Information Visualization

Course Module CI6221

Visualization Tools Fundamental

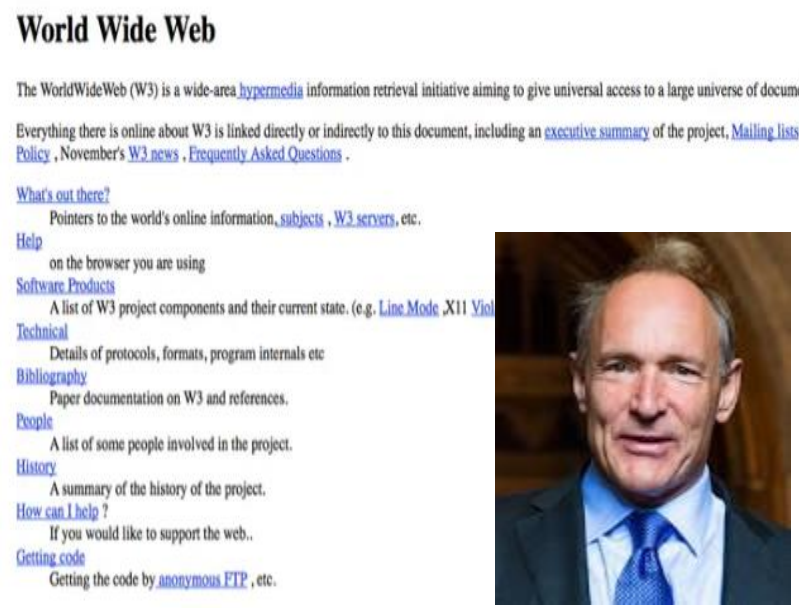
WKW School of Communication and Information,
NTU

HTML - Introduction

- **HTML** stands for **Hypertext Markup Language** - allows the creation and structure of sections, paragraphs, headings, links, and blockquotes for web pages (first written in **1993** by **Sir Tim Berners-Lee**)

- Sir Tim Berners-Lee invented the **World Wide Web** in 1989 – while working at **CERN** (***Conseil Européen pour la Recherche Nucléaire***).

  The World Wide Web was initially created to **meet the demand for automated information-sharing** between scientists in universities and institutes around the world



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viol
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

CERN – research institute with world's biggest particle accelerators, where monumental discoveries such as the detection of the once-elusive Higgs boson particle (2012)

HTML - Hypertext Markup Language

HTML - A computer language that consists **of easily understood keywords, names, and tags that help format the overall view** of a page and the data it contains.

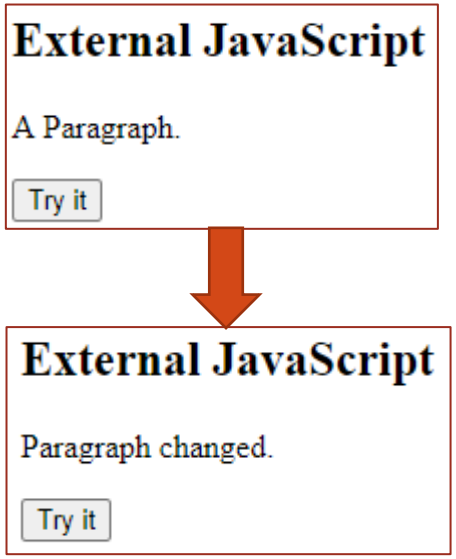
```
<!DOCTYPE html>
<html>
<body>

<h2>External JavaScript</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<script src="myScript.js"></script>

</body>
</html>
```

Document type declaration

HTML

"myScript.js"

