Responsive Web Design

Software Technology Engineering - 1. Semester

VIA University College

Today’s topics

- Presentation of the teacher
- Some practical things
- What is HTML?
- HTML lists
- HTML links
- HTML images

Presentation of teacher

- A few things about the teacher

Consequences

- If you come to late – you miss some of the lesson
  - You need 75% attendance
  - Wait outside class until next break!
- Assignments are expected to be handed in on time

Consequences

- If these requirements are ***not*** met, the student is not allowed to attend the ordinary exam and misses one of the three attempts

Text books

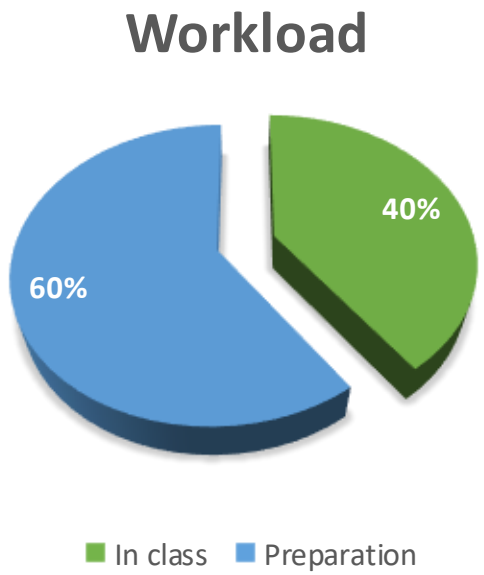
- **[Duckett, 2011]:** HTML & CSS Design and Build Websites, ISBN: 978-1-118-00818-8
- **[LaGrone, 2013]** : HTML5 and CSS3 Responsive Web Design Cookbook, ISBN: 978-1-84969-544-2
- **[Duckett, 2014]**: Javascript & jQuery, ISBN: 978-1-118-53164-8

- Tutorials on www.w3schools.com

Expected workload

- The course counts 5 ETCS point
- One ETCS point = 27.5 hours work => total 137 hours
- We are in the class 4 lessons per week distributed over 12 weeks (48 hours in total)

- For preparation/homework: 137-48 = **89 hours!**



Conclusion:

Preparation/homework: 7.4 hours per week!!!!

Software development is learned by doing!

Compulsory assignments

- Three assignments
  1. HTML5 & CSS3 based Website
  2. Make the Website responsive (suitable for mobile devices)
  3. Using jQuery and JavaScript

- All must be handed in on time to attend the exam.
- Assignment 1 & 2 are part of SEP1.

Structure of the lectures

1. Questions about the exercises from last time.
2. Presentation of new material.
3. Exercises.
4. Learning path on itslearning (including exercises).

- You are expected to complete ***everything*** before the next session.
- If you don’t finish in class, then it’s **homework**.-

Evaluation of the course

Digital written exam

1) Multiple choice questions covering the topics of the course
2) Short answer questions
   - Write code
   - Explain code

The duration of the digital written exam is 2 hours.

To attend the exam: 75% attendance and course assignments **MUST** be handed in on time and approved.

Students will be graded according to the 7-point grading scale

What is HTML?

- Hyper Text Markup Language (HTML).
- Markup's tells how the content of a webpage should be presented through the use of standardized tags and codes.
- Standards from World Wide WEB Consortium (W3C) – available on www.w3.org
- The "recipe" language for webpages.

HTML document

- A plain text file (ASCII)
- All editors can be used to edit HTML
  - We use Visual Studio Code 
- File extension **must** be *.html*
- The “front page” of any website is called *index.html*
- A HTML file has a minimum contents that must be there
  - More about this later

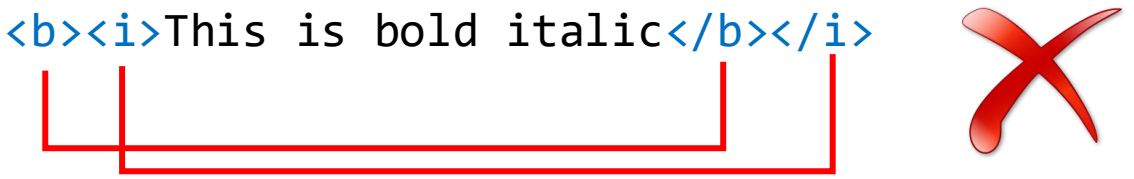
Principles of HTML

Markup

- *Start tags*: `<h1>`, `<p>`, `<ul>`, `<b>`, `<i>`
- *End tags:* `</h1>`, `</p>`, `</ul>`, `</b>`, `</i>`

Tags are nested:

Not like this:



- Empty/open tags: `<br>`, `<hr>`

Work this way

1. Make a draft in plain text
2. Find the logical elements in the text
3. Mark up the text using HTML elements

The HTML

```
<h1>RWD I1, Session plan, Autumn 2021</h1>
<h2>Session 1</h2>
<p>Week 36, Monday 06-09-2021</p>
<h3>Main topics</h3>
<ul>
    <li>Introduction to the course</li>
    <li>Introduction to HTML</li>
</ul>
```

The HTML (continued)

```
<h3>Exercises</h3>
<ol>
    <li>Download "Star Pizza.txt" and mark up the text.</li>
    <li>Create the HTML page for Star Pizza.</li>
    <li>Create a personal web page: CV, Hobbies, Home Country/Town etc.</li>
    <li>Go to www.w3schools.com and take the HTML tutorial.</li>
</ol>
```

Work this way - comments

```
<!-- This is a comment for the paragraph below -->
<p> This text wrapped as a paragraph </p>


<!-- These lines of paragraph are not being instanced by
the browser because they have been "out-commented"
<p> Paragraph line 1 </p>
<p> Paragraph line 2 </p>
-->
```

Always add comments to your code!

Minimum HTML File Contents

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="utf-8">
      <title>Title of the document</title>
</head>
<body>
      <p>Your text goes here</p>
</body>
</html>
```

HTML lists

`<li>…</li>` Items in a list are surrounded by

`<ul>…</ul>` The unordered list (*will give you bullet points*)

`<ol>…</ol>` The ordered list (*will give you numbers*)

The definition/description list (*will not give you an indentation*)

```
<dl>
   <dt>…</dt>
   <dd>…</dd>
</dl>
```

HTML lists

HTML

Unordered list:

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

Ordered list:

```
<ol>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ol>
```

In Browser

Unordered list:

- Item 1
- Item 2
- Item 3

Ordered list:

1. Item 1
2. Item 2
3. Item 3

HTML lists

Description list:

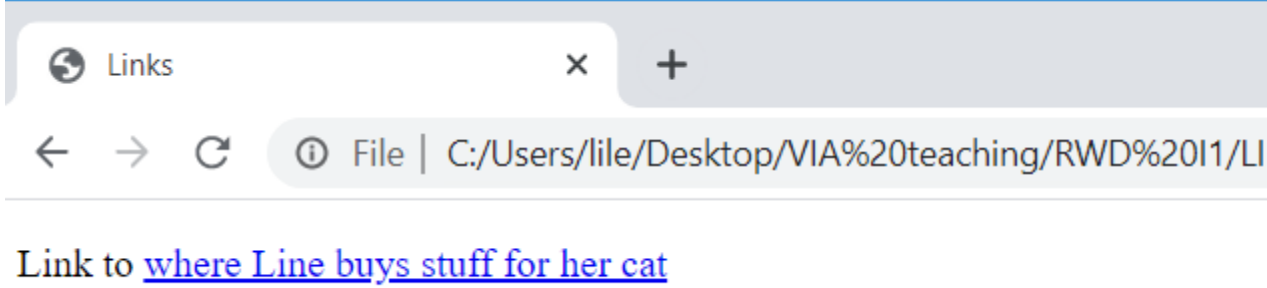
HTML

```
<dl>
    <dt>RWD I1</dt>
        <dd>Responsive WEB Design course</dd>
    <dt>LRL I1</dt>
        <dd>LEGO Robot Lab course</dd>
</dl>
```

In Browser

RWD I1
    Responsive WEB Design course
LRL I1
    LEGO Robot Lab course

HTML links



<a> This is a link tag (anchor)

Hypertext reference

The page the link takes you to

```
<p>Link to <a href="http://www.zooplus.dk">
 where Line buys stuff for her cat</a></p>
```

The text the user will see, and can click on

Open page in new Browser window

```
<a href="http://www.zooplus.dk" target="_blank">
Where Line buys stuff for her cat</a>
```

HTML images

```
<img src="images/sofus.jpg" alt="Picture of my cat Sofus"
title="RWD I1 lecturer Line’s cat Sofus"/>
```

src: URL to the picture/image to show

alt: Description of the picture – shown if the picture can’t be seen
Can be read by screen readers

title: Description of the picture
Often shown as tool tip when hovering
over the picture

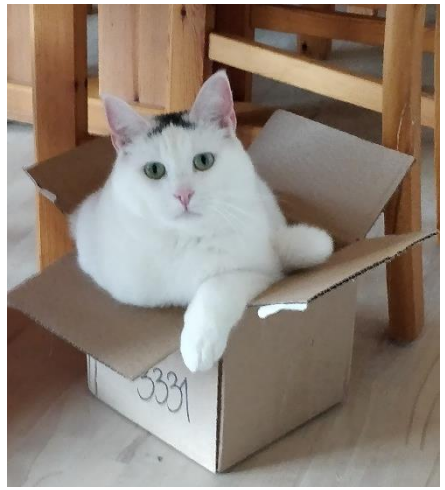


Image links

- To create an image link use a link element to contain an image element (instead of the link text)

```
<a href="index.html"> <img src="home.gif"
height="19" width="85" alt="Home"/></a>
```

- Many browsers automatically add a border to image links.
- Later we will handle this with CSS

Thumbnail Image

A small image configured to link to a larger version of that image

```
<a href="big.jpg"><img src="small.jpg" alt="country road"
width="200" height="100"/></a>
```

Exercises and itslearning learning path

- Find the exercises on itslearning (zip file containing an HTML file and an images folder)
  - Open it in Visual Studio Code to see the HTML structure and content
  - See the result in the browser window
  - Go step by step and make your Star Pizza homepage
  - Just ask me if you get stuck ☺
- When you finish the exercises, go through the learning path on itslearning

Responsive Web Design

Software Technology Engineering - 1. Semester

VIA University College

Today’s topics

- Pop quiz?
- Introduction to CSS
- Types of styling
- CSS syntax
- class and id selectors
- and elements
- The CSS box

Pop Quiz

What is CSS?

- CSS is an acronym for **C**ascading **S**tyle **S**heets
- CSS is a style language that defines layout of HTML documents
- CSS offers more options and is more accurate and sophisticated than inline styling in HTML
- ***HTML is used to structure content***
- ***CSS is used for formatting structured content***

Types of styling

- Inline Styles
  - Configured in the body of the Web page
  - Use the style attribute of an HTML tag
  - Apply only to the specific element’
- Embedded Styles
  - Configured in the head section of a Web page.
  - Use the HTML <style> element
  - Apply to the entire Web page document
- External Styles/Imported styles
  - Configured in a separate text file with .css file extension
  - The HTML <link /> element in the head section of a Web page associates it with the .css file

Inline styles

```
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

An inline CSS is used to apply a unique style to a single HTML element.

Embedded and external styles sheets

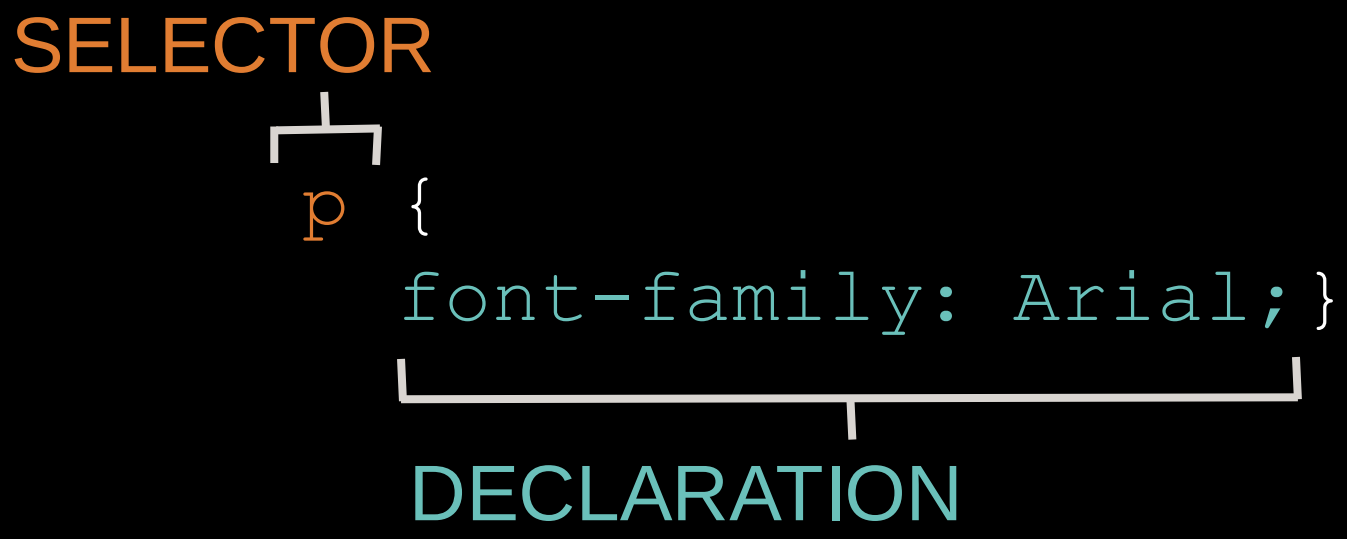
- All layout and formatting are separated from contents
  - The HTML files are exactly the same
  - No need to copy style code from elements
  - Changes to CSS automatically apply to the entire website

CSS associates style rules with HTML elements

```
p {
   font-family: Arial;}
```

CSS associates style rules with HTML elements

CSS associates style rules with HTML elements



CSS properties affect how elements are displayed

```
h1, h2, h3 {
            font-family: Arial;
            color: yellow;}
```

CSS properties affect how elements are displayed

```
h1, h2, h3 {
        font-family: Arial;
        color: yellow;}
```

PROPERTY

CSS properties affect how elements are displayed

```
h1, h2, h3 {
        font-family: Arial;
        color: yellow;}
```

PROPERTY VALUE

CSS properties

The CSS font-family property defines the font to be used.

```
body {font-family: Georgia, Times, serif;}
```

The CSS color property defines the text color to be used.

```
h1 {color: DarkCyan;}
```

The CSS font-size property defines the text size to be used.

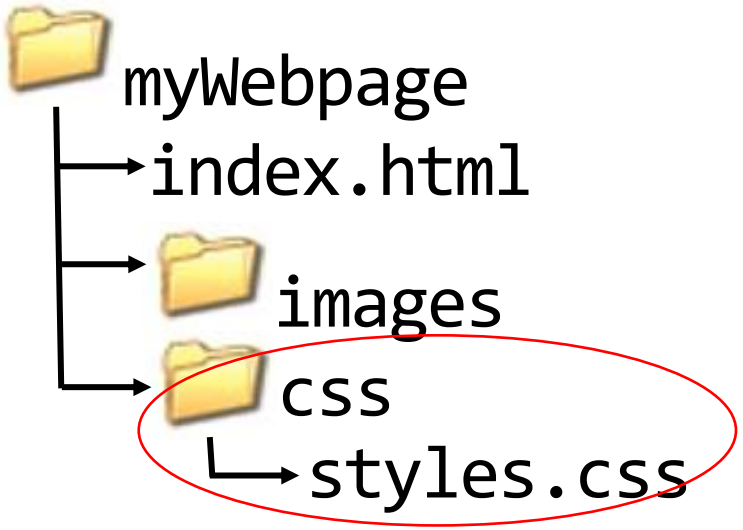
```
h1 {font-size: 200%;}
```

Embedded styles

```
<head>
  <title>Some page title</title>
  <style type="text/css">
    body {
      background-image: url("background.jpg");
      color: #FFFFFF;
    }
    h1 {
      background-color: #cccccc;
      color: #FF0000;
    }
  </style>
</head>
<body>
...
```

External styles

- Separate file(s) with the .css extension
- Often placed in dedicated subdirectory




```
<head>
<title>Some page title</title>
<link href="css/styles.css" type="text/css" rel="stylesheet" />
</head>
<body>
...
```

Always the same

Inside the .css file

```
* {
  font-size: 1.2em;
}

body {
  background-image: url("background.jpg");
  color: #FFFFFF;
}
h1 {
  background-color: #cccccc;
  color: #FF0000;
}
```

Cascading in the CSS file

```
* {font-family: Arial, sans-serif;}

h1 {font-family: ”Courier New”, monospace;}

li {color: violet;}

p {color: green;}

p#myId {color: red;}

p.myClass {color: pink;}

li {color: blue !important;}

li {color: gray;}

li,p {font-size: 75%;}
```

CSS selectors

`* {…}`	Style applies to all elements.
`body {…}`	Style applies to all elements within body.
`div {…}`	Style applies to all divisions and elements within.
`.myClass {…}`	Style applies to all elements of with the given class name.
`#myId {…}`	Style applies to the element with the unique id.

class and id selectors

- **class**
  - Applies a CSS rule to a certain "class" of elements on a Webpage.
  - Does not associate the style to a particular HTML element.
  - Configure with `.classname` in CSS file.

- **id**
  - Applies a CSS rule to **ONE** element on a Webpage
  - Configure with `#idName` in CSS file
  - The most specific CSS selector!

class example

```
<body>
  <div>
  <p>This is an ordinary paragraph</p>
  <p class="myClass">This is a paragraph with the myClass style</p>
  </div>
</body>
```

```
div {
 height: 300px;
 width: 400px;
 background-color: #EE3E80;
}

.myClass {
  background-color: #CCCCCC;
}

p.myClass {
  color: #FF0000;
}
```

id selector example

```
<body>
    <p>This is an ordinary paragraph</p>
    <p id="myId">This is a paragraph with a different style</p>
</body>
```

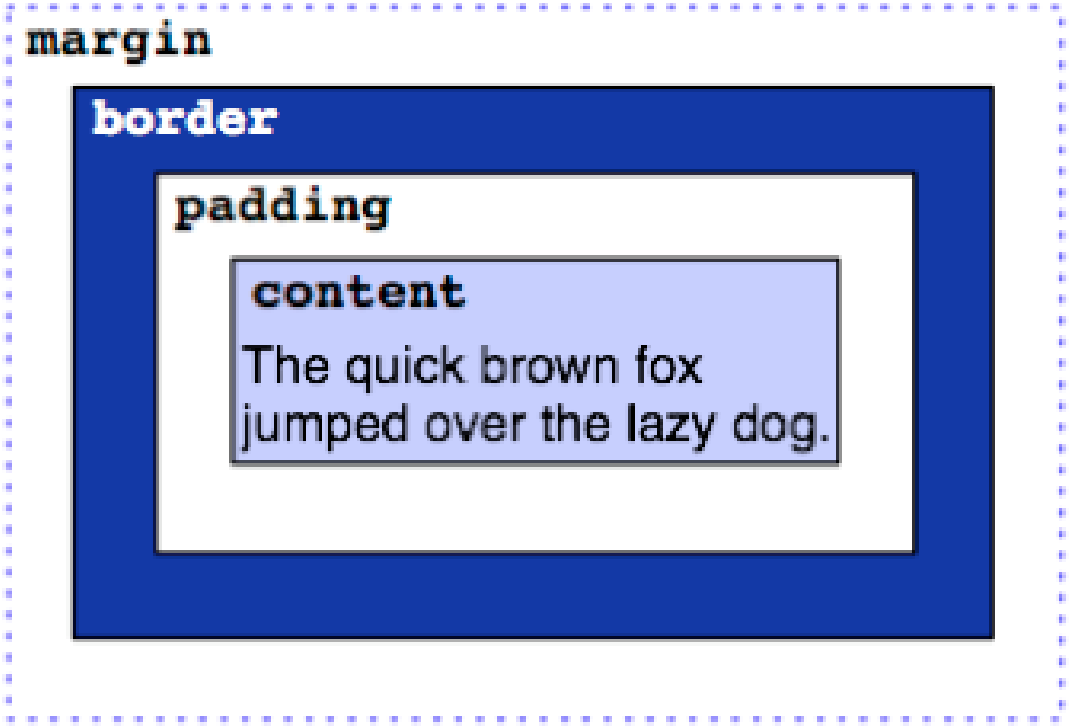
```
#myId {
  color: #0000FF;
  background-color: lemonchiffon;
  width: 100px;
  height: 100px;
}
```

and elements/selectors

- What if there's no "natural" tag (<p>, <h1>…)?
- You can use or
- is for block/paragraph-like selections
- is for parts of text like …

- Examples:
  - `<div id="myId">All Inside a division.</div>`
  - `This is only<span class="myClass">a part of the text</span>`

The CSS box



Configure empty space around the content

```
padding: 20px 80px 10px 40px
          top  right  bottom left
```

Configure visible borders around the element

```
border:  10px 5px 10px 200px
          top  right bottom  left
```

Configure transparent space around the border

```
margin:  10px 5px 10px 200px
          top  right bottom  left
```

Exercises and itslearning learning path

- Find the exercises on itslearning
  - Open it in Visual Studio Code to see the HTML structure and content
  - See the result in the browser window
  - Go step by step and make your Star Pizza homepage
  - Just ask me if you get stuck ☺

