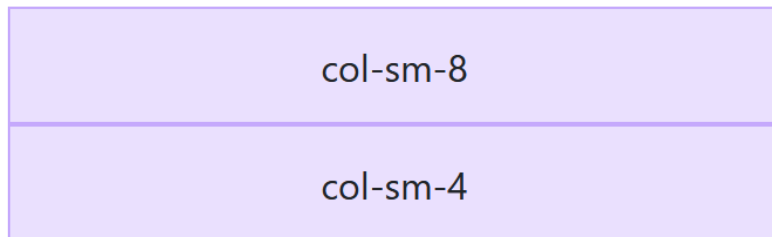
Grid

We can use **col-n** (such as col-2 or col-6) to make a column bigger.



We can also use **col-<breakpoint>-n** (such as col-md-2) to change layout responsively.

Here, col-sm-8 means that col-8 is only applied when screen is larger than sm (576px):



Grid

We can use the **row-cols-n** class on the **row** container as opposed to **col-n** on individual columns:

```
<div class="container">
  <div class="row row-cols-2">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column
Column	Column

Grid Gutters

Gutters or gaps can be applied between columns using the **g-n** (g-0 to g-5) classes. We can also only apply gutters on one axis (gx-1 or gy-5).

```
<div class="container">  
  <div class="row g-4">  
    ...  
  </div>  
</div>
```

Padding & Margin

Padding and margin are also applied to elements using the same syntax as gutters.

We have p-n, px-n, py-n, ps-n, pe-n, pt-n, pb-n.

The same applies for margin: m-n, mx-n, my-n, ms-n, me-n, pt-n, pb-n.

```
<p class="px-2 mt-1">Test</p>
```

p-n: all 4 sides

px: left and right, py: top and bottom

ps: left (start), pe: right (end)

pt: top, pb: bottom

```
/* Time to check some code! */
```

Thank you for your attention.