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

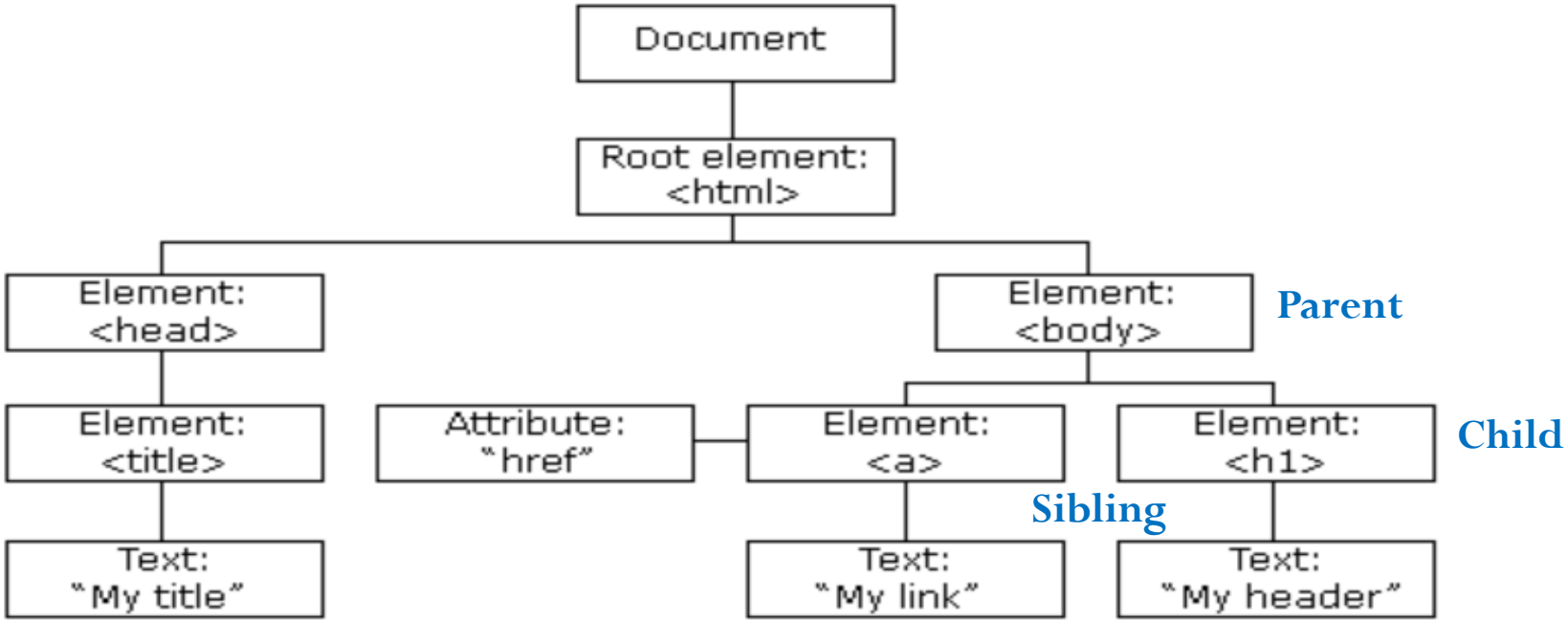


JavaScript is a scripting or programming language that allows you to implement complex features on web pages

HTML - DOM

- **Document Object Model** refers to the **hierarchical** **structure** of HTML- the standard **object model** and **programming interface** for HTML. Defines elements as **objects** and their **properties**.

The HTML DOM Tree of Objects



HTML - DOM

Examples of **single tag elements**: **
** - **line break**, **<hr>** - **thematic break** (horizontal rule), **<img>** - **insert image**

- Each pair of **bracketed tags** (or, in some cases, a **single tag**) is the **element**, and refer to elements' **relationships** to each other as: **parent**, **child**, **sibling**, **ancestor**, and **descendant**.

```
<html>
  <body>
    <h1>Breaking News</h1>
    <p></p>
  </body>
</html>
```