- When you finish the exercises, go through the learning path on itslearning

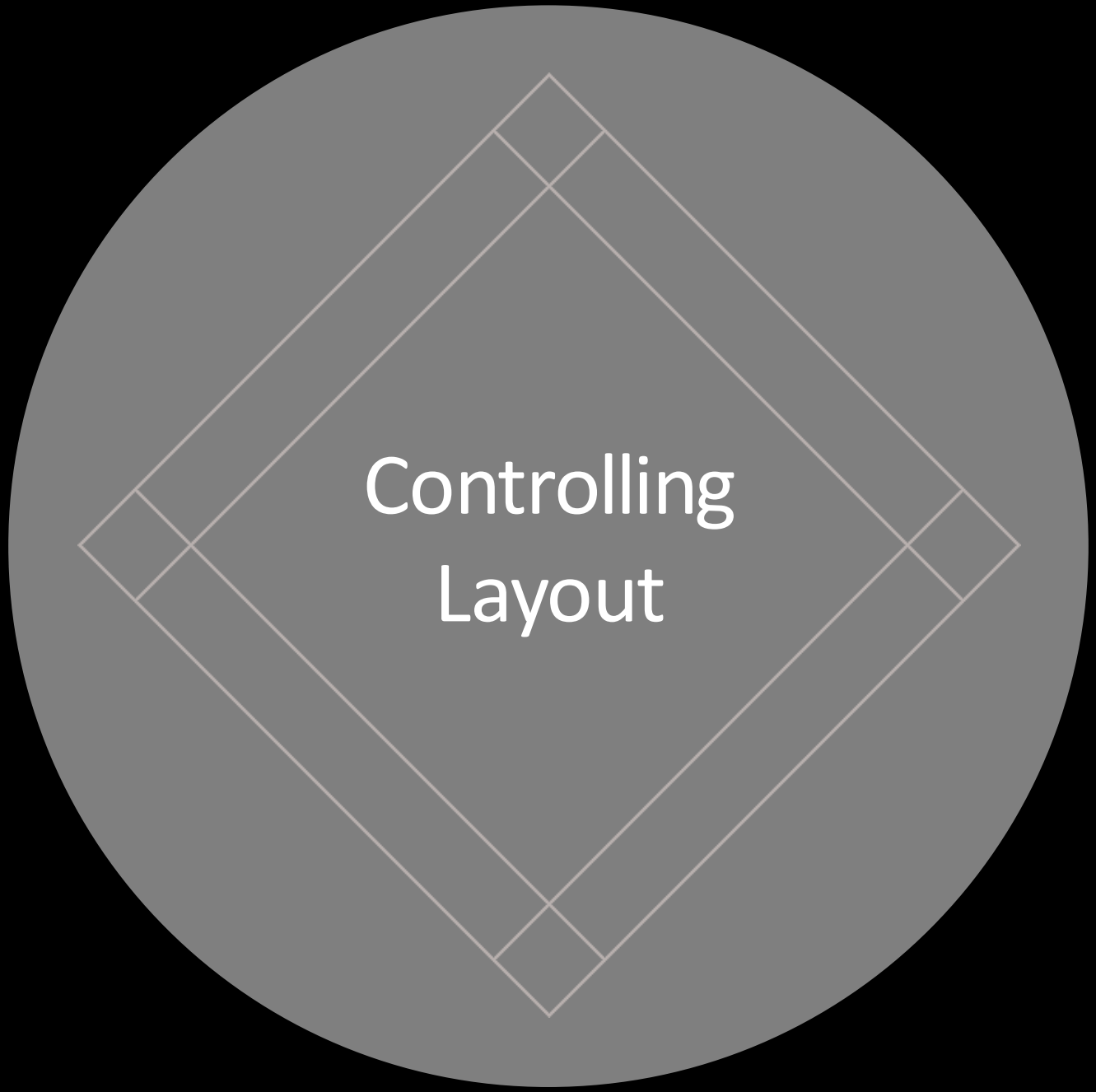
Responsive Web Design

Software Technology Engineering - 1. Semester

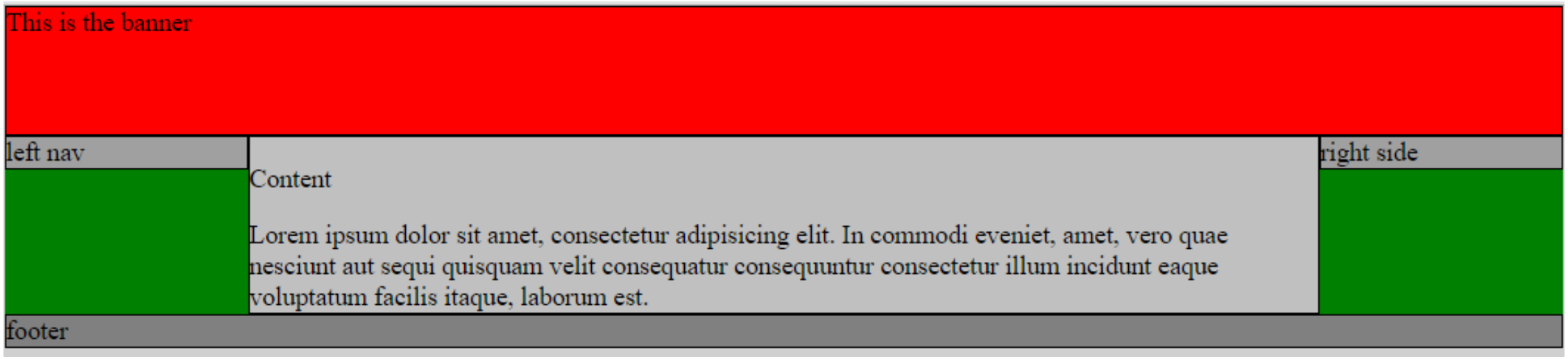
VIA University College

Today’s topics (session 3)

- Controlling layout
- The position property
- The float property
- The clear property
- HTML tables
- The nth-child() selector


Controlling
Layout

for different sections of your layout



HTML

```
<div id= "wrapper">
    <div id="banner">This is the banner</div>
    <div id="left_nav">left nav</div>
    <div id="content">
        <p>Content</p>
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. In commodi eveniet,
            amet, vero quae nesciunt aut sequi quisquam velit consequatur  consequuntur
            consectetur illum incidunt eaque voluptatum facilis itaque, laborum est.
    </div>
    <div id="right_side">right side </div>
    <div id="footer">footer</div>
</div>
```

CSS

```
* {
    box-sizing: border-box;
}
#wrapper {
    width: 960px;
    margin-left: auto;
    margin-right: auto;
    background-color: green;
}

#banner {
    height: 80px;
    background-color: red;
    border: 1px solid black;
}
```

```
#left_nav {
    background-color: #A0A0A0;
    border: 1px solid black;
    width: 150px;
    float: left;
}

#content{
    border: 1px solid black;
    background-color: #C0C0C0;
    width: 660px;
    float: left;
}
```

```
#right_side{
    border: 1px solid black;
    background-color: #A0A0A0;
    width: 150px;
    float: left;
}

#footer{
    clear: both;
    border: 1px solid black;
    background-color: #808080;
}
```

The position property

Value	Description
static	Default value. Elements render in order, as they appear in the HTML document flow.
absolute	The element is positioned relative to its first positioned (not static) parent element.
fixed	The element is positioned relative to the browser window.
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position.
sticky *(relatively new positioning value)*	The element is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position.

Box offset properties

```
top: 100px;
bottom: 100px;
left: 100px;
right: 100px;
```

http://www.w3schools.com/cssref/pr_class_position.asp

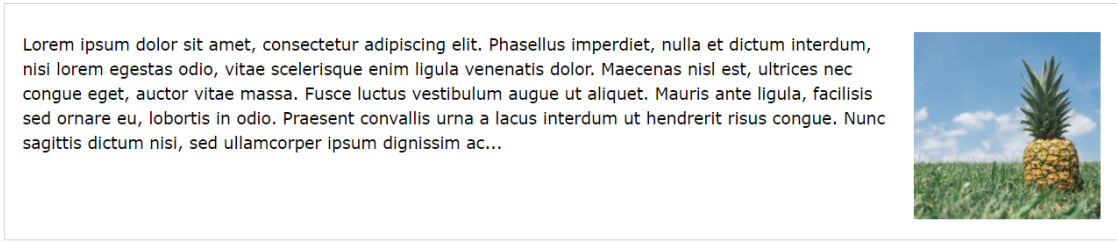
The float property

- The CSS float property specifies how an element should float.

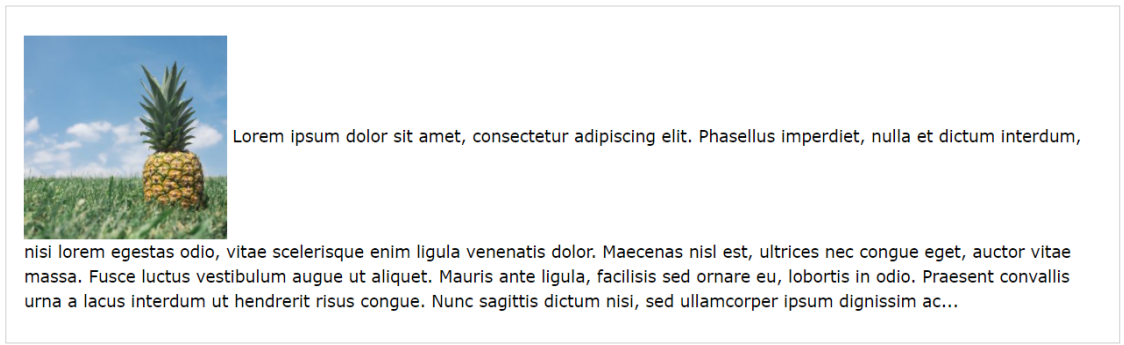
HTML

```
<p><img src="pineapple.jpg" alt="Pineapple" style=
"width:170px;height:170px;"/>
Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Phasellus imperdiet,…</p>
```

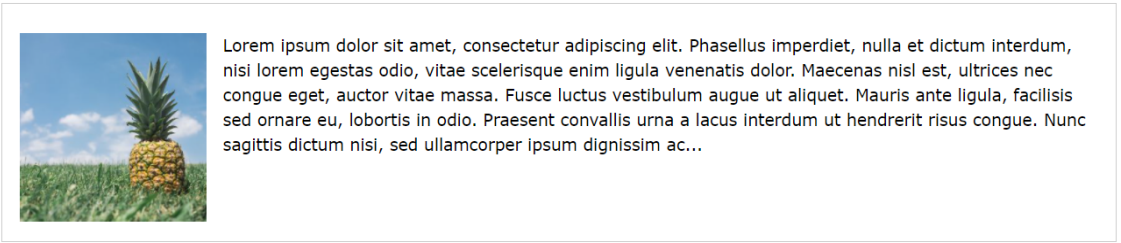
img {float: right;}



img {float: none;} *(default)*



img {float: left;}



Floating next to each other

- Normally elements will be displayed on top of each other. However, if we use float: left we can let elements float next to each other

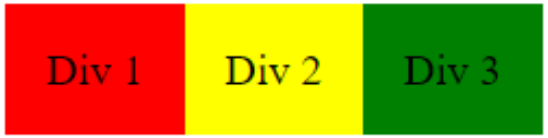
HTML:

```
<div class="div1">Div 1</div>

<div class="div2">Div 2</div>

<div class="div3">Div 3</div>
```

CSS:

```
div {
  float: left;
  padding: 15px;
}

.div1 {
  background-color: red;
}

.div2 {
  background-color: yellow;
}

.div3 {
  background-color: green;
}
```

Result:



https://www.w3schools.com/css/css_float.asp

The clear property

- When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.
- The clear property specifies what should happen with the element that is next to a floating element.

Value	Description
none	The element is not pushed below left or right floated elements. This is default
left	The element is pushed below left floated elements
right	The element is pushed below right floated elements
both	The element is pushed below both left and right floated elements
inherit	The element inherits the clear value from its parent

The clearfix Hack

- If a floated element is taller than the containing element, it will "overflow" outside of its container.
- We can then add a clearfix hack to solve this problem

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

HTML:

```
<div class= "clearfix" >
  <img class ="img2" src="pineapple.jpg" alt="Pineapple"
width="170" height="170"/>
  Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Phasellus imperdiet...
</div>
```

CSS:

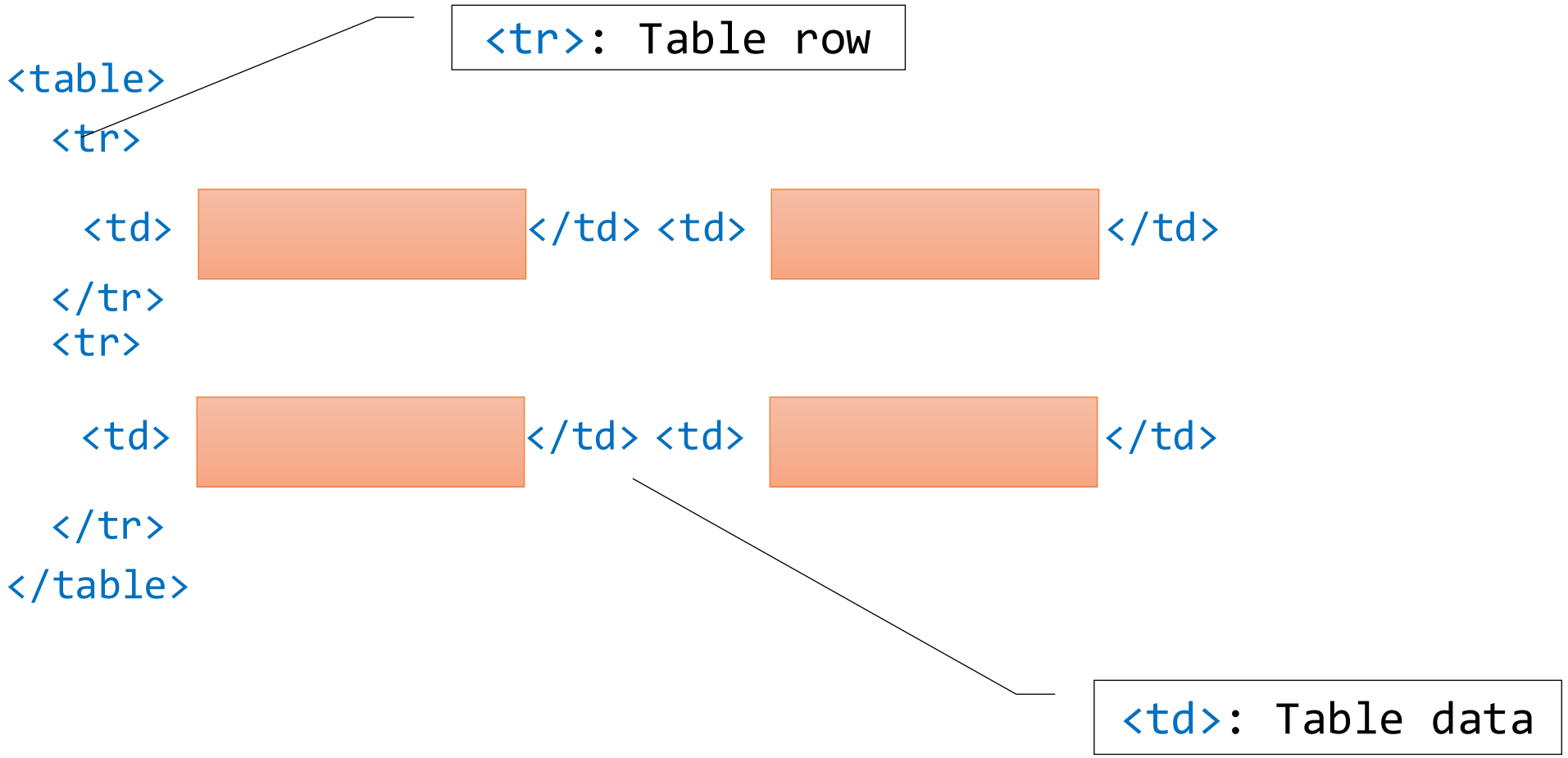
```
.clearfix {
  overflow: auto;
}
```

Tables in HTML

- Arranges items in tables
- Can have borders
- ~~Can be used to make the structural layout the page~~ – **Not recommended**

Qty	Unit	Item
2	liter	Milk
5	boxes	Cake
3	pcs	Washer

Table Tags


<tr>: Table row
<table>
<tr>
<td>
</td> <td>
</td>
</tr>
<tr>
<td>
</td> <td>
</td>
</tr>
</table>
<td>: Table data

The colspan attribute

```
<table>
  <tr>
    <td>  </td> <td>  </td>
  </tr>
  <tr>
    <td colspan="2">  </td>
  </tr>
</table>
```

The rowspan attribute

```
<table>
  <tr>
    <td rowspan="2">
  </tr>
  <tr>


  </tr>
</table>
```

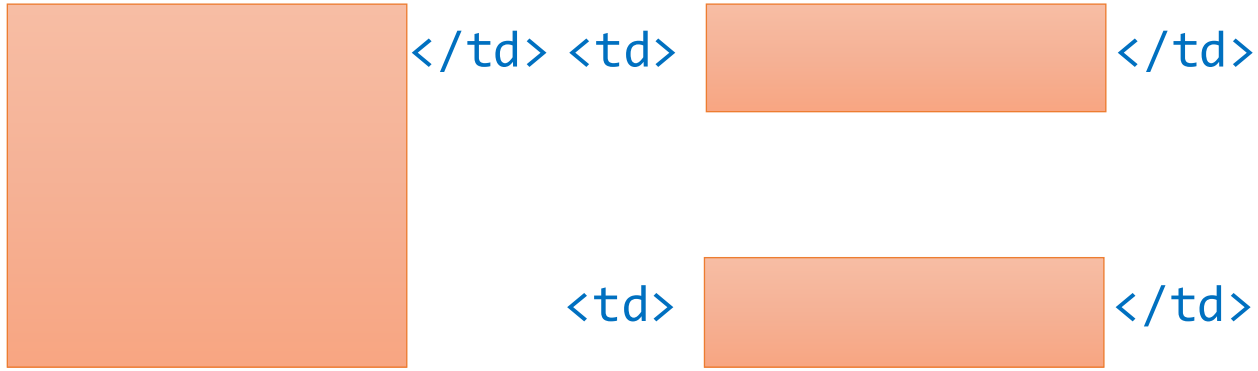


Table tags <th>

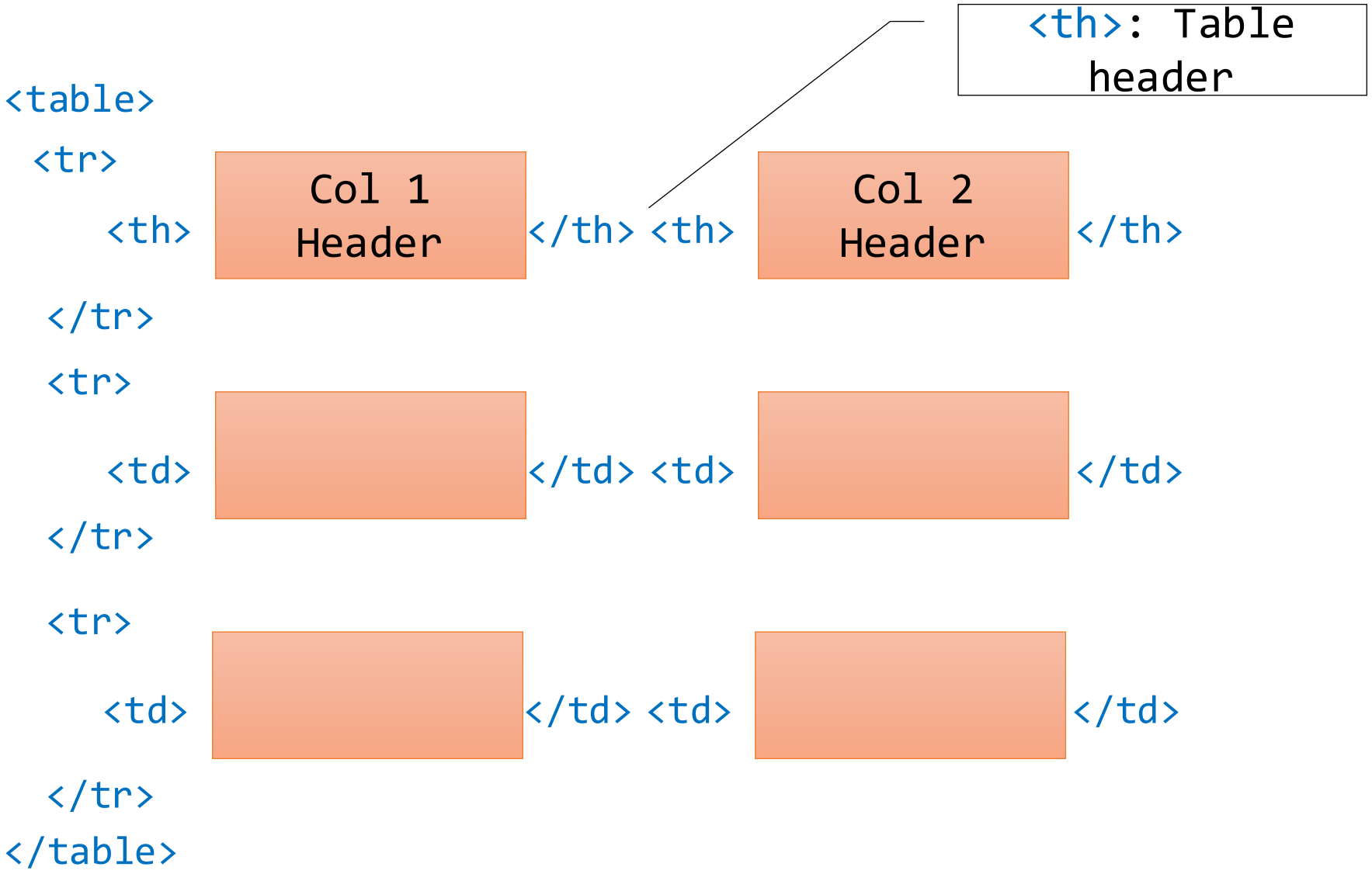


Table Tags – <th>

	Col 1 Header	Col 2 Header
Row 1 Header	Content 1	Content 2
	Content 3	Content 4

```
<table>
  <tr>
    <th></th>
    <th>  Col 1 Header  </th>
    <th>  Col 2 Header  </th>
  </tr>
  <tr>
    <th>  Row 1 Header  </th>
    <td>  Content 1  </td>
    <td>  Content 2  </td>
  </tr>
  <tr>…</tr>
</table>
```

Table Example

```
<table>
  <tr>
    <th>Qty</th>
    <th>Unit</th>
    <th>Item</th>
  </tr>
  <tr>
    <td>2</td>
    <td>liter</td>
    <td>Milk</td>
  </tr>
  <tr>
    <td>5</td>
    <td>boxes</td>
    <td>Cake</td>
  </tr>
</table>
```

Qty	Unit	Item
2	liter	Milk
5	boxes	Cake

Adding Borders

```
table,th,td {
    border-width: 2px;
    border-color: black;
    border-style: solid;
    border-collapse: collapse;
}
```

Qty	Unit	Item
2	liter	Milk
5	boxes	Cake

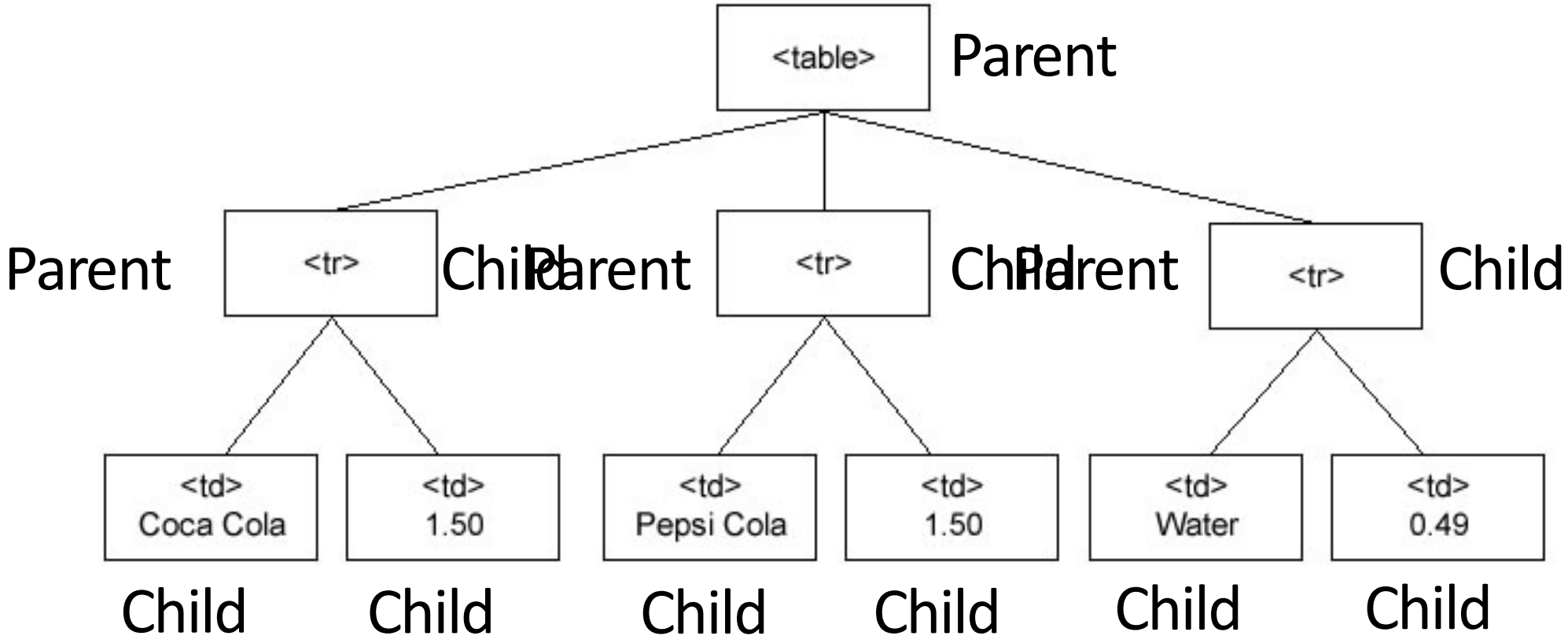
Qty	Unit	Item
2	liter	Milk
5	boxes	Cake

Additional CSS Magic

```
tr:nth-child(even) {
        background-color: #BBBBBB;
}
tr:nth-child(odd) {
        background-color: #999999;
}

td:nth-child(1) {
        text-align: center;
}
```