parent → <body>

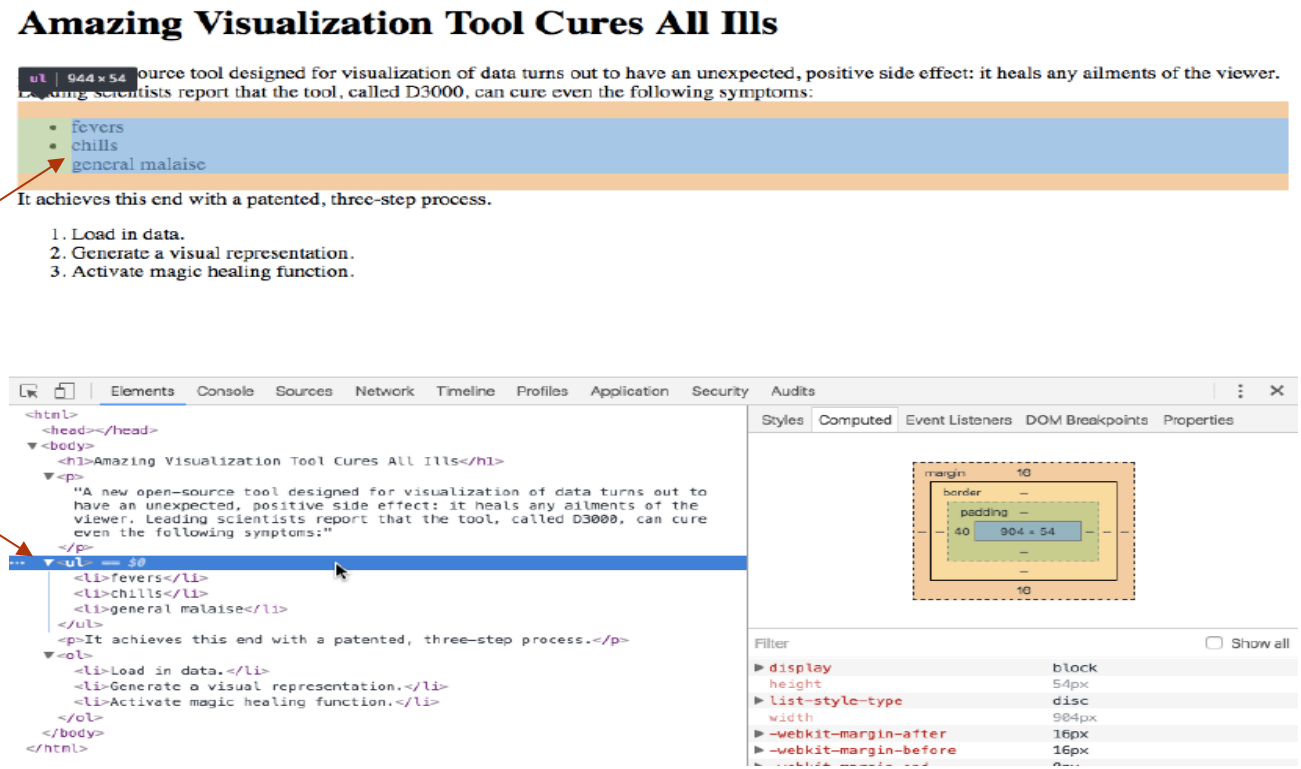
siblings → <h1>Breaking News</h1>,

<body> is the **parent** element to both of its **children**, **<h1>** and ** ** (which are **siblings** to each other). All elements on the page are **descendants** of **html**.

- Web browsers **parse the DOM** to make sense of a page's content. As coders building visualizations, we use DOM to **navigate** its **hierarchy** to apply **styles** and **actions** to its **elements**.

Rendering and the Box Model

- **Rendering** is the process by which browsers, after parsing the HTML and generating the DOM, **apply visual rules to the DOM contents** and **draw those pixels** to the screen.
- When rendering content: to a browser, **everything is a box**. **Paragraphs**, **divs**, **spans** – two dimensional rectangles, with properties such as width, height, and positions in space.



Box associated with that element is highlighted in blue when moused over.

ul => unordered list
li => list item

Viewing the source code (F12)

Open the **1_style_div.html** file

This is a heading

This is a paragraph.

Elements Console Sources Network Perforn

```
<!DOCTYPE html>
<html>
▼<head> == $0
  <style> h1 {color:red;} p {color:blue;} </style>
 </head>
▼<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
 </body>
</html>
```

Key Value

Style element

HTML Introduction - Elements

In HTML, **tags** define the start and end of headings, paragraphs, lists, character highlighting and links. HTML documents can be edited with a **text editor**.

Most HTML **elements** are identified with a **start tag** <a href = "https://www.abc.com"> which gives the **element name** and **attributes**, followed by the **content**, and the **end tag**.

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head> metadata	Contains metadata/information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr> horizontal rule - draws a line	Defines a thematic change in the content
<!--...-->	Defines a comment

href – hypertext reference

```
<!DOCTYPE html>
<html>
<body>

<h1>The a element</h1>

<a href="https://www.w3schools.com">Visit W3Schools.com!</a>

</body>
</html>
```

Names are **not case sensitive**. E.g., H1 is same as h1.

Attribute

End tag

Content

Element name 'a' – anchor tag

The a element

Visit W3Schools.com!

Reference list of HTML elements: https://www.w3schools.com/tags/default.asp

HTML – Anchor Element

The HTML **<a>** anchor tag defines a **hyperlink** - click on the link and jump to the referenced document.

When moving the mouse over a link, the cursor will turn into a little hand.

```
<!DOCTYPE html>
<html>
<body>

<h1>HTML Links</h1>

<p><a href="span.html">Click to open span.html</a></p>

</body>
</html>
```

Anchor text

Points to linked document

HTML Links

Click to open span.html

The span element

The next word will show blue and the next word will show dark green as the colored words are tagged between span elements

HTML – Span Element

The **** tag is an inline container to mark up **part of a text/document** =>
** ** is a **block-level** element while **** is an **inline** element.

 标签是一个内联容器，用于标记文本/文档的一部分 =>
 是一个块级元素，而 是一个内联元素。

```
<!DOCTYPE html>
<html>
<body>

<h1>The span element</h1>

<p>The next word will show <span style="color:blue;font-weight:bold">blue</span> and the next word will show
<span style="color:darkolivegreen;font-weight:bold">dark green</span> as the colored words are tagged between
span elements</p>

</body>
</html>
```

The span element

The next word will show **blue** and the next word will show **dark green** as the colored words are tagged between span elements

HTML – Style Element

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {color:red;}
p {color:blue;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

The HTML **style element's attributes** are used to **add styles** to an element, such as color, font, size, and more - Related to **Cascading Style Sheet (CSS)**

This is a heading

This is a paragraph.

HTML – Division Element

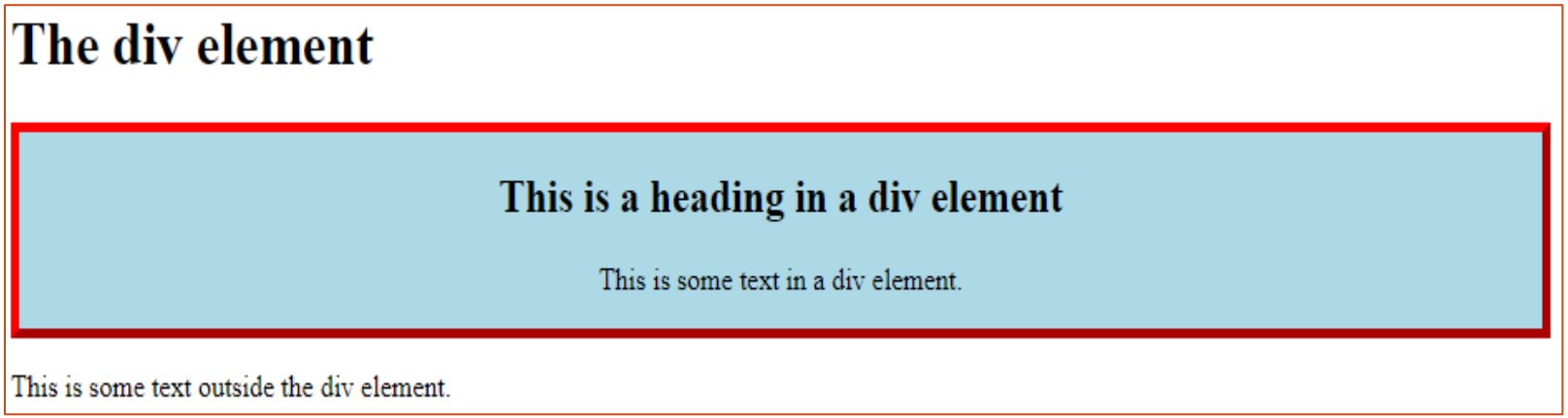
```html
<!DOCTYPE html>
<html>
<head>
<style>
.myDiv {
  border: 5px outset red;
  background-color: lightblue;
  text-align: center;
}
</style>
</head>
<body>

<h1>The div element</h1>

<div class="myDiv">
  <h2>This is a heading in a div element</h2>
  <p>This is some text in a div element.</p>
</div>

<p>This is some text outside the div element.</p>

</body>
</html>
```

The **class attribute myDiv** (**referenced by a dot in the style element**) points to the specified class in HTML

<style> element apply a simple style sheet to the **myDiv class**

The **class attribute** specifies one or more classnames (**separated between a space**) for an element.

The ** ** tag defines a **division or section**.
- used as a **container for HTML elements** - which is then styled via the **myDiv class**



HTML - Attributes

- In a **start tag**, **white space** and **attributes** are allowed between the **element name** and the **closing delimiter**.

- The **value of the attribute** may be either:

  1. A **string literal**, delimited by **single** quotes or **double** quotes

  2. A **name token** (a sequence of letters, digits, periods, or hyphens – assigned for **certain attributes**)

  <a NAME="alt5">some text

```
<!DOCTYPE html>
<html>
<body>

<h1>The a element</h1>

<a href="https://www.w3schools.com">Visit W3Schools.com!</a>

</body>
</html>
```

Closing delimiter

An **attribute** consists of an **attribute name**, an **equal sign**, and a **value**. White space is allowed around the equal sign.