Qty	Unit	Item
2	liter	Milk
5	boxes	Cake

nth-child()


<table>
Parent
Parent
<tr>
Child
Parent
<tr>
Child
Parent
<tr>
Child
<td>
Coca Cola
<td>
1.50
<td>
Pepsi Cola
<td>
1.50
<td>
Water
<td>
0.49
Child
Child
Child
Child
Child
Child

nth-child()

```
<table>
  <tr>
    <th>Qty</th>
    <th>Unit</th>
    <th>Item</th>
  </tr>
  <tr>
    <td>2</td>
    <td>liter</td>
    <td>Milk</td>
  </tr>
  <tr>
    <td>5</td>
    <td>boxes</td>
    <td>Cake</td>
  </tr>
</table>
```

Parent

1st tr child element of its parent element

2nd tr child element of its parent element

3rd tr child element of its parent element

Select and style only those <tr> elements which are the n'th child of their parent element (n is given by the argument).

```
(CSS)
tr:nth-child(even) {
        background-color: #BBBBBB;
}
tr:nth-child(odd) {
        background-color: #999999;
}
```

nth-child()

```
<table>
  <tr>
    <th>Qty</th>
    <th>Unit</th>
    <th>Item</th>
  </tr>
  <tr>  ← Parent
    <td>2</td> ← 1st td child element of its parent element
    <td>liter</td>
    <td>Milk</td>
  </tr>
  <tr>  ← Parent
    <td>5</td> ← 1st td child element of its parent element
    <td>boxes</td>
    <td>Cake</td>
  </tr>
</table>
```

Select and style only those <td> elements which are the n'th child of their parent element (n is given by the argument).

```
td:nth-child(1) {
        text-align: center;
}
```

Exercises and itslearning learning path

- Find the exercises on itslearning
  - Open it in Visual Studio Code to see the HTML structure and content
  - See the result in the browser window
  - Go step by step and make your Star Pizza homepage
  - Just ask me if you get stuck ☺

- When you finish the exercises, go through the learning path on itslearning

Responsive Web Design

VIA University College

Today’s Agenda

- Setting size for elements.
- Box-sizing.
- Media query and Breakpoints.
- Flexbox.
- The mobile device emulator in Chrome
- Exercises.

What is Responsive Web Desing?

- A technique that makes it possible to create webpages that can be shown with good overview and no need to scroll the page on different devices.
  - Screen size
  - Screen orientation
  - Media: Screen, TV, Print etc.

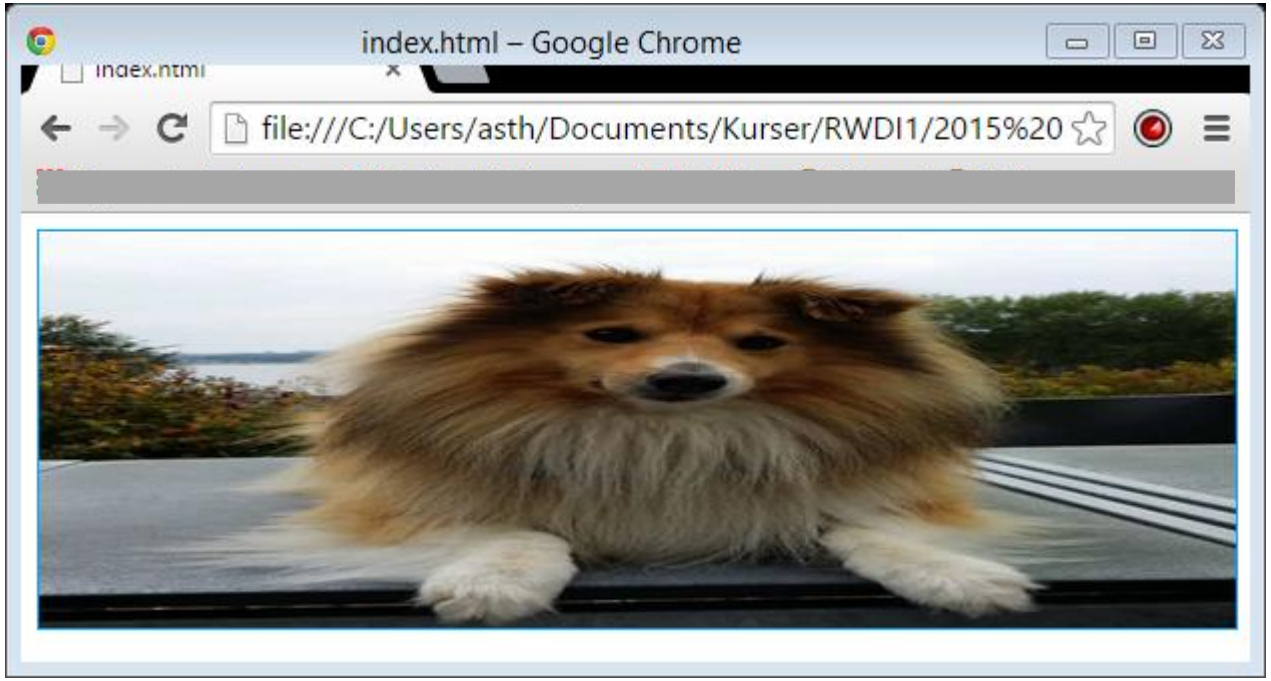
- Notice that the layout of menus and contents change depending on device.

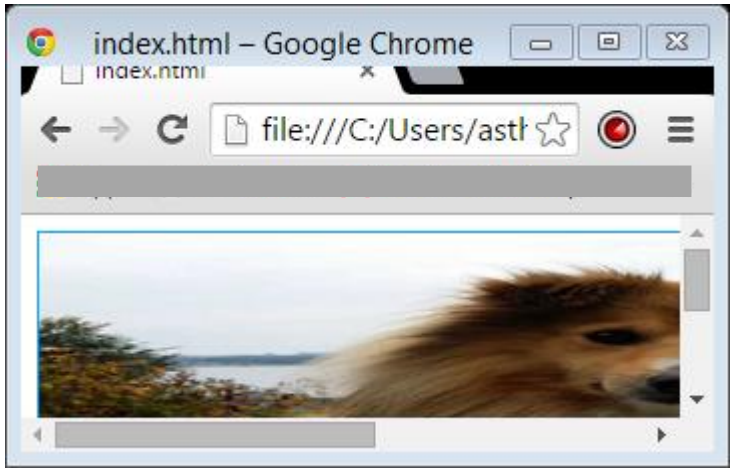
- The HTML code is the same; all layout changes are done in CSS.

Fixed Size Elements

```
img {
      width: 600px;
      height: 200px;
}
```

Element size is set regardless of the size of parent and children elements.

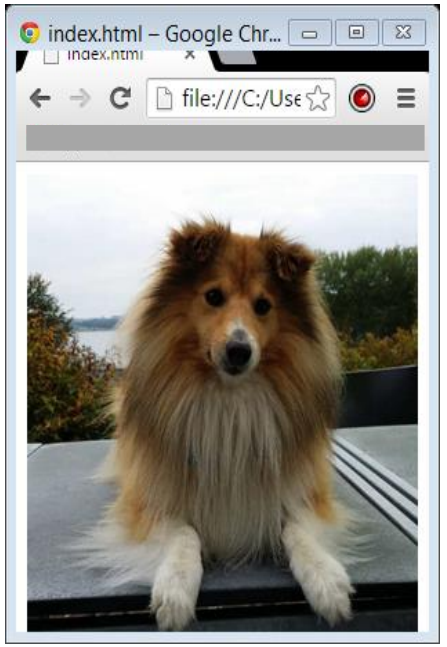




Relative Size

```
img {
      width: 100%;
      height: auto;
}
```

Element size is set relative to size of parent (in this case the browser window view/the body element).



Relative Size

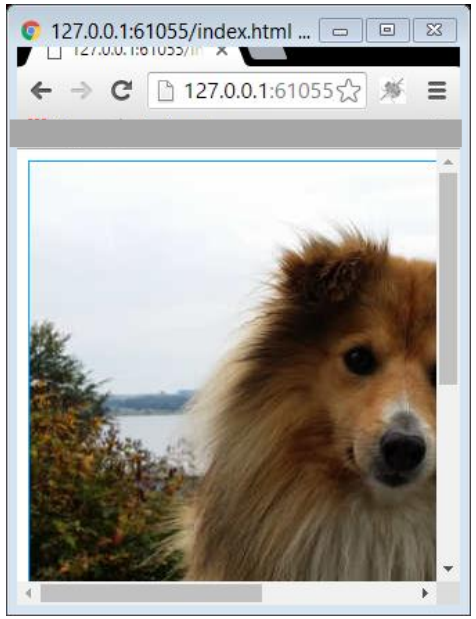
```
img {
        width: auto;
        height: 100%;
}
```

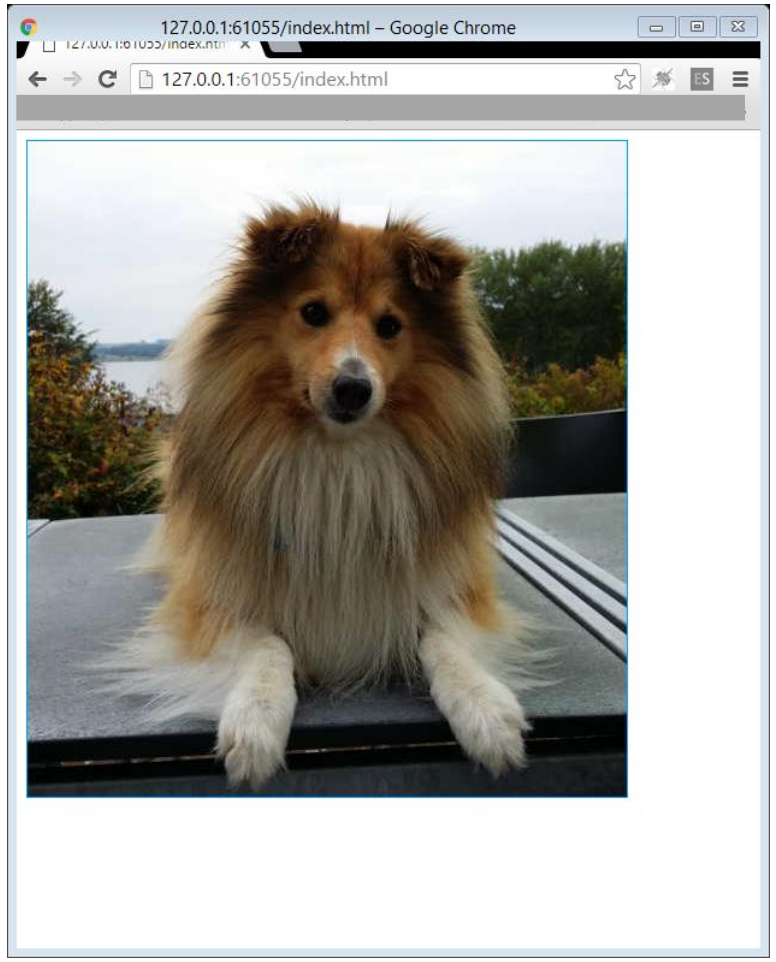
Does not work when the parent element is the browser view/the body element, but otherwise will.

What we see here is the default sizing.

To fix add:

```
html, body {
    height: 100%;
}
```



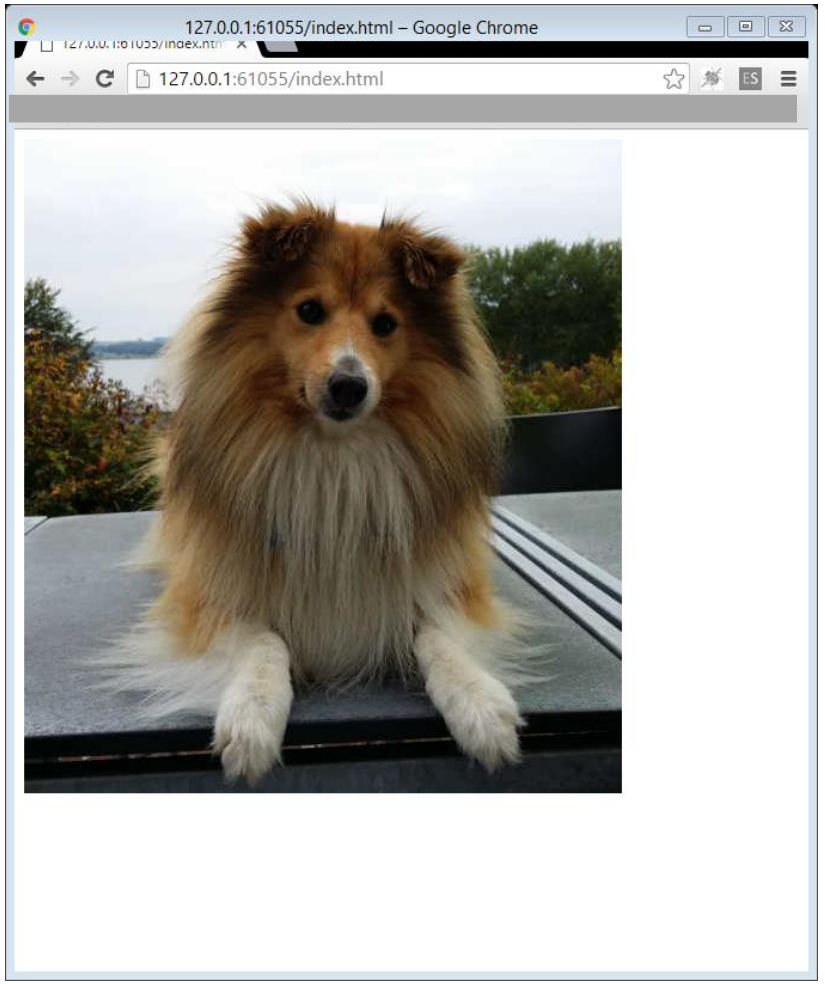


Relative Size

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

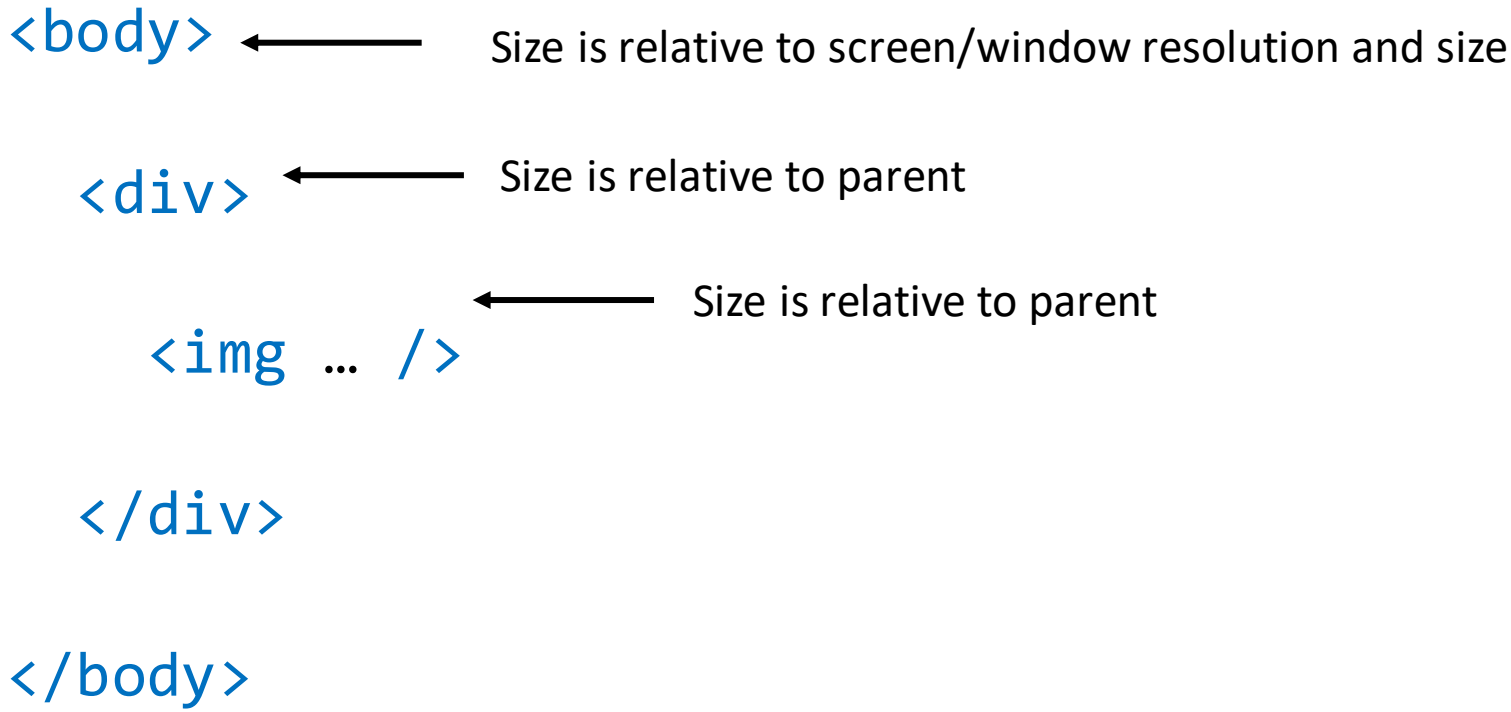
Element size is set relative to size of the child element (the original pixels size of the image) or downsized if the parent element is smaller.

This only works on images though.



Relative Size

Size is generally always inherited outward to inward.



Pixels

Pixel contra Pixel contra Pixel

Device pixels.

- Physical pixels on screen (given by the screen resolution).

CSS pixels.

- Element pixels.
- Is what the ‘`px`’ unit is referring to when used with properties in CSS.

Image pixels.

- The resolution of the image file.
- By default, one image pixel equals one CSS pixel unless you set a different `width` and `height` value for the image element.

Zooming.

- Likewise, one CSS pixel equals one device pixel unless you zoom.
- Zooming stretches CSS pixels over more device pixels visually, but does not change element size.

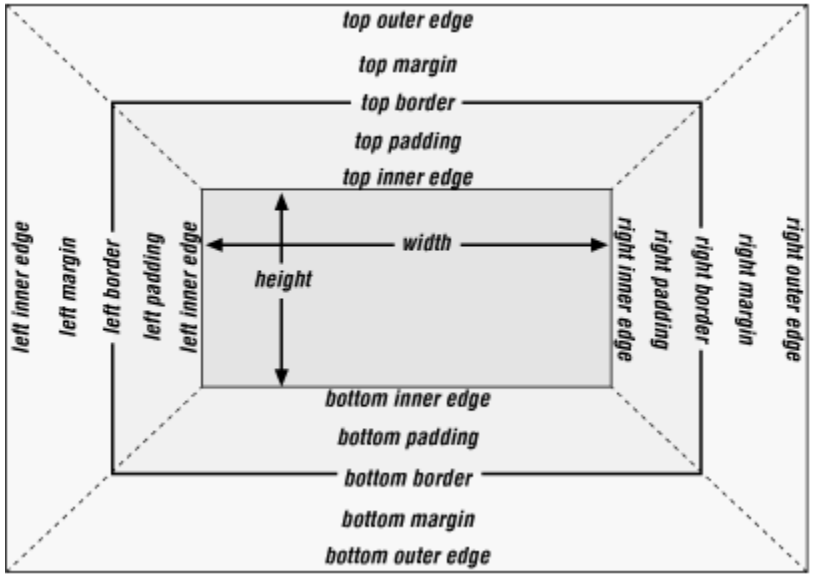
The point here is that ***you are mostly interested in CSS pixels.***

Those are the pixel size you have most in your control.

Box-sizing

What part of the element does `width` and `height` refer to?

```
* {
  box-sizing: content/border-box;
}
```



`content-box`

Default value. The width and height properties (and min/max properties) includes only the content. Border, padding, or margin are not included

`border-box`

The width and height properties (and min/max properties) includes content, padding and border, **but not the margin**
- makes it easier to scale correct

Box-sizing Example

```
img {
  width: 10%;
  height: auto;
  background-color: greenyellow;
  border: 8px solid black;
}

#image2, #image4 {
    padding: 8px;
}

#image3, #image4 {
    box-sizing: border-box;
}
```

Media Query – How To Make Things Really Responsive

Change the CSS styling to adjust to the client media (screen resolution/window size).

Select different stylesheets depending on media:

```
<link media="mediatype and (media feature)"
href="myStylesheet.css" rel="stylesheet" type="text/css">
```

Or manipulate the CSS Styles directly:

```
@media mediatype and (media feature) {
  CSS-Code;
}
```

Media Query – Select Different Stylesheets

```
<header>
  <link rel="stylesheet" media="screen and (max-width: 480px)"
  href="css/smallStylesheet.css" type="text/css">

  <link rel="stylesheet" media="screen and (min-width: 481px)"
  href="css/largeStylesheet.css" type="text/css">
  …
</header>
```

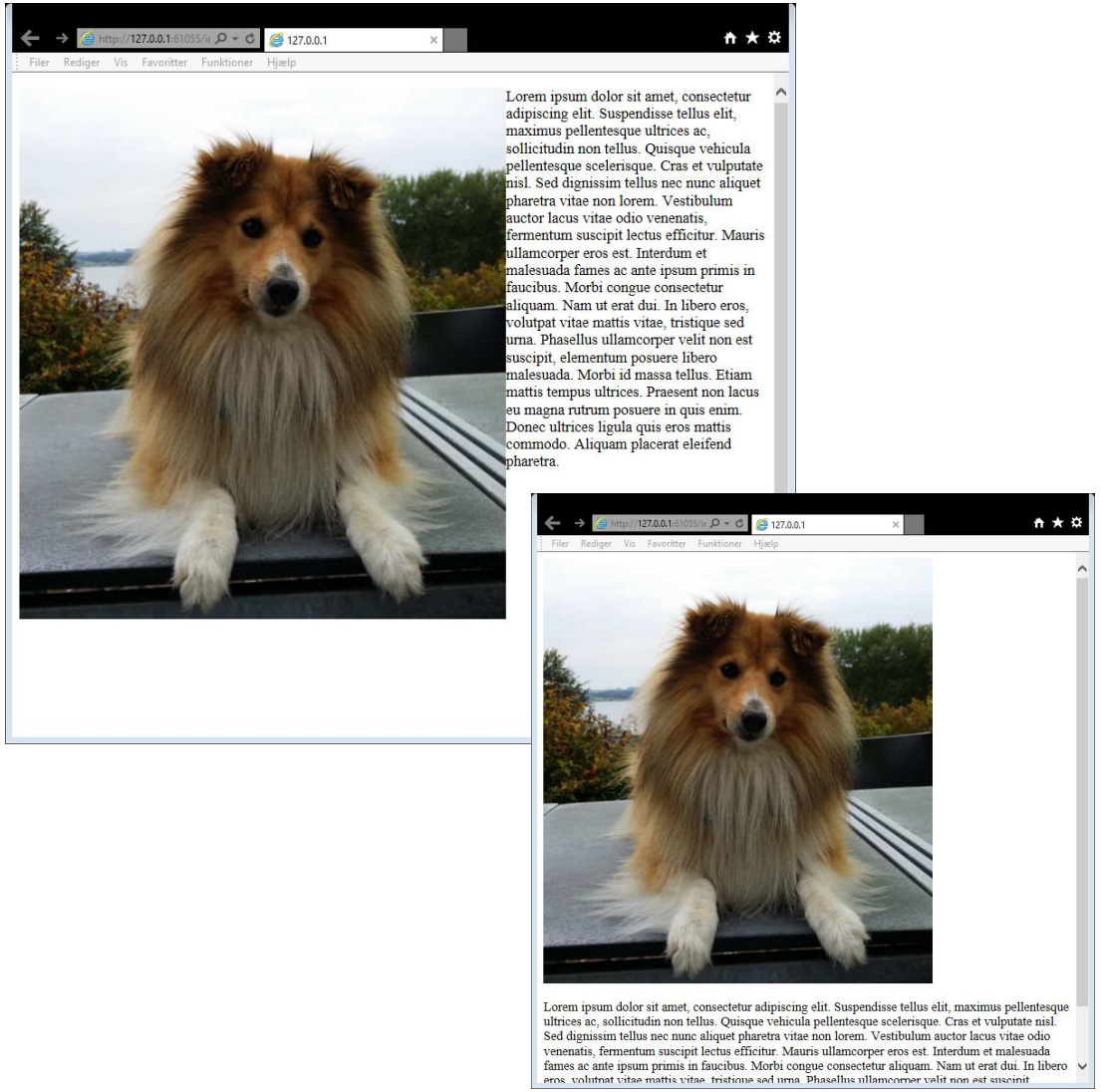
Breakpoints

When `mediatype` is `screen` and `media feature` is `max-width/min-width`, you are comparing against browser window size (CSS pixels) and not the actual device size (to do that, use `max-device-width/min-device-width` as `media feature` instead).

Media Query – In the CSS

```
@media screen and (min-width: 800px) {
    img {
        float: left;
    }
    …
}
```

Breakpoint



Hide Elements

Elements like divisions, images etc. can be hidden (not displayed).

```
@media screen and (max-width: 800px) {
   img {
      display: none;
   }
   …
}
```

Where to Set Breakpoints

- Your instinct might be to write media query breakpoints around common screen resolutions, such as 320px, 480px, 768px, 1024px, 1224px, and so forth,
  - This is a bad idea

- When building a responsive website it should adjust to an array of different window sizes, regardless of the device. Breakpoints should **only** be introduced when a website starts to break, look weird, or the experience is being hampered

- Additionally, new devices and resolutions are being released all of the time. Trying to keep up with these changes could be an endless process

Multiple Breakpoints

```
@media screen and (max-width: 480px) {
  …
}

@media screen and (min-width: 481px) and (max-width: 1024px) {
  …
}

@media screen and (min-width: 1025px) {
  …
}
```

Screen Orientation

```
@media screen and (max-width: 680px) and (orientation:
landscape) {

  …

}
```

```
@media screen and (max-width: 680px) and (orientation:
portrait) {

  …

}
```

Normal HTML Layout

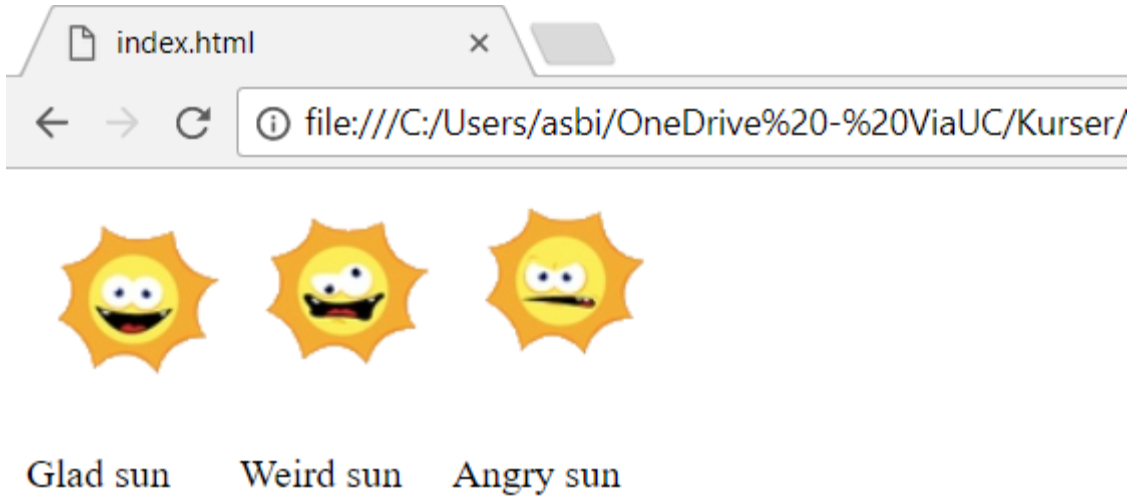
```
<div class="flex-container">
  <div>
    <img src="images/sun_glad.png">
      <p>Glad sun</p>
  </div>
  <div>
    <img src="images/sun_weird.png">
    <p>Weird sun</p>
  </div>
  <div>
    <img src="images/sun_angry.png">
    <p>Angry sun</p>
  </div>
</div>
```

Flexbox

- A new way to layout your websites with CSS (new CSS3)
- Very powerful and useful for aligning elements, distributing space equally between elements and much more

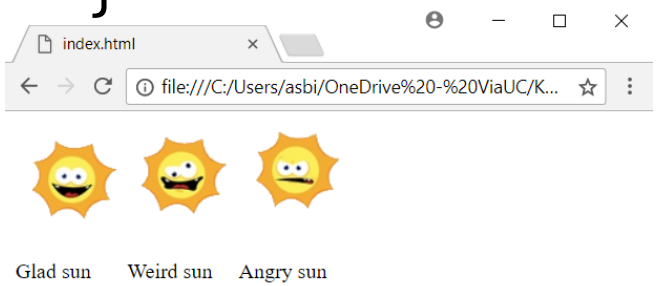
How to use it (CSS):

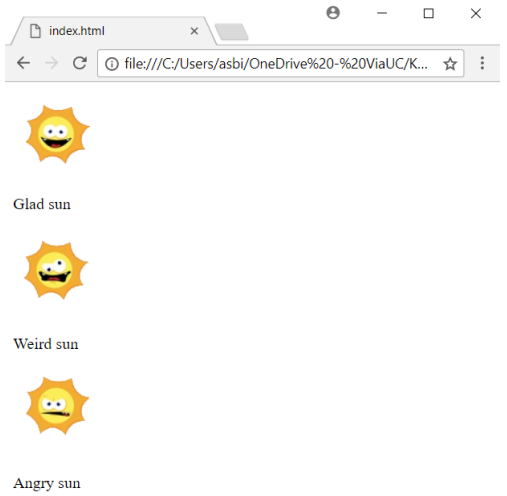
```
.flex-container {
  display: flex;
  width: 1000px;
  height: 1000px;
}
```

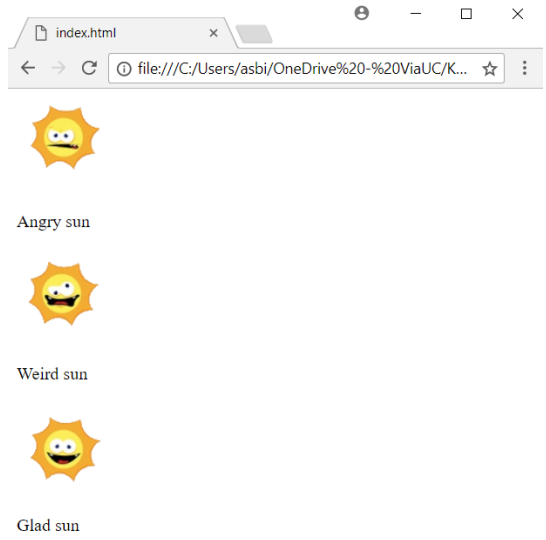


Flexbox - Direction

```
.flex-container {
  display: flex;
  flex-direction: row|row-reverse|column|column-reverse;
  width: 1000px;
  height: 1000px;
}
```







Flexbox – Justify Content

Aligns content inside the container horizontally.

```
.flex-container {
  display: flex;
  justify-content: flex-start|flex-end|center|space-between| space-around|initial|inherit;
  width: 1000px;
  height: 1000px;
}
```

See examples:
https://www.w3schools.com/cssref/css3_pr_justify-content.asp

Flexbox – Align Content

Aligns content inside the container vertically*.

```
.flex-container {
  display: flex;
  align-items: stretch|center|flex-start|flex-end|baseline|initial|inherit;
  width: 1000px;
  height: 1000px;
}
```

See examples:
https://www.w3schools.com/cssref/css3_pr_align-items.asp

*: As previously mentioned, aligning vertically can be a tad more tricky than horizontally.

Other Flexbox Properties

`flex-wrap`

`flex-flow`

`align-content`

See: https://www.w3schools.com/css/css3_flexbox.asp

The Remainder of the Day

- The mobile device emulator in Chrome.
- **Todays Exercise and homework for next session**:
  - Additional responsive features.

Responsive Web Design

VIA University College

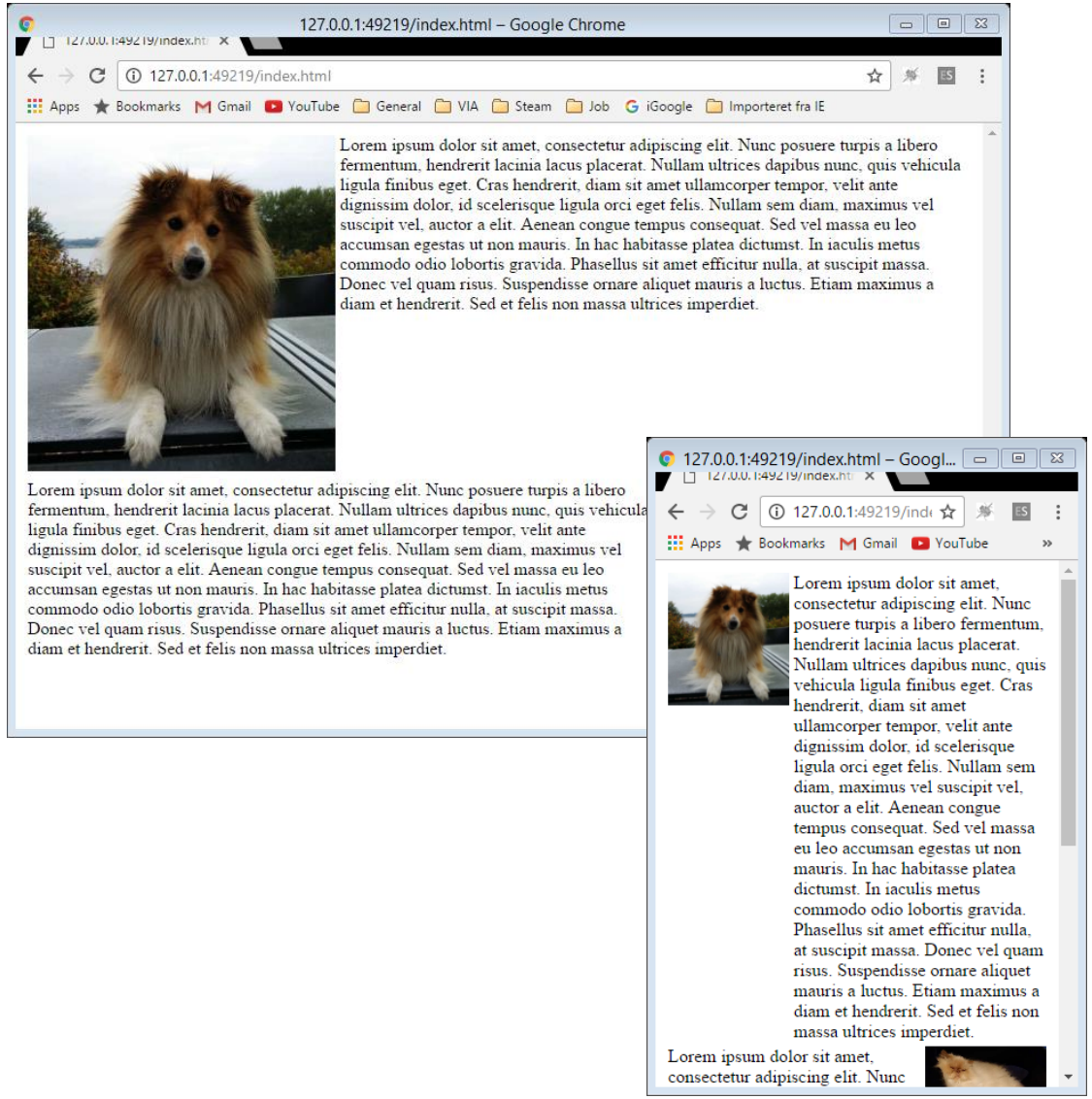
Today’s topics (session 6)

- Questions to the exercises from last time?
- The Bootstrap Framework
- Grid Layout
- Navigation bar
- Carousel
- Exercises

The (Not So) Responsive Table

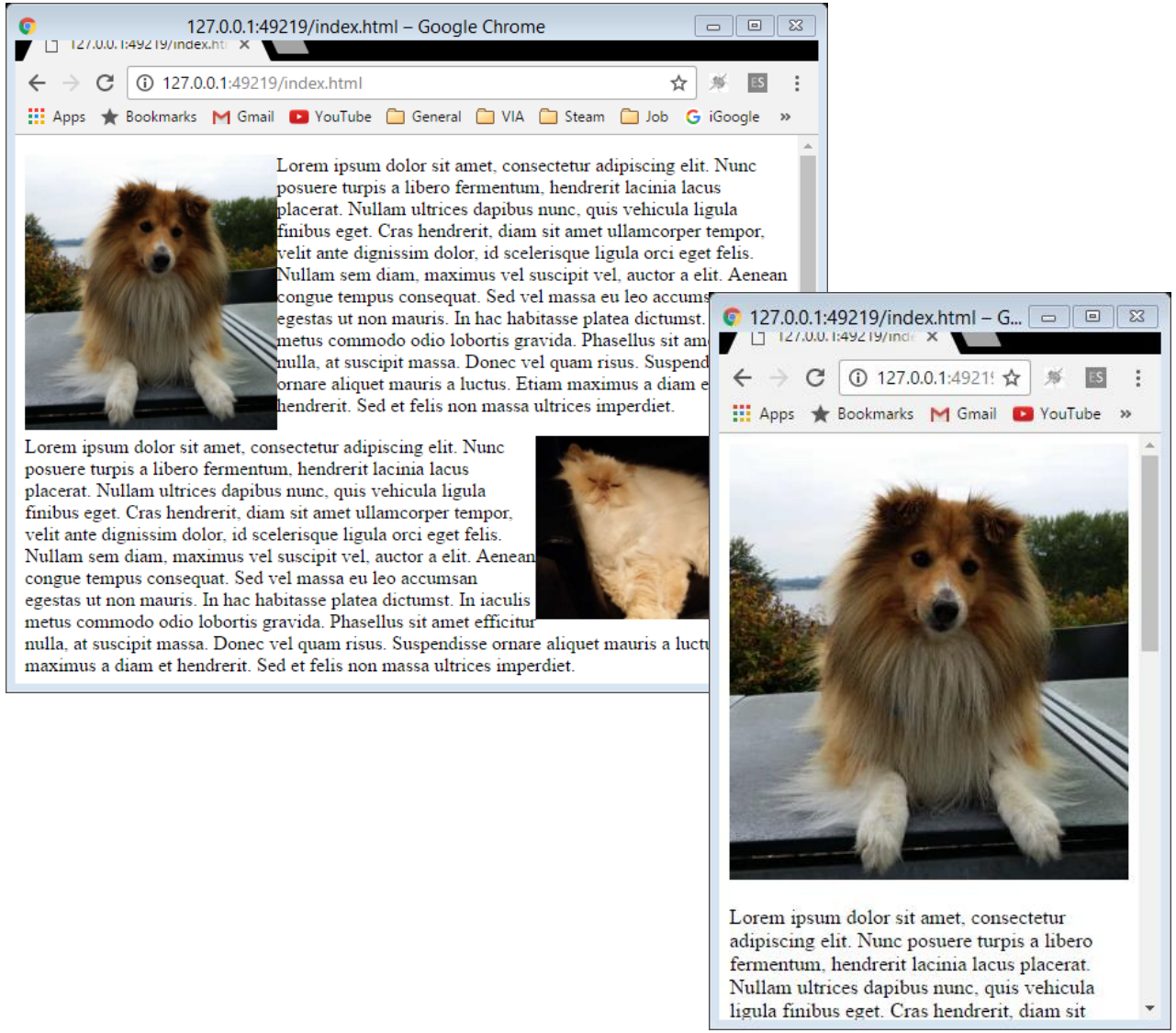
```
<table>
  <tr>
    <td>…</td>
    <td colspan="2">…</td>
  </tr>…
</table>
```

```
td {
    width: 33%;
}
```



The Responsive Float

```
<div>
  <img …>
  <p>…</p>
</div>
…

@media screen and (min-width: 650px) {
    #nickiImg {
        width: 33%;
        float: left;
    }

    #simbaImg {
        width: 33%;
        float: right;
    }
}
```



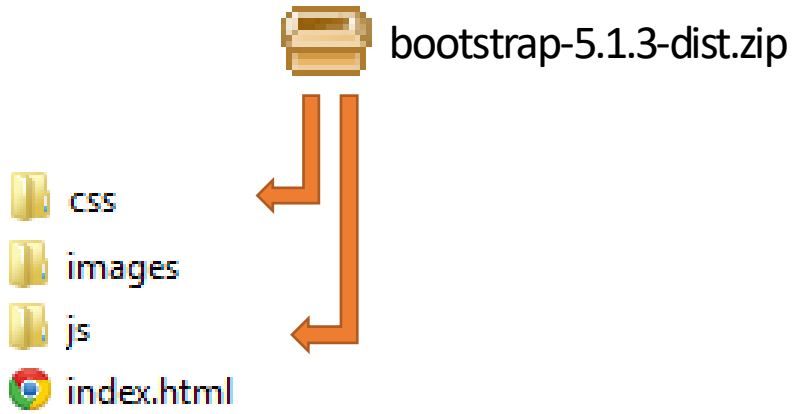
Combining the best of both – The Bootstrap Grid Framework

In General – The Bootstrap Framework

- Latest major release version: 5.1
- A collection of CSS styling and JavaScript files.
- Can be customized.
- Controlled by assigning keyword class names in your HTML tags.
- Greatly reduces the amount of CSS code you have to write.
- In fact, setting or changing CSS properties while at the same time trying to utilize Bootstrap features, while often cause the features not to function properly.
  - Especially `width`, `height`, `float` , `position` and `display` properties.

What To Do

- Go here: https://getbootstrap.com/docs/5.1/getting-started/download/
- Download the Bootstrap file bundle.
- Unzip and add the Bootstrap related CSS, font and JavaScript files to your webpage root folder.



dist/js/bootstrap.bundle.min.js

Adjust Your HTML File(s)

To be able to utilize Bootstrap features in a single HTML document, you need to add the following lines:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="css/bootstrap.min.css" type="text/css" rel="stylesheet">
    <link href="css/styles.css" type="text/css" rel="stylesheet">

    <script src="js/bootstrap.bundle.min.js"></script>
  </head>
  <body>
    …
  </body>
</html>
```

To ensure proper rendering and touch zooming for all devices, add this responsive viewport meta tag.

Your own external CSS styling file (be careful what properties you overwrite).

Bootstrap use JavaScript to support its features, so a link to a JS bundle is needed.

Grid Layout – One of Many Bootstrap Features

Bootstrap's grid layout divides the page (or element) into 12 columns.
If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Bootstrap's grid layout is responsive, and the columns will re-arrange automatically depending on the browser window size.

Grid Layout – Syntax

```
<div class="container">
  <div class="row">
    <div class="col-α-β">…</div>
  </div>
</div>

<div class="container">
  <div class="row">
    <div class="col-md-2">…</div>
    <div class="col-md-4">…</div>
    <div class="col-md-6">…</div>
  </div>
</div>
```

Breakpoints!

α
- sm (small)
- md (medium)
- lg (large)
- xl (extra large)
- xxl (extra extra large)

No. columns this element should cover

β
- 1
- 2
- 3
- …
- 12

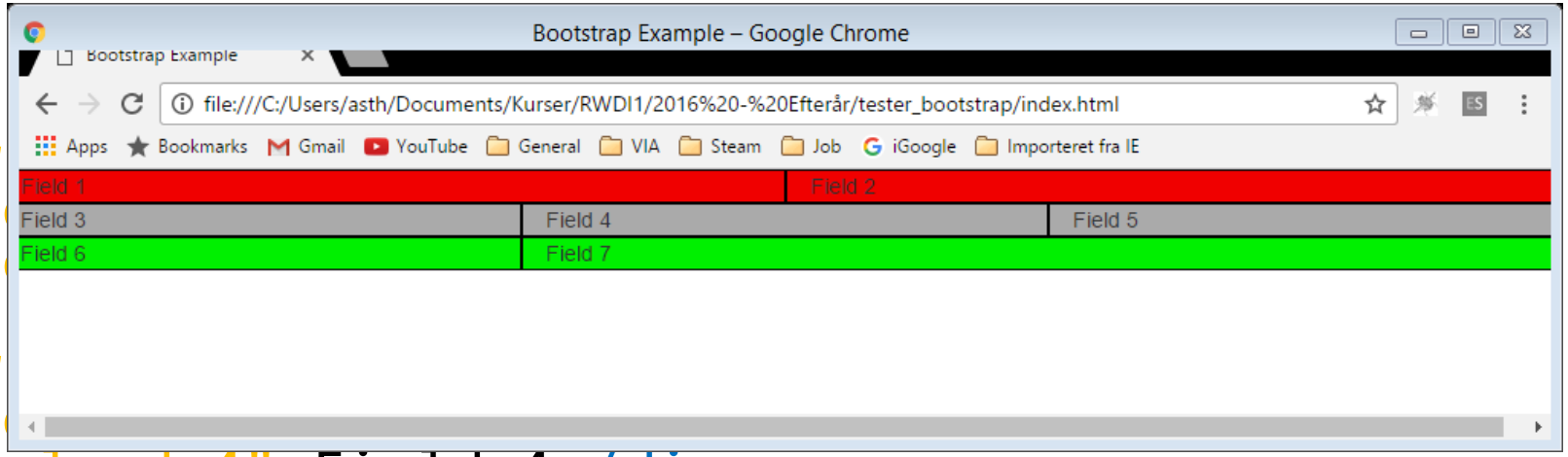
Grid Layout - Example

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Field 1</div>
    <div class="col-md-6">Field 2</div>
  </div>
  <div class="row">
    <div class="col-md-4">Field 3</div>
    <div class="col-md-4">Field 4</div>
    <div class="col-md-4">Field 5</div>
  </div>
  <div class="row">
    <div class="col-md-4">Field 6</div>
    <div class="col-md-8">Field 7</div>
  </div>
</div>
```

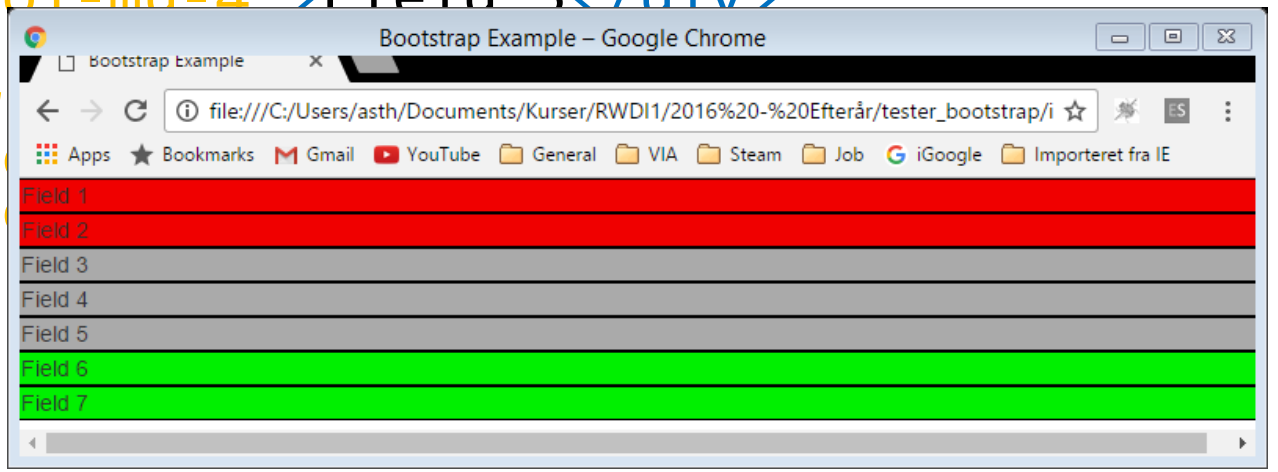
Grid Layout - Example

```
<div class="conta
    <div class="row
      <div class="c
      <div class="col-md-4">Field 4</div>
      <div class="col-md-4">Field 5</div>
    </div>
    <div class="row
```

Browser window size greater than the md breakpoint.



Browser window size smaller than the md breakpoint.



Grid Layout - Breakpoints

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px	Extra extra large ≥1400px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`

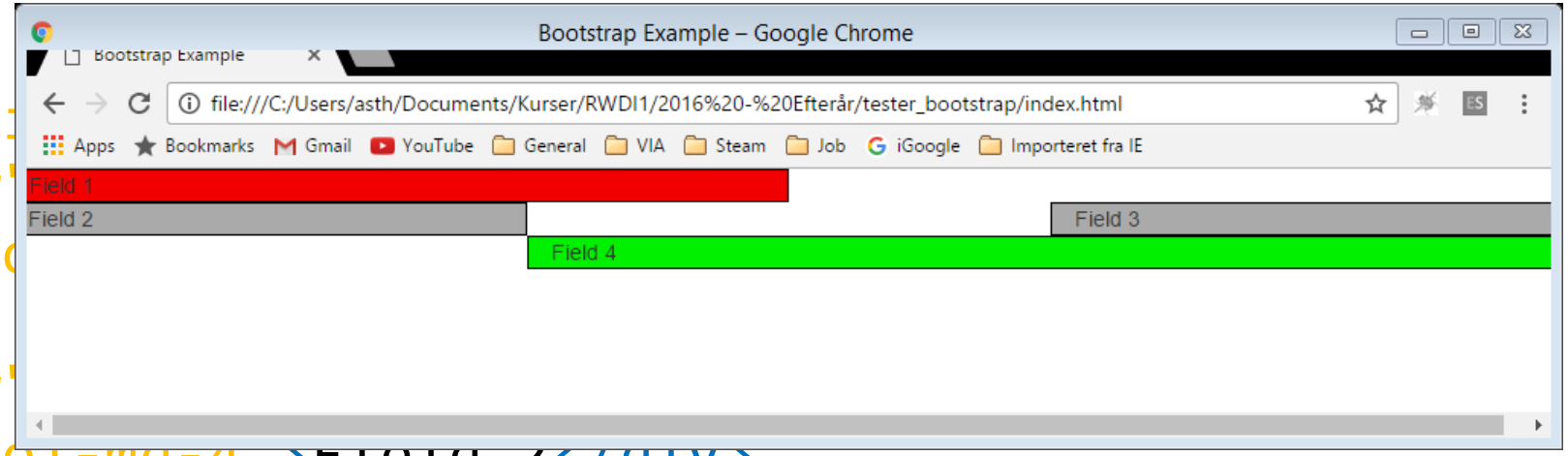
Grid Layout – Offset and Empty Columns

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Field 1</div>
  </div>
  <div class="row">
    <div class="col-md-4">Field 2</div>
  <div class="col-md-4 offset-md-4">Field 3</div>
  </div>
  <div class="row">
    <div class="col-md-8 offset-md-4">Field 4</div>
  </div>
</div>
```

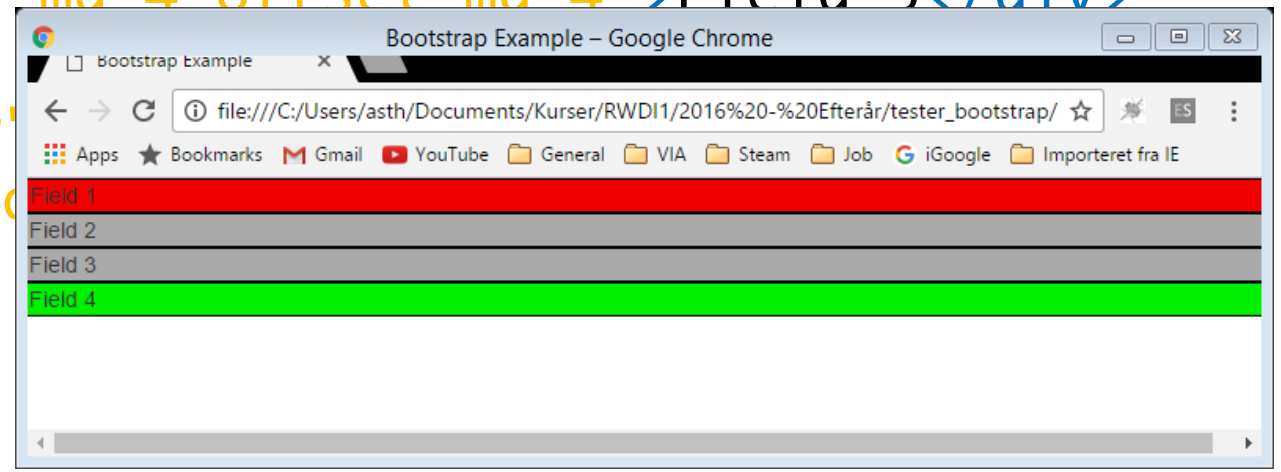
Grid Layout – Offset and Empty Columns

```
<div class="conta
    </div>
    <div class="row
      <div class="col-md-4 ">Field 2</div>
    <div class="col-md-4 offset-md-4">Field 3</div>
    </div>
    <div class="row
    </div>
```

Browser window size greater than the md breakpoint.



Browser window size smaller than the md breakpoint.



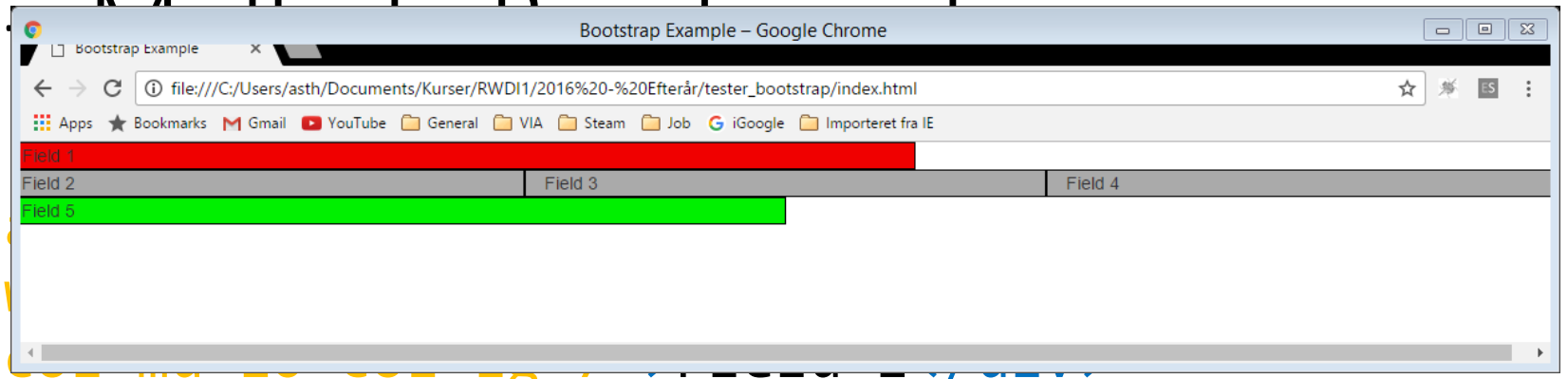
Grid Layout – Multiple Breakpoints

```
<div class="container">
  <div class="row">
    <div class="col-md-10 col-lg-7">Field 1</div>
  </div>
  <div class="row">
    <div class="col-sm-6 col-md-4">Field 2</div>
    <div class="col-sm-4">Field 3</div>
    <div class="col-sm-2 col-md-4">Field 4 </div>
  </div>
  <div class="row">
    <div class="col-sm-6 col-m      -lg-6">Field 5</div>
  </div>
</div>
```

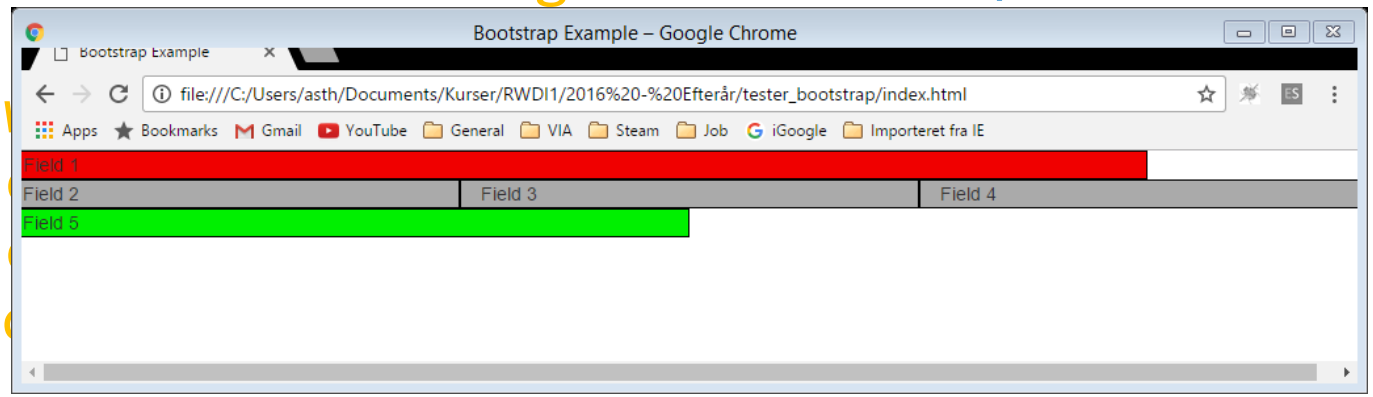
Completely redundant; avoid at all cost!

Grid Layout

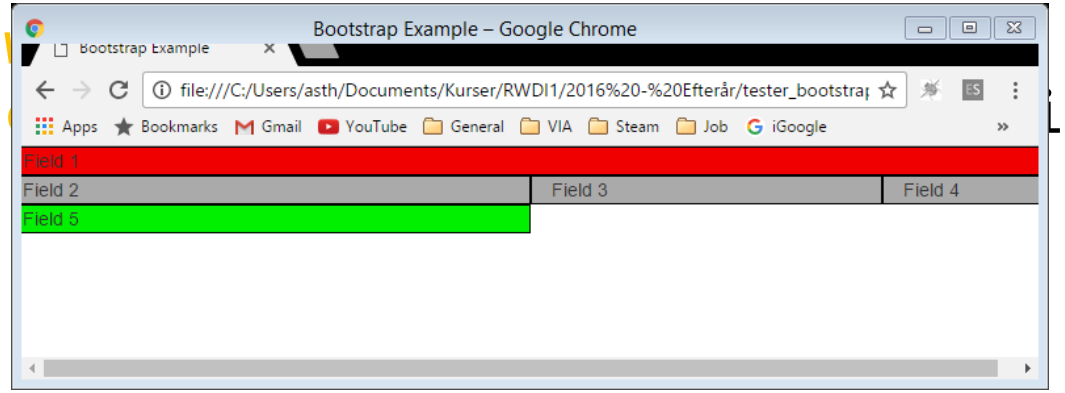
Browser window size greater than the lg breakpoint.



Browser window size greater than the md breakpoint.



Browser window size greater than the sm breakpoint.



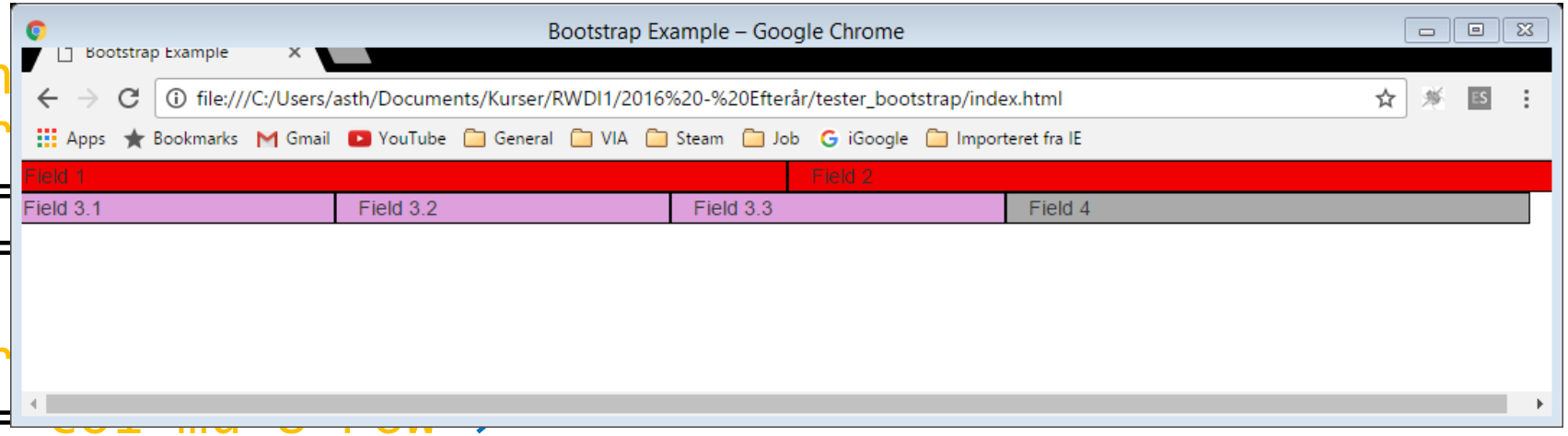
Grid Layout – Nested Grids

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Field 1</div>
    <div class="col-md-6">Field 2</div>
  </div>
  <div class="row">
    <div class="col-md-8 row">
      <div class="col-md-4">Field 3.1</div>
      <div class="col-md-4">Field 3.2</div>
      <div class="col-md-4">Field 3.3</div>
    </div>
    <div class="col-md-4">Field 4</div>
  </div>
</div>
```

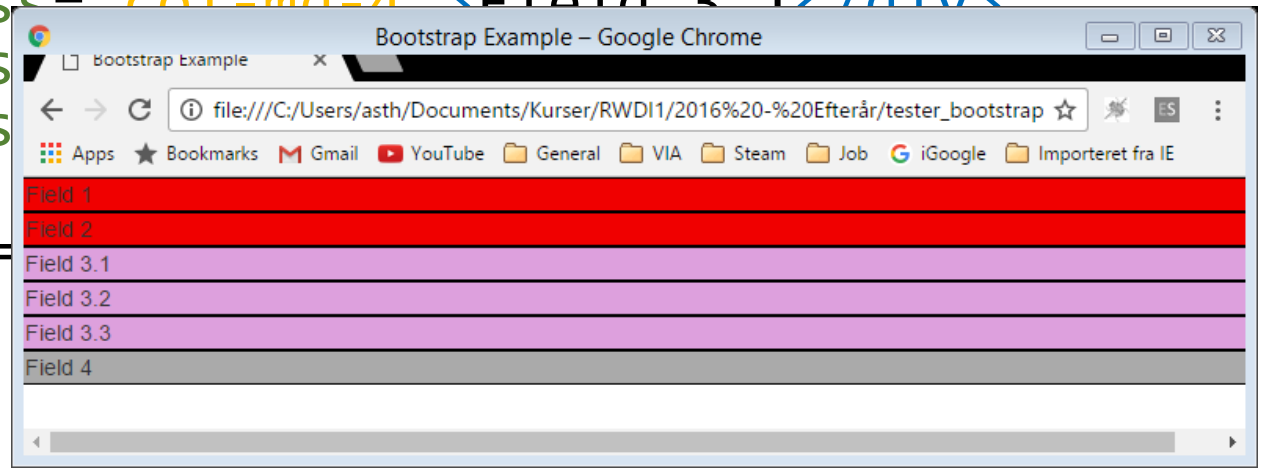
Grid Layout – Nested Grids

```
<div class="con
    <div class="n
        <div class="n
            <div class=
                <div class="col-md-4">Field 3.1</div>
                <div clas
                <div clas
    </div>
```

Browser window size greater than the md breakpoint.



Browser window size smaller than the md breakpoint.



Grid Layout – Simplified Notation

```
<div class="row">
    <div class="col-md-4">Field 1</div>
    <div class="col-md-4">Field 2</div>
    <div class="col-md-4">Field 3</div>
</div>
```

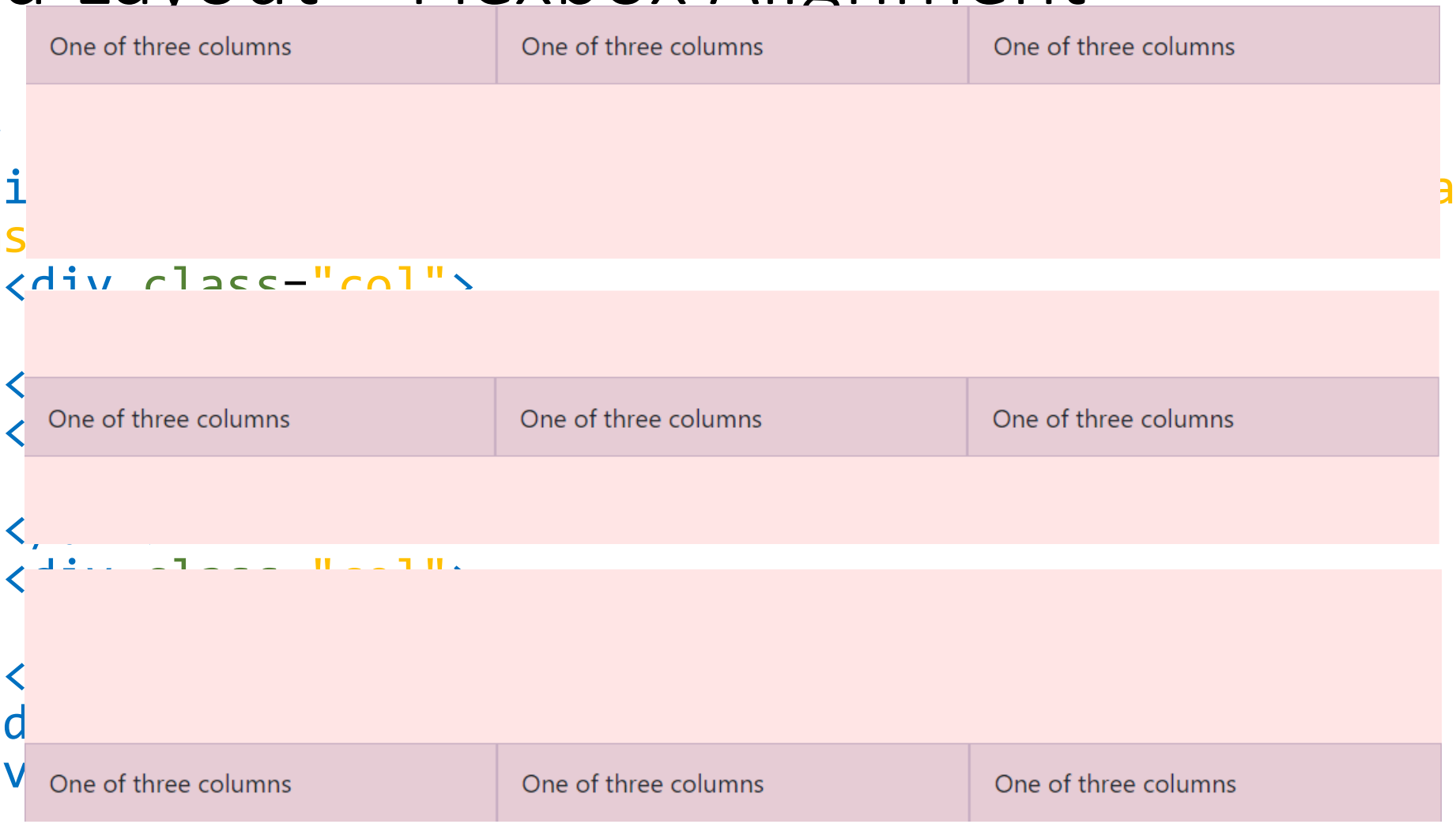
Alternative and better:

```
<div class="row">
    <div class="col-md">Field 1</div>
    <div class="col-md">Field 2</div>
    <div class="col-md">Field 3</div>
</div>
```

Grid Layout – Flexbox Alignment

```
<div class="container">
  <div class="row align-items-start|align-items-center|align-
items-end">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>
```

Grid Layout – Flexbox Alignment



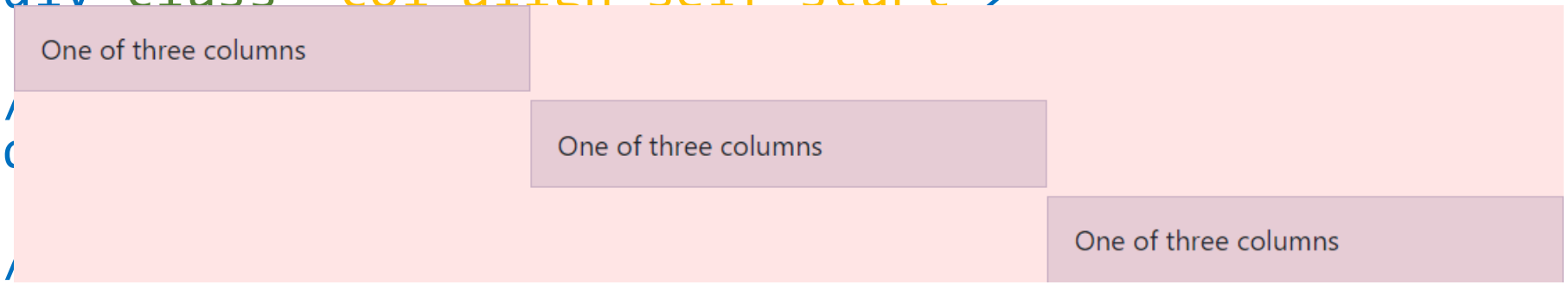

```
<div
  <di                                                    align-
items
    <div class="col">

    <
    <

    </
    <div class="col">

    <

  </d
</div
```

Grid Layout – Flexbox Alignment

```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```

Grid Layout – Flexbox Alignment

```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```



Grid Layout – Other Familiar Flexbox Properties

```
<div class="container">
  <div class="row justify-content-start|justify-content-
center|justify-content-end|justify-content-around|justify-
content-between">
…
```

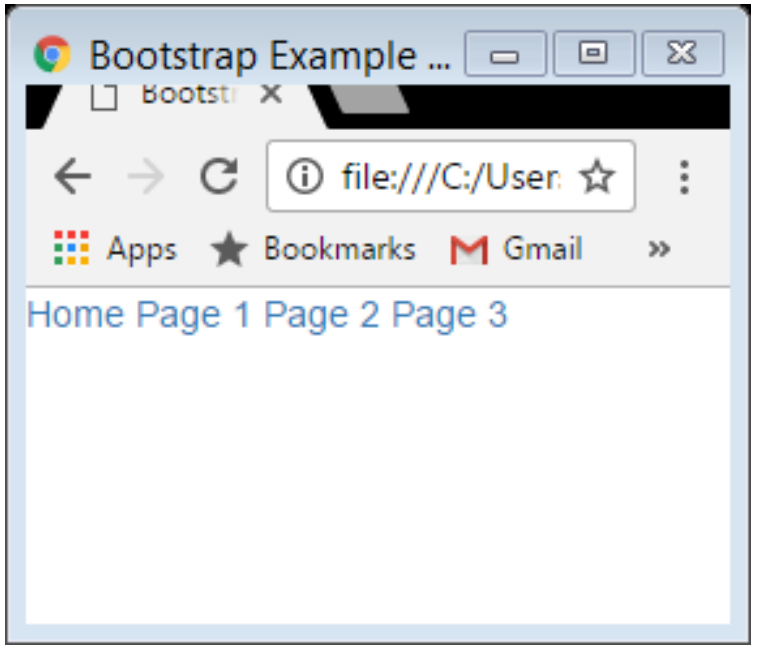
```
<div class="container">
  <div class="row">
    <div class="col order-(first,last,1,2,…,12)">
    Or
    <div class="col order-(sm,md,lg,xl)-
(first,last,1,2,…,12)">
…
```

A look at some other Bootstrap Features

Navigation Bar

The HTML 5 navigation bar tag:

```
<nav>
  <a href="#">Home</a>
  <a href="#">Page 1</a>
  <a href="#">Page 2</a>
  <a href="#">Page 3</a>
</nav>
```



The # should be replaced with an URL path to the corresponding HTML document file for that page.

Navigation Bar

Creates a standard Bootstrap navigation bar

Stacks the navbar vertically on small screens
navbar-expand-xl|lg|md|sm

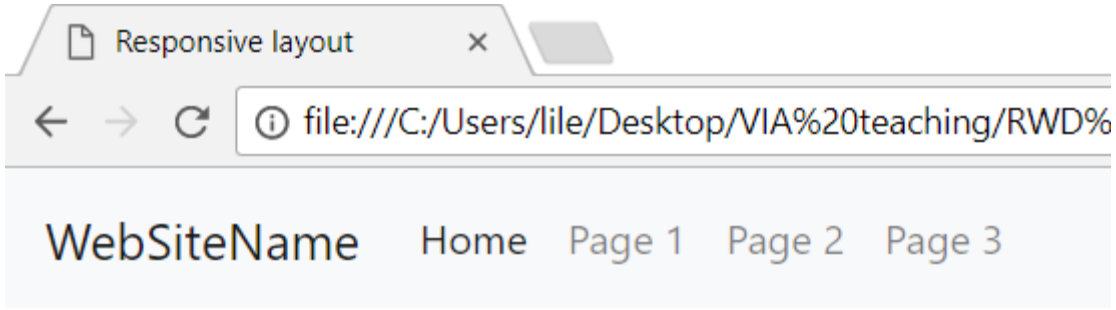
Applies the bg-light color scheme to the navbar and black text

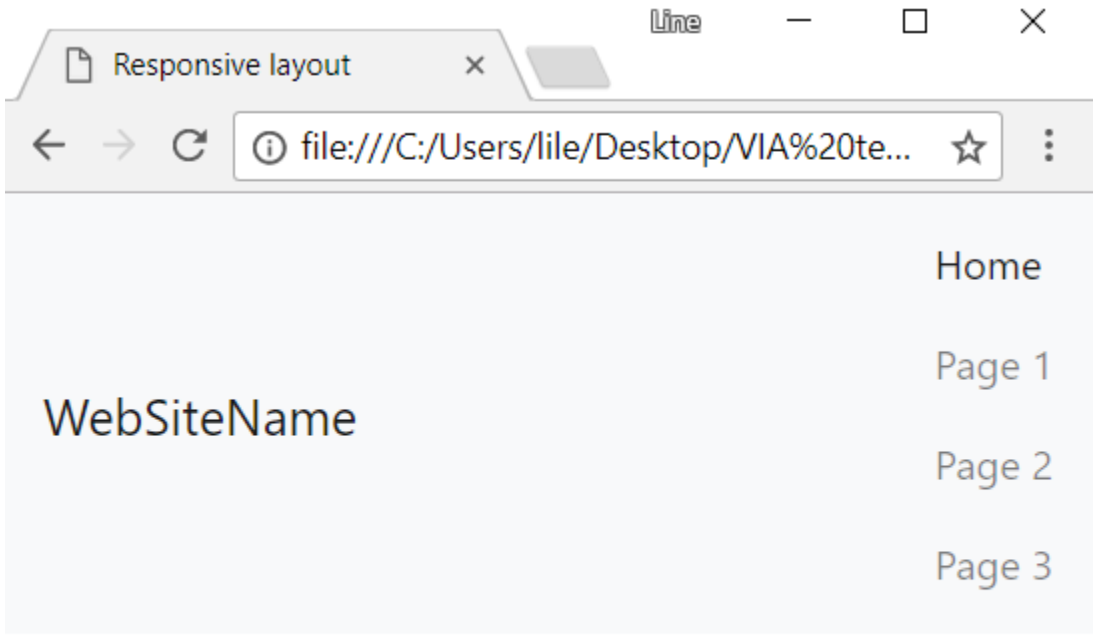
```
<nav class="navbar navbar-expand-sm bg-light navbar-light">
<a class="navbar-brand" href="#">WebSiteName</a>

  <ul class="navbar-nav">
    <li class="nav-item active"><a class="nav-link" href="#">Home</a></li>
    <li class="nav-item"><a class="nav-link" href="#">Page 1</a></li>
    <li class="nav-item"><a class="nav-link" href="#">Page 2</a></li>
    <li class="nav-item disabled"><a class="nav-link" href="#">Page 3</a></li>
  </ul>
</nav>
```

Creates a header section in the navigation bar, where you can put a link to the front page of the website (index.html) and/or a logo.

Creates links to 4 pages (Home included) and styles one of the links as active and one as disabled.

Navigation Bar





Build-in responsiveness!

The navigation bar collapse at the `sm` break point.

Navigation Bar – Collapsible

```
<nav class=" navbar navbar-expand-sm bg-light navbar-light">
 <a class="navbar-brand" href="#">WebSiteName</a>
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#myNavbar">
  <span class="navbar-toggler-icon"></span>
 </button>

 <div class="collapse navbar-collapse" id="myNavbar">
  <ul class="navbar-nav">
   <li class="nav-item active"><a class="nav-link" href="#">Home</a></li>
   <li class="nav-item"><a class="nav-link" href="#">Page 1</a></li>
   <li class="nav-item"><a class="nav-link" href="#">Page 2</a></li>
   <li class="nav-item disabled"><a class="nav-link" href="#">Page 3</a></li>
  </ul>
 </div>
</nav>
```

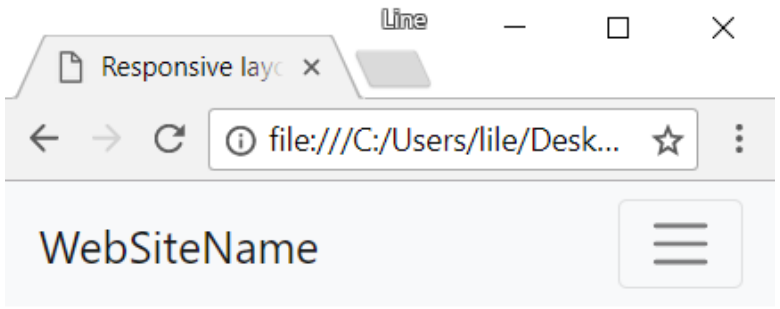
A button element is being placed inside the navigation bar.

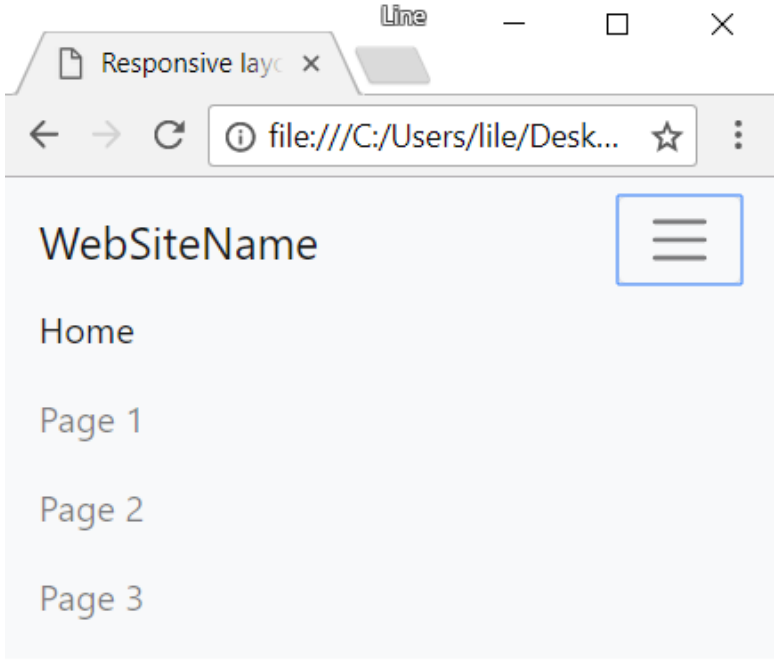
Its data target is the id of the just added division element.

The button displays three icon bars.

Same code as before, except the unordered list element has been encapsulated in a division element, which has the appropriate Bootstrap classes added and the `id` attribute set.

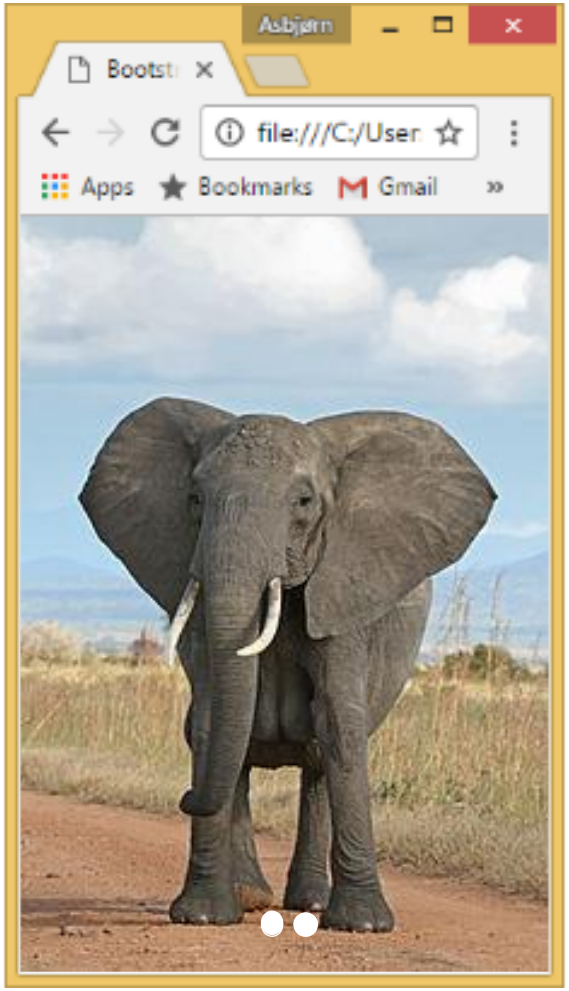
Navigation Bar – Collapsible





Same breakpoint as before.

Carousel



Carousel

Note:

Before starting coding your carousel, it is a good idea to use a image editing tool to ensure that all images have the same width and height.

At the very least they should have the same height and then center aligned (.mx-auto).

Carousel

The outmost element of the carousel.
Choose an `id` value.
Add the `slide` class if you want the carousel to do a sliding transition between images.

```
<div id="myCarousel" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner"> <!-- Wrapper for slides -->
    <div class="carousel-item active">
      <img class="d-block img-fluid mx-auto" src= "images/elephant.jpg" alt="elephant">
    </div>
    <div class="carousel-item">
      <img class="d-block img-fluid mx-auto" src="images/tiger.jpg" alt="tiger">
    </div>
    <div class="carousel-item">
      <img class="d-block img-fluid mx-auto" src="images/giraffe.jpg" alt="giraffe">
    </div>
  </div>
  <ul class="carousel-indicators"> <!-- Indicators -->
    <li data-bs-target="#myCarousel" data-bs-slide-to="0" class="active"></li>
    <li data-bs-target="#myCarousel" data-bs-slide-to="1"></li>
    <li data-bs-target="#myCarousel" data-bs-slide-to="2"></li>
  </ul>
</div>
```

Add all the images the carousel should cycle between.
One parent division needs to have the `active` class added.

Add classes `d-block img-fluid mx-auto` to the display block, to make the image responsive, and center align the images

Add one carousel indicator for each image.
The indicator with the class `active` added should correspond to the image whose parent division got the class as well.

Exercises and itslearning learning path

- Find the exercises on itslearning
  - Open it in Visual Studio Code to see the HTML structure and content
  - See the result in the browser window
  - Go step by step and do the exercises
  - Just ask me if you get stuck ☺

- When you finish the exercises, go through the learning path on itslearning

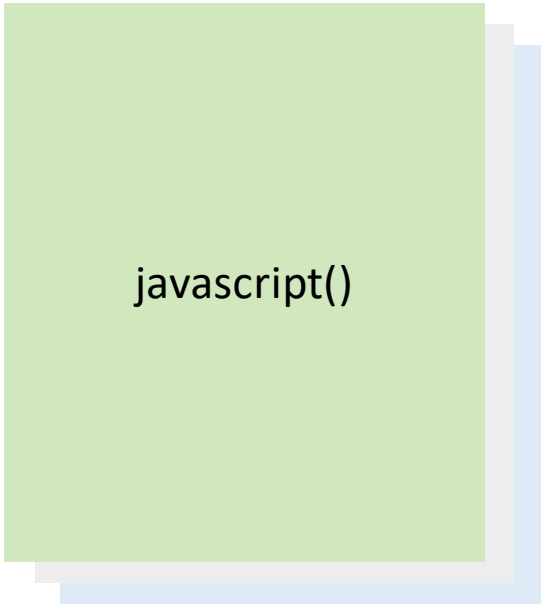
Responsive Web Design

VIA University College

Today’s Agenda

- What is JavaScript?
- The Document Object Model.
- Adding JavaScript to Your HTML Document.
- Variables.
- Manipulating HTML Elements.
- Arrays .
- Functions.
- Global Objects: Math & Date.
- Truthy and Falsy Expressions.
- Loops.
- Timeout & Interval.
- Exercises.

The Buildup



Behavior Layer

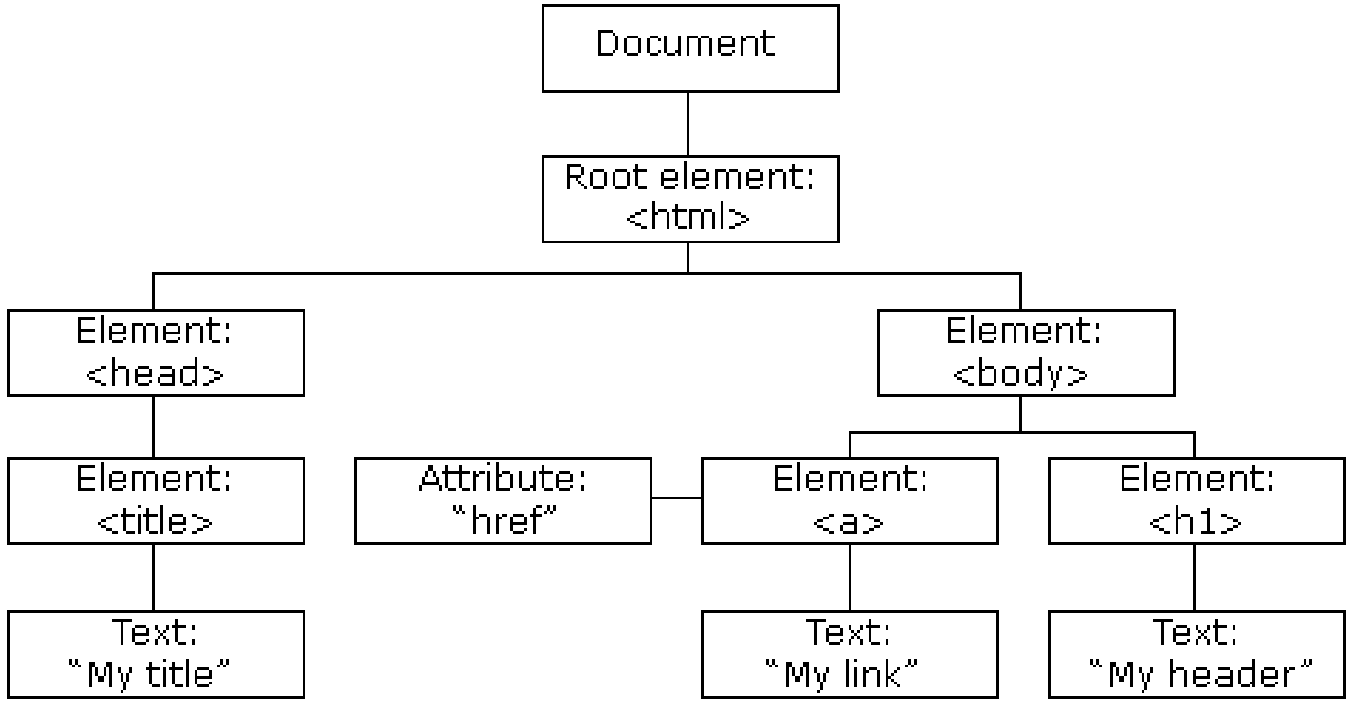
What is JavaScript

A Scripting language

- Typically used for client/browser scripting.
- Not server side.
- JavaScript cannot do changes to the HTML document, but instead affects the Document Object Model which has been instanced by the browser, based on the HTML document.
- Most (if not all) browsers are equipped with the ability to interpret HTML, CSS and JavaScript code syntax, so you do not have to "include" anything for the browser to be able to work with JavaScript.

Document Object Model (DOM)

- All HTML tags that the browser interprets are stored in the DOM in the computer memory.
- Tags define elements in the DOM.
- JavaScript is used to manipulate these elements and the results are then "instantly" reflected in the browser.



Where to Put the JavaScript Code

- In the <body> section of the HTML document:

```
<body>
  <div> This is a section before the script </div>
  <script>
    <!--Put the JavaScript code here-->
    …
  </script>
  <div> This is a section after the script </div>
</body>
```

- The script is loaded/executed at this point when the browser is interpreting the HTML document.
- Similar to CSS in-line styling, we do NOT do this.

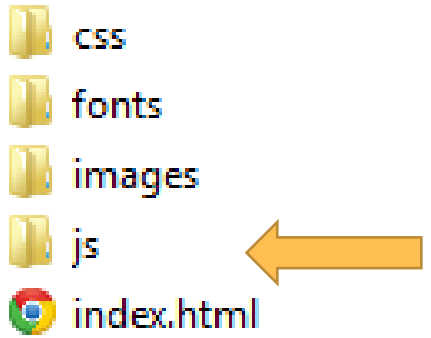
Where to Put It Instead

- In (a) separate file(s) with .js extension:

```
<body>
   <div> This is a section before the script </div>
   <script src="js/myScript.js"></script>
   <div> This is a section after the script </div>
</body>
```

- Preferable in the bottom (but still inside) of the <body> section of the HTML document. This is to ensure all elements have been created in the DOM before the script executes. A script cannot access elements which are declared after the script has been loaded.

– Store JavaScript files with the .js-extension in an appropriately named subfolder:



JavaScript Vs. Java

```
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println(message(1));
  }

  private String message(int aInput) {
    String output;
    if (aInput == 1) {
      output = "Hello, World";
    } else if (aInput == 2) {
      output = "Goodbye, World";
    } else
      output = "Unknown input";
    }
  return output;
  }
}
```

```
document.write(message(1));

function message(aInput) {
  var output;
  if (aInput == 1) {
    output = "Hello, World";
  } else if (aInput == 2) {
    output = "Goodbye, World";
  } else {
    output = "Unknown input";
  }
  return output;
}
```

JavaScript is NOT Java!

Variable Declaration

```
var myVariable, myOtherVariable;
myVariable = 1234;

var myThirdVariable = "Some text";

myVariable = "Also some text";

myvariable = "This text";
```

JavaScript variables can also be used to store an object or element reference.

Manipulating HTML Elements

Select an element:

```
document.getElementById("id")
```

Eg. In the HTML document:

```
<div id="labelId">
  This is a Label!
</div>
```

In the JavaScript file:

```
var theLabel = document.getElementById("labelId");
theLabel.innerHTML = "Ha, I changed it!!!";
```

`innerHTML` is the property that represents the HTML content inside the `<div>` element.

Manipulating HTML Elements

```
var myElement = document.getElementById("elementId");
myElement.innerHTML = myElement.innerHTML + "I am adding this
text after the element.";

myElement.innerHTML = "<b>" + myElement.innerHTML + "</b>";
```

Manipulating HTML Elements

Selecting more HTML elements:

```
document.getElementsByTagName("tag")
document.getElementsByClassName("class")
```

Eg. In the HTML document:

```
<div id="label">
  <h1 class="blue">This is a Label!</h1>
</div>
```

In the JavaScript file:

```
var headers = document.getElementsByTagName("h1");
var blues = document.getElementsByClassName("blue");
```

Arrays

Two different initializing options:

```
1. var colorArray = ["red", "blue", "green"];
2. var colorArray = new Array("red", "blue", "green");
```

```
var myArray = ["red",123,"green"];
```

```
var myArray = ["red",123,"green"]*2;
```

```
var myArray = [1,2,3]*2;
```

Indexing starts at 0 for first element in an array: `myArray[0];`

```
myArray[4] = 444;
```

Functions

Basic JavaScript function declaration:

```
function myFunction(argOne,argTwo ) {
  return argOne * argTwo;
}

var size = myFunction(2,3);
```

Function declarations can be made anywhere in the script file, regardless of function calls; the browser will always interpret function declarations first before looking at the rest of the JavaScript.

Anonymous Functions

```
var area = function(argOne,argTwo) {
  return argOne * argTwo;
}

var size = area(2,3);
```

This is called a functional expression, because the browser interpreter is thinking that the variable `area` is being assigned some expression, which can be a number, string, reference and, as you can see, also a function.

Functional expressions can only be utilized after they have been declared, just like all other variable assignments.

Scope

```
var variableOne = 1; ← Global scope
var variableTwo = myFunction(variableOne);

variableOne = 2;
var variableThree = myFunction(variableOne);

function myFunction(arg) {
  var variableFour = 4; ← Local scope
  variableFive = 5; ← Global scope… extremely bad practice
  return arg + variableOne + variableFour;
}
```

Global Objects: Math

Math is a global object that can always be accessed, no imports required.
It contains some mathematical constants and functions.

Constant	Description
`Math.PI`	Pi (approx. 3.14)
`Math.E`	Euler's number (approx. 2.72)

Function	Description
`Math.round(x)`	Rounds x to the nearest integer
`Math.floor(x)`	Returns x, rounded downwards to the nearest integer
`Math.ceil(x)`	Returns x, rounded upwards to the nearest integer
`Math.pow(x,y)`	Returns the value of x to the power of y
`Math.sin(x)`	Returns the sine of x (x is in radians)

See more: http://www.w3schools.com/jsref/jsref_obj_math.asp

Math.random()

`Math.random()` generates a random number between 0 (inclusive) and 1 (not inclusive).

E.g. to create a random number between 0 and 5:

```
var myNumber = Math.floor(Math.random() * 6);
```

This value could be used as index in an array to pick an element by random:

```
myArray[myNumber];
```

Global Objects: Date

Like `Math`, `Date` is a global object that can always be accessed.
It contains functions that allow you to work with dates and time.

Create a date object:

```
var myDate = new Date();
```

Given no argument, the variable will contain the date and time of its creation.

To specify a different date:

```
var myDate = new Date(1996,11,26,15,45,55);
var myDate = new Date(1996,11,26);
```

The date can also be specified as a string, see http://www.w3schools.com/js/js_dates.asp

Date Difference

```
var firstDate = new Date(2010,0,1);
var secondDate = new Date(2000,0,1);
var dateDifference = firstDate - secondDate;
```

`dateDifference` will contain the value of `315619200000` which corresponds to the number of milliseconds between the two dates.

To calculate into days, do for example:

```
var dayDifference = dateDifference / (1000 * 60 * 60 * 24);
```

Decision Making: Comparison Operators

>	Greater than	<	Less than
>=	Greater or equal to	<=	Less or equal to
==	Is equal to	!=	Is not equal to
===	Is strict equal to	!==	Is strict not equal to

"3" == 3 returns true
"3" === 3 returns false

"3" != 3 returns false
"3" !== 3 returns true

Truthy and Falsy Expressions

Falsy Expressions.
All these statements will be regarded as false:

```
if(false)
if(0)
if("")
if(10/0), or if(NaN) Not a Number
if(null), or (undefined)
if(var value)
```

Truthy Expressions.
All these statements will be regarded as true:

```
if(true)
if(1), or any other number than 0
if("carrot")
if(10/5)
if(var value = "carrot")
if("0"), or if("false")
```

Checking Equality

Expression	Result
`(false == 0)`	true
`(false === 0)`	false
`(false == "")`	true
`(false === "")`	false
`(0 == "")`	true
`(0 === "")`	false

Expression	Result
`(undefined == null)`	true
`(undefined === null)`	false
`(null == false )`	false
`(undefined == false )`	false
`(null == 0)`	false
`(undefined == 0)`	false

Expression	Result
`(NaN == null)`	false
`(NaN == NaN)`	false

Loops

for-Loop:

```
for (var myCount = 0; myCount < 10; myCount++) {
  var myVariable = 10 + myCount;
}
```

Initialization: `var myCount = 0`

Condition: `myCount < 10`

Update: `myCount++`

Variables declared in the loop initialization (`myCount`) and inside the loop (`myVariable`) have global scope!

while-Loop:

```
while(condition) {
 …
}
```

do-while-Loop:

```
do {
  …
} while(condition)
```

Timeout

The `setTimeout()` function calls another function or evaluates an expression after a specified number of milliseconds.

```
var myTimerVar = setTimeout(function(){ alert("Hello") }, 3000);
```

To stop the function call from happening before the time is up:

```
clearTimeout(myTimerVar);
```

The function is only executed once. Initializing a timeout does not pause the execution of the rest of the script.

Repeated Timeout

In JavaScript you can call a function from within itself:

```
var myCounter = 0;

timedCount();

function timedCount() {
  if (myCounter < 10) {
    myCounter++;
    setTimeout(function(){ timedCount() }, 1000);
  }
}
```

Interval

```
timedCount();

function timedCount() {
  var myCounter = 0;
  var myIntervalRef = setInterval(function() {
    if (myCounter < 10) {
      myCounter++;
    } else {
      clearInterval(myIntervalRef);
    }
  }, 1000);
}
```

The variable is being used by the expression that declares it!

The Remainder of the Day

- **Todays Exercise and homework for next session**:
  - Get familiar with JavaScript.

Responsive Web Design

VIA University College

Today’s Agenda

- Introducing jQuery
- The jQuery selector
- Events
- Effects
- Animation
- Exercises

jQuery

- jQuery is a piece of JavaScript that allows programmers easy access to common used JavaScript functionality.

- jQuery IS JavaScript written as a single .js file. Current version: jquery-3.6.0.js

- File can be downloaded for free at http://jquery.com/.

- jquery-3.6.0.min.js is the jQuery file with no spaces or carriage returns to keep the file size to a minimum.

How to Import

- Put the jQuery file in your js-subfolder, where you put all your JavaScripts.

- Add a script tag import in the body section of your HTML document, before your own script import:

```
<body>
  …
  <script src="js/jquery-3.6.0.min.js"></script>
  <script src="js/script.js"></script>
</body>
```

Selector Symbols

jQuery utilizes CSS syntax in its selector `$(…)`.

- `$("value")` All elements created using that HTML tag. Ex. `$("li")`
- `$("#value")` All elements with that id. Ex. `$("#listItem1Id")`
- `$(".value")` All elements with that class. Ex. `$(".listItemClass")`
- `$("*")` Simply all elements.

```
<ul id="listId" class="listClass">
        <li id="listItem1Id" class="listItemClass"/>
        <li id="listItem2Id" class="listItemClass"/>
        <li id="listItem3Id" class="listItemClass"/>
</ul>
```

Do Stuff With Your Selection

With plain JavaScript:

```
var listItemArray = document.getElementsByTagName("li");
for (var i = 0; i < listItemArray.length; i++) {
      listItemArray[i].textContent = "Hello";
}
```

With jQuery:

```
$("li").text("Hello");
```

Text Content?

`.innerHTML` specifies that the content should be regarded by the browser interpreter as HTML.

- Accessed with the `.html("blablabla")` function when using a jQuery selector.

`.textContent` specifies that the content should be regarded by the browser interpreter as text, regardless of any HTML tags/symbols. This is more safe!

- Accessed with the `.text("blablabla")` function when using a jQuery selector.

You can also remove the text/HTML content inside a jQuery selection with `.remove()`

Getting and Setting Attribute Values

Getting the attribute value for an element:

```
var srcValue = $("#elementId").attr("src");
```

Setting the attribute value for an element:

```
$("#elementId").attr("src","someValue");
```

An attribute value can also be removed:

```
$("#elementId").removeAttr("src");
```

Adding and Removing Class Associations

Adding a class for an element:

```
$("#elementId").addClass("myClass");
```

Removing a class for an element:

```
$("#elementId").removeClass("myClass");
```

Adding and removing class associations for an element does not overwrite other class associations for that element.

Changing CSS Properties

- Change one property:

```
$("#textBoxId").css("background-color", "red");
```

- Using a variable:

```
var myVar = "red";
$("#textBoxId").css("background-color", myVar);
```

- Change multiple properties:

```
$("#textBoxId").css({
      "background-color": "red",
       "font-size" : "20px"
});
```

Using `.each()` And `this`

```
$("li").each(function() {
      var idValue = $(this).attr("id");
      $(this).text("The id for this list item is: " +
idValue);
});
```

Anonymous Functions

- Used to wrap multiple statements that you want to execute inside a jQuery selection.

- Or be used in an event listener to indicate what should happen when the event triggers.

Operator Chaining

Multiple operators can be chained to the same selector, but separated by “dots”.

The operators execute in the order from left to right following one another.

```
$("#firstId").removeClass("Aclass").addClass("Bclass");
```

```
$("#secondId").text(…).css(…).…;
```

Handling Events with jQuery

What is an Event?

Events are the browser's way of saying, "Hey, this just happened."

Event examples:

- Click an element with the mouse
- **The cursor hover over an element**
- Move from one field to another in a form
- Load a new webpage

Click Me!!

What is an Event?

When an event **fires**, your script can then react by running code (e.g. a function)
By running code when an event fires, your website responds to the user's actions.

It becomes interactive.

How Events Trigger JavaScript Code

1

Select the **element** node(s) the script should respond to

2

Indicate the **event** on the selected node(s) that will trigger a response

3

State the code you want to run when the event occurs

jQuery Event Listeners – Generic Approach

- Single element assignment:

```
$("#buttonOne").on("click", function() {
    $("#outputField").text("You pressed button One");
});
```

- Element selection
- Event
- Anonymous function
- Function body with statements to execute when event happens

jQuery Event Listeners – Generic Approach

- Multiple element assignment:

```
$("li").on("click", function() {
      var itemId = $(this).attr("id");
      $("#outputFieldId").text("You pressed the list item
with id: " + itemId);
});
```

Event Types

Mouse Events	Triggers when…
click	You click on an element
dblclick	You double-click on an element
mouseenter	The mouse cursor enters an element
mouseleave	The mouse cursor leave an element

Keyboard Events	Triggers when…
keydown	A key is on its way down
keypress	A key is fully pressed down
keyup	A key is released

Document/Window Events	Triggers when…
resize	An element is resized
scroll	You scroll inside an element
focus	An element receives focus
blur	An element loses focus

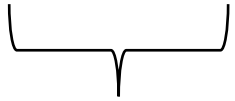
Event Functions

Single event listener assignment:

```
$("#buttonOne").click(function() {
	$("#outputField").text("You pressed button One");
});

$("#fieldOne").mouseenter(function() {
	$("#outputField").text("The mouse entered field One");
});
```

The Event Object

```
$("#buttonOne").click(function(event) {…
});
```



The Event Object

```
$("#textField").keypressed(function(event) {
  var keyValue = event.which;
  …
});
```

jQuery - Animation and Effects

Hide an Element

- `.hide()` Hide a previously shown element.
- `.show()` Show a previously hidden element.
- `.toggle()` Toggle between hidden and shown.

Fade an Element

- `.fadeOut()` Fades out a previously shown element.
- `.fadeIn()` Fades in a previously hidden element.
- `.fadeToggle()` Toggle between fading in and out.

Slide an Element

- `.slideUp()` Hide a previously shown element with a slide motion.
- `.slideDown()` Show a previously hidden element with a slide motion.
- `.slideToggle()` Toggle between sliding up and down.

Animate

- `.animate()` Change CCS properties of an element in a smooth transition.

Example: `$("#boxId").animate({height: 100, width: 100});`

All optional arguments:
`.animate({…}, speed, easing, callback)`

- *speed* transition time in milliseconds or `"slow"`/`"fast"`.
- *easing* `"linear"` or `"swing"` (default).
- *callback* a function call when the animation is complete.

Animate – Moving Elements

Important note!: To move elements using `.animate()`, they need to have their CSS `position` property set to `"relative"`, `"fixed"` or `"absolute"`!

NOT `"static"` which is the default value.

Example: `$("#boxId").animate({left: 200, top: 50});`

Element Position

- `.offset()` Element coordinates (in pixels) relative to the top left corner of the document (the whole webpage).
- `.position()` Element coordinates (in pixels) relative to the top left corner of any parent element. If no parent is present, it works equal to `.offset()`.

To get the coordinates:

- `.offset().left` Gets horizontal coordinate.
- `.offset().top` Gets vertical coordinate.

To set the coordinates:

- `.offset({left: x, top: y})`

Moving From Current Position

Example:

```
var posX = $("#boxId").offset().left + 200;
var posY = $("#boxId").offset().top + 50;
$("#boxId").animate({left: posX, top: posY});
```

Same, but shorter notation:

```
$("#boxId").animate({left: "+=200", top: "+=50"});
```

Keeping Inside the Frame

- `.height()` Height of an element (not incl. margin, border and padding).
- `.width()` Width of an element (not incl. margin, border and padding).

In particular:

- `$(document).height();` Full height of the web page.
- `$(window).height();` Height of web page visible on screen.

Animation Timing

```
$("#boxId").animate({left: 200, top: 100});
```

```
$("#boxId").animate({left: 200}).animate({top: 100});
```

Animation Timing

```
$("#boxId").animate({left: 200}).delay(1000).animate({top: 100});
```

```
$("#boxId").animate({left: 200}).animate({top: 100}).fadeOut();
```

Interrupting an Animation

To stop animation on an element:

```
$("#boxId").stop();
```

Example:

```
$("#stopButtonId").click(function(){
  $("#boxId").stop().animate({height: 100, width: 100});
});
```

To stop ALL queued animation on an element:

```
$("#boxId").stop(true);
```

The Callback Function

```
$("#boxId").animate({left: 200, top: 100}, doSomethingElse());
```

The doSomethingElse function is ensured to be executed after the animation has finished, except if the animation is stopped.

```
function myLoopedFunction(idRef) {
  $(idRef).animate({left: "+=200"},function(){
    myLoopedFunction(idRef);
  });
}
```

The Remainder of the Day

- **Todays Exercise and homework for next session**:
  - Working with jQuery

Responsive Web Design

VIA University College

Today’s topics

- XML
  - Syntax
  - XML DOM
  - childNodes
  - Display XML content on website
- AJAX
  - XMLHttpRequest object
    - Request
    - Response
- Exercises

SEP1: Your website needs to be able to do import an external XML file

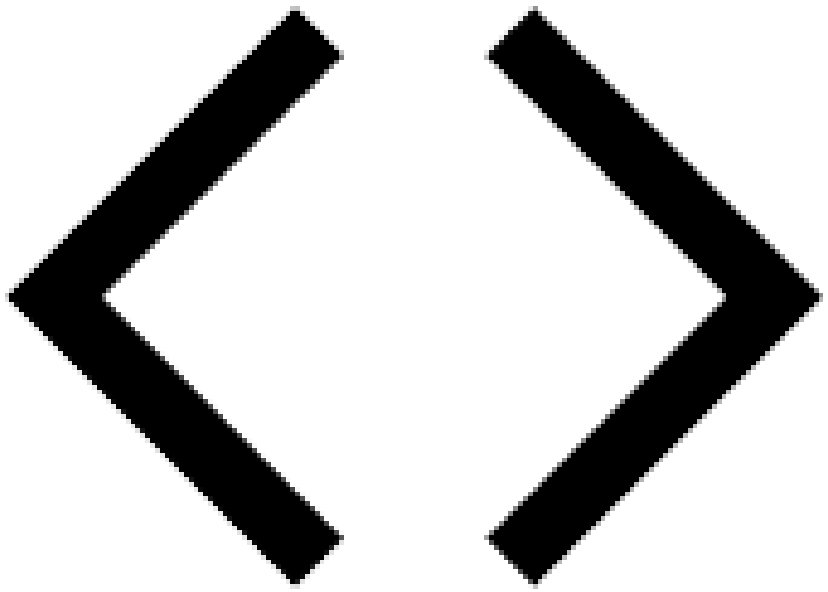
What is XML?

- XML stands for eXtensible Markup Language.
- XML was designed to store and transport data.
- XML plays an important role in many different IT systems.
- XML is often used for distributing data over the Internet.

- XML in itself does not DO anything – it’s just a format.
  - XML is just information wrapped in tags
  - Someone must write a piece of software to send, receive, store, or display it

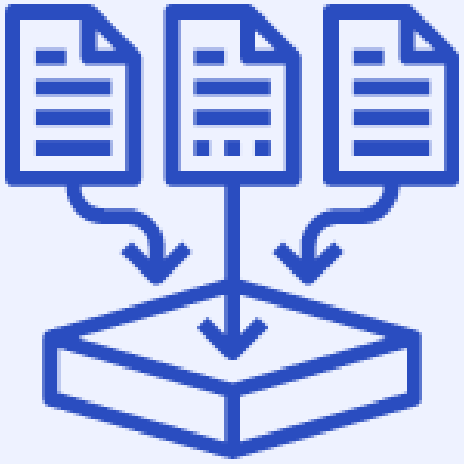
It lets you define your own tags, the order in which they occur, and how they should be processed or displayed.

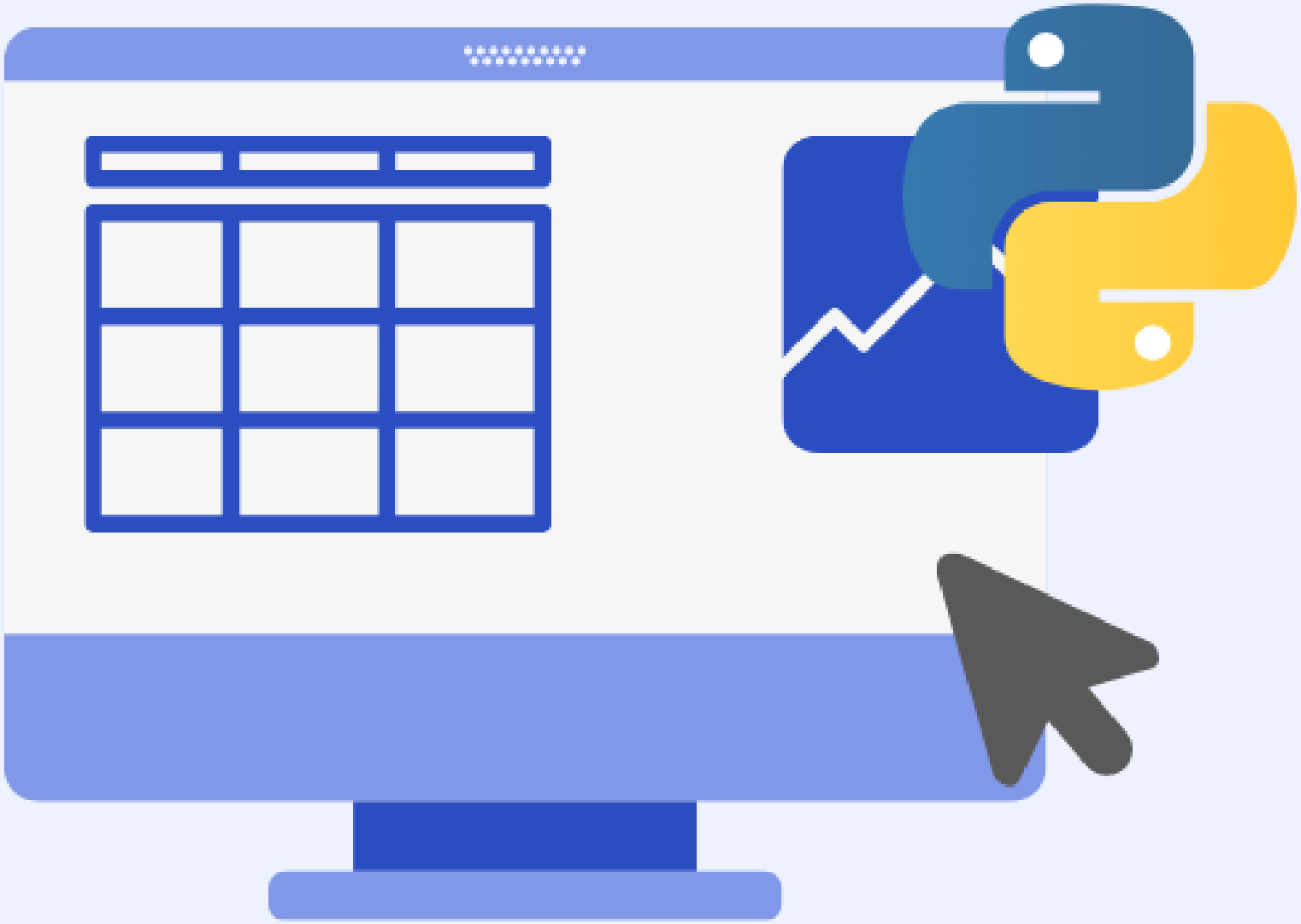
XML is defined with tags just like HTML



XML was designed to carry data

HTML was designed to display data





XML is a compliment to HTML

What’s in an XML document?

- Tags
- Attributes

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

XML syntax

- The XML Prolog (optional)

- One **root** element that is the **parent** of all other elements:

```
<root>
  <child>
 <subchild>.....</subchild>
  </child>
</root>
```

XML example 1

```
<?xml version="1.0" encoding="UTF-8"?>
    <note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
    </note>
```

XML example 2

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
	<person gender="female">
		<firstname>Anna</firstname>
		<lastname>Smith</lastname>
		<birthday>1/1-2000</birthday>
		<email>AnnaSmith@mail.com</email>
	</person>
	<person gender="male">
		<firstname>John</firstname>
		<lastname>Doe</lastname>
		<birthday>9/9-1999</birthday>
		<email>johnDoe@mail.com</email>
	</person>
…
</root>
```

XML example 2.1

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
	<person gender="female">
		<firstname>Anna</firstname>
		<lastname>Smith</lastname>
		<birthday>
			<day>1</day>
			<month>January</month>
			<year>2000</year>
		</birthday>
		<email>AnnaSmith@mail.com</email>
	</person>
…
</root>
```

XML naming rules

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

See more here https://www.w3schools.com/xml/xml_elements.asp

XML attributes

- XML elements can have attributes, just like HTML.
- Attributes are designed to contain data related to a specific element.
- There are no rules about when to use attributes or when to use elements in XML.

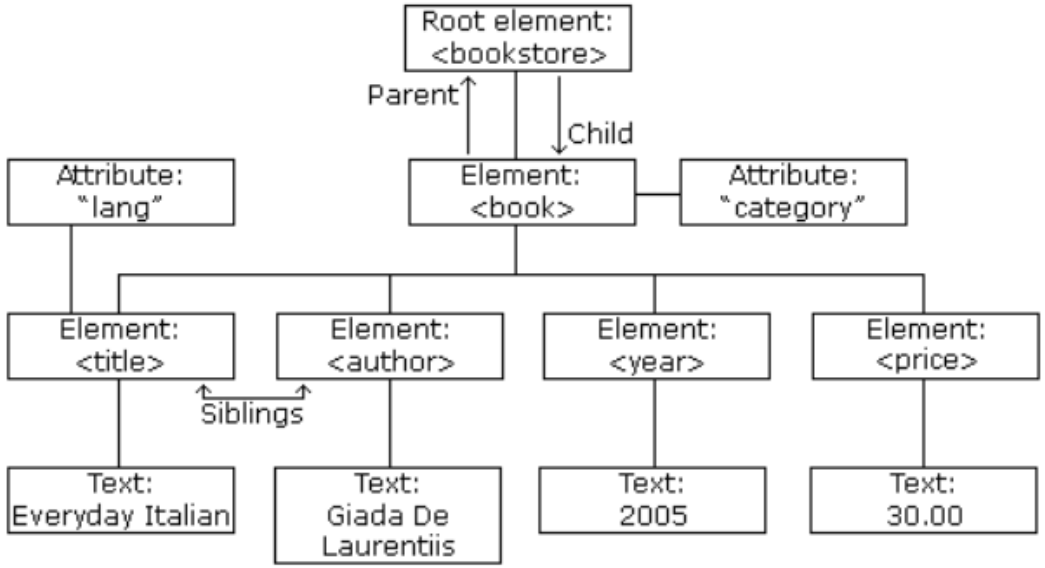
- Metadata (data about data) should be stored as attributes, and the data itself should be stored as elements
  - Such as id or class names used for identifying the elements

XML code and tree structure

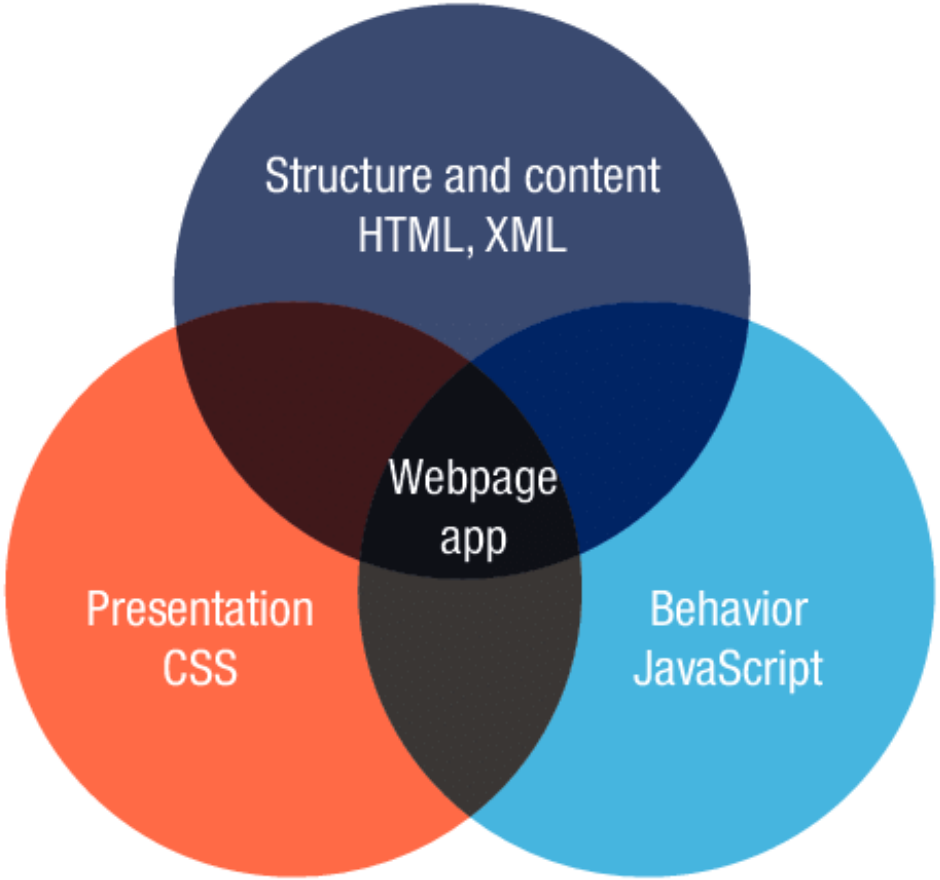
XML code

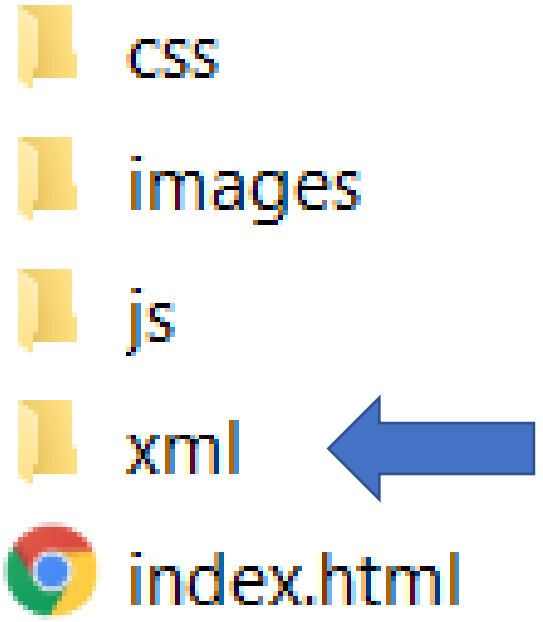
```
<?xml version="1.0" encoding="UTF-8"?>
  <bookstore>
      <book category="cooking">
          <title lang="en">Everyday Italian</title>
          <author>Giada De Laurentiis</author>
          <year>2005</year>
          <price>30.00</price>
      </book>
…
…
  </bookstore>
```

XML tree structure



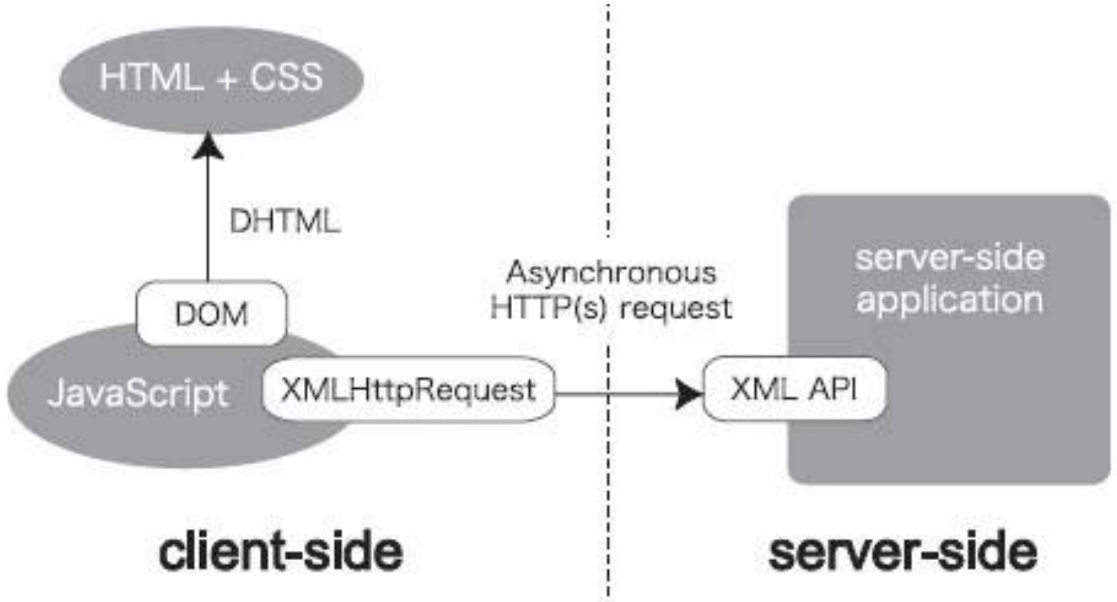
Web page structure on web server





Get content from server

- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.

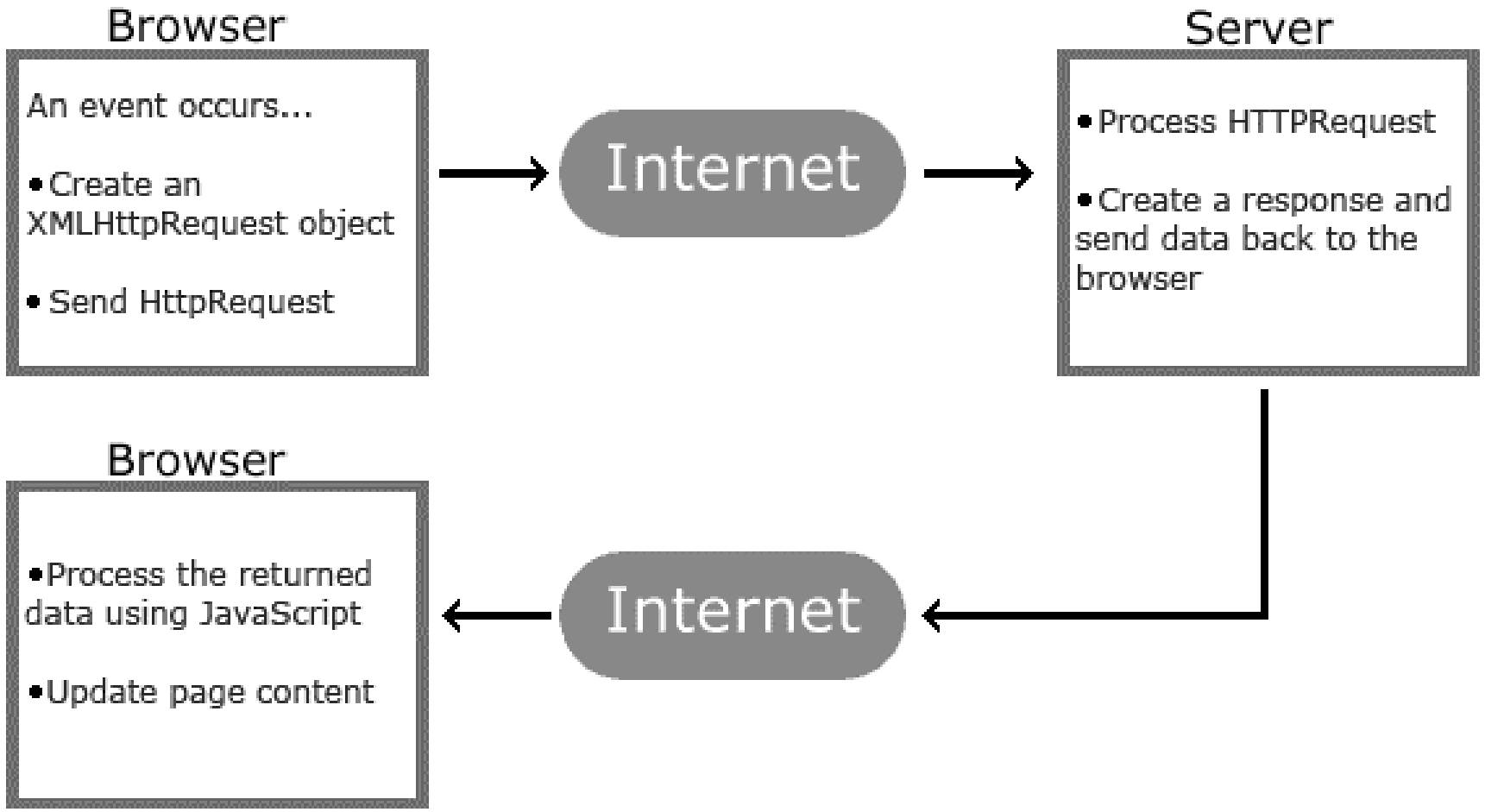


What is AJAX?

- AJAX is **NOT** a programming language.
- AJAX stands for Asynchronous JavaScript And XML.

- AJAX is a technique for accessing web servers from a web page.
  - Update a web page without reloading the page
  - Request data from a server - after the page has loaded
  - Receive data from a server - after the page has loaded
  - Send data to a server - in the background

How does AJAX work?

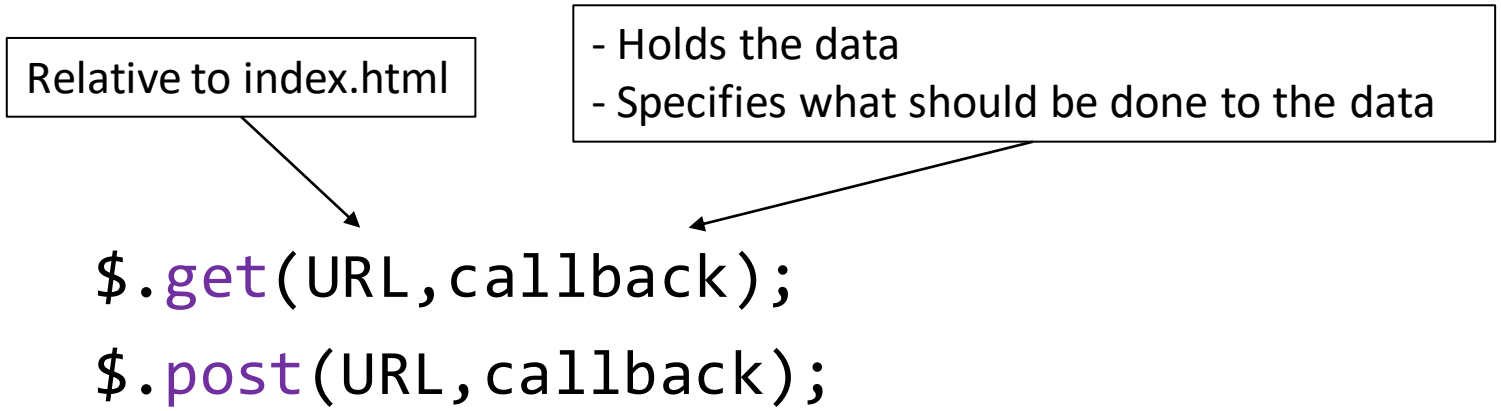


Without jQuery

AJAX coding can be a bit tricky!

HTTP Request: GET vs. POST

- Request-response between a client and server
  - **GET** - Requests data from a specified resource
  - **POST** - Submits data to be processed to a specified resource



All jQuery AJAX methods: https://www.w3schools.com/jquery/jquery_ref_ajax.asp

GET/POST callback example

HTML

```
<!DOCTYPE html>
<html>

<head>
    <script src="js/jquery-3.6.0.min.js"></script>

</head>

<body>
    <p id="textField1">Text goes here</p>
    <p id="textField2"></p>
    <script src="js/script.js"></script>

</body>

</html>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <person gender="female">
    <firstname>Anna</firstname>
    <lastname>Smith</lastname>
  </person>
  <person gender="male">
    <firstname>John</firstname>
    <lastname>Doe</lastname>
  </person>

</root>
```

GET/POST callback example

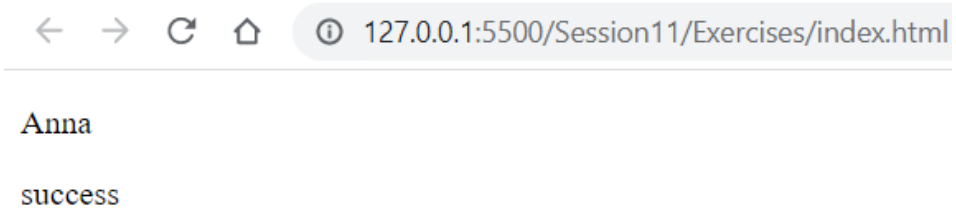
JavaScript/jQuery

```
$.get("xml/person.xml",function (xml, status) {
    var firstname = $(xml).find("firstname");
    $("#textField1").html(firstname[0]);
    $("#textField2").html(status);
});
```

The data

The status of the HTTP request

Result in browser

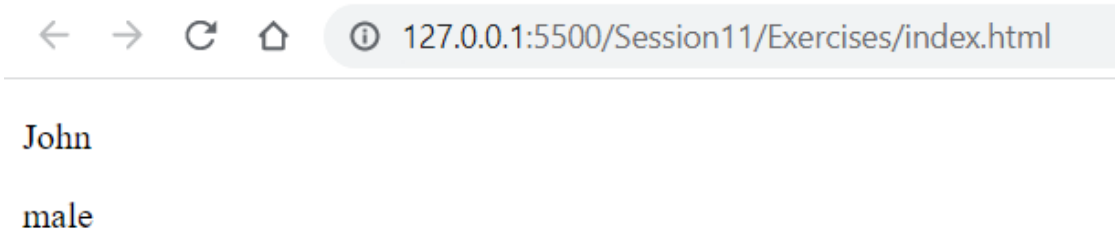


Getting and displaying attributes

JavaScript/jQuery

```
$.get("xml/person.xml",function (xml, status) {
    var firstname = $(xml).find("firstname");
    $("#textField1").html(firstname[1]);
    var persons = $(xml).find("person");
    var gender = $(persons[1]).attr("gender");
    $("#textField2").html(gender);
});
```

Result in browser

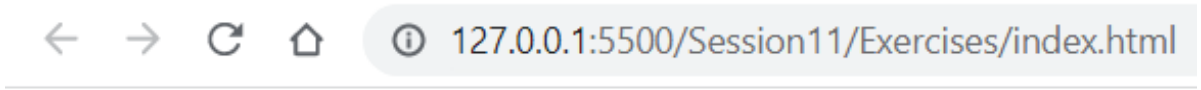


Looping through data in XML files

JavaScript/jQuery

```
$.get("xml/person.xml", function (xml, status) {
    txt = "";
    $(xml).find("person").each(function() {
        var firstname = $(this).find("firstname").text();
        var lastname = $(this).find("lastname").text();
        txt += "firstname:" + firstname + "; " + "Lastname:" + lastname + "<br>";
    });
    $("#textField1").html(txt);
});
```

Result in the browser window



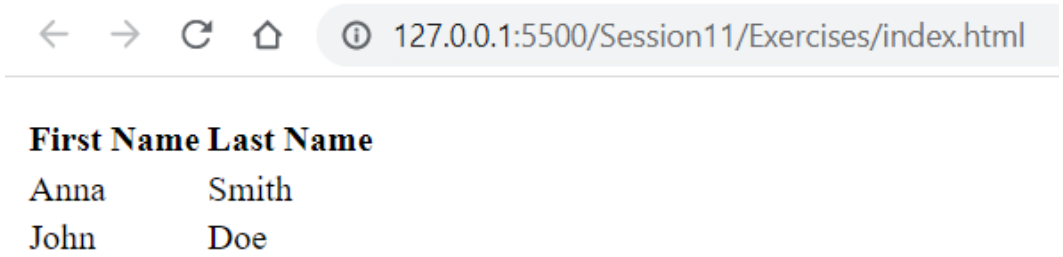
firstname:Anna; Lastname:Smith
firstname:John; Lastname:Doe

Displaying XML data in an HTML table

JavaScript/jQuery

```
$.get("xml/person.xml", function (xml, status) {
    txt = "<table><tr><th>First Name</th><th>Last Name</th></tr>";
    $(xml).find("person").each(function() {
        var firstname = $(this).find("firstname").text();
        var lastname = $(this).find("lastname").text();
        txt += "<tr>" + "<td>" + firstname + "</td>" + "<td>" + lastname + "</td>";
    });
    txt += "</table>";
    $("#textField1").html(txt);
});
```

Result in the browser window



Displaying XML data in an HTML table: Styling

Method 1

CSS:

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
```

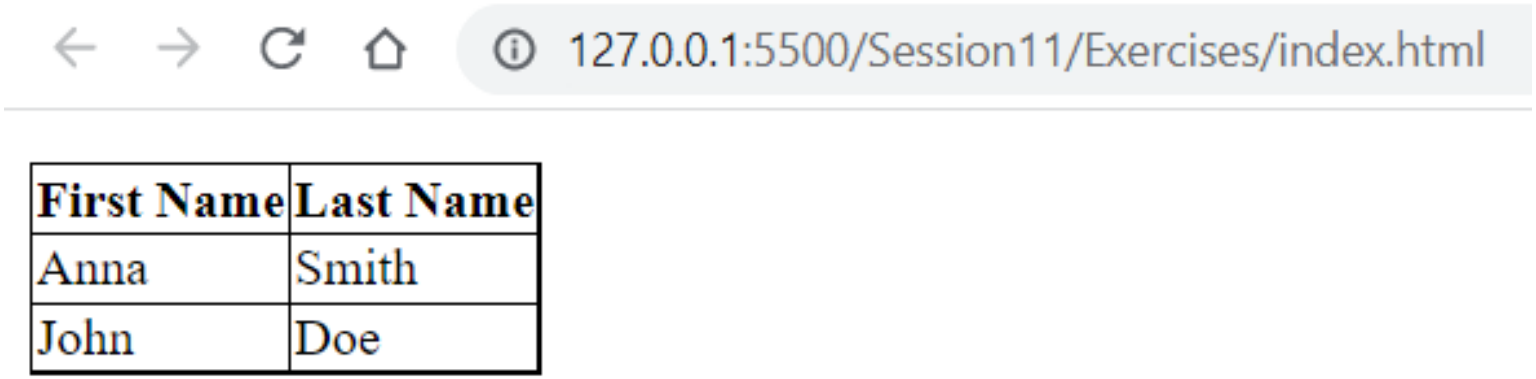
Method 2

CSS:

```
.myClass {
    border: 1px solid black;
    border-collapse: collapse;
}
```

```
$.get("xml/person.xml", function (xml, status) {
    txt = "<table class='myClass'><tr><th class='myClass'>First Name</th><th class='myClass'>Last Name</th></tr>";
    $(xml).find("person").each(function() {
        var firstname = $(this).find("firstname").text();
        var lastname = $(this).find("lastname").text();
        txt += "<tr>" + "<td class='myClass'>" + firstname + "</td>" + "<td class='myClass'>" + lastname + "</td>";
    });
    txt += "</table>";
    $("#textField1").html(txt);
});
```

The result in the browser window



Exercises (no learning path today)

- Find the exercises on itslearning
  - Working with XML files
  - Displaying content from XML files on a webpage
  - Retrieving an external XML file, processing it, and displaying the contents on a webpage
  - Just ask me if you get stuck ☺

- Note: you can only see the result of the exercises if you use the ”Live Server” extension in Visual Studio Code