For example, **a** is the **element name**, **href** (hypertext reference) is the **attribute name**, **http://host/dir/file.html** is the **attribute value**:

<a HREF="http://host/dir/file.html">

HTML – Attributes (ID/Class)

The **id attribute** is a unique identifier to **specify element** - used to perform a certain task for a **unique (only one) element**.

```
<!DOCTYPE html>
<html>

<head>
    <title>
        HTML id attribute
    </title>

    <style>
        #geeks{
            color:green;
            font-size:25px;
        }
    </style>
</head>

<body style="text-align:center">
    <h1>Geeks for Geeks</h1>
    <p id="geeks">Welcome to Geeks for Geeks</p>

<p>A Computer Science portal for geeks</p>

</body>

</html>
```

In CSS, the **id attribute** is identified using **# symbol followed by the id**

id = "geeks" identify CSS attributes for the **p** element

Geeks for Geeks

Welcome to Geeks for Geeks

A Computer Science portal for geeks

HTML – Attributes (ID/Class)

The **class attribute** identifies the style for (**multi**) **elements** with the specified **class name**. **ID attribute** only for a **single** element.

```
<html>
<head>
    <style>
        .country {
            background-color: black;
            color: white;
            padding: 8px;
        }
    </style>
</head>
<body>
    <h2 class="country">CHINA</h2>
<p>China has the largest population
    in the world.</p>
    <h2 class="country">INDIA</h2>
<p>India has the second largest
    population in the world.</p>
    <h2 class="country">UNITED STATES</h2>
<p>United States has the third largest
    population in the world.</p>
</body>
</html>
```

CHINA

China has the largest population in the world.

INDIA

India has the second largest population in the world.

UNITED STATES

United States has the third largest population in the world.

HTML – Attributes (ID/Class)

We can identify element with a **combination of ID and Class**

HTML

<h1 id="one" class="two">this should be red</h1>

ID .CLASS

CSS

#one{color:red;}
.two{color:blue;}

The ID and Class can have different styles of their own => **ID has higher specificity (ID style will be applied)**

An element can be identified with **multiple classes**.

HTML

<h1 class="three four">color depends on CSS order</h1>

takes priority

The classes can have different styles of their own => order given in HTML does not matter. **Class specified later in Cascading Style Sheet (CSS) takes priority**.

HTML - Attributes

- All HTML elements can have **attributes**
  - **href** attribute of <a> specifies the **URL** of the page the link goes to
  - **src** attribute of <img> specifies the **path to the image** to be displayed
  - **width** and **height** attributes of <img> provide **size** information for images
  - **alt** attribute of <img> provides an **alternate text** for an image
  - **style** attribute adds styles to an element, such as **color**, **font**, **size**, and more
  - **lang** attribute of the <html> tag declares the **language** of the Web page
  - **title** attribute defines **main information** about an element

Reference list of HTML Attributes: https://www.w3schools.com/tags/ref_attributes.asp

Class Exercise –HTML

- Open any text editor – e.g., Notepad++

- Go to the 1_HTML_Lab folder and edit the codes in the following HTML files to make them work properly:
  - 1_style_div.html
  - 2_span.html
  - 3_hyperlink.html

```html
<!DOCTYPE html>
<html>
<head>
<style>
h1 {:red;}
p {:blue;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
h1 {color: black;}
p {color: red;}
</style>
</head>
<body>

<h1>This is a test heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```html
<h1>The span element</h1>

<p>The next word will show <span style="color: blue;font-weight:bold">blue <!-- </span> --> and the next word will show <span style="c

</body>
</html>
```

Remove comment tag <!-- -->

```html
<!DOCTYPE html>
<html>
<body>

<h1>HTML Links</h1>

<p>< href="2_span.html">Click to open the span html file</a></p>

</body>
</html>
```

Add starting “a” anchor element tag

Cascading Style Sheets (CSS)

- CSS are used to **style the visual presentation** of the elements:

```
<style type="text/css">
body {
      background-color: powderblue;
      color: black;
      }
h1    {color: blue;}
p     {color: red;}
</style>
```

```
<body>This is a body</body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

This is a body

This is a heading

This is a paragraph.

CSS styles consist of **selectors** and **properties**.

Selectors are followed by properties, grouped in **curly brackets**.

A **property** and its **value** are **separated by a colon**, and the line is terminated with a **semicolon**.

```
selector {
    property: value;
    property: value;
    property: value;
}
```

Apply the **same properties** to **multiple selectors** by separating the selectors with a comma

```
selectorA,
selectorB,
selectorC {
    property: value;
    property: value;
    property: value;
}
```

CSS

- What does the following CSS specify?

selectors

properties

```
p,
li {
    font-size: 12px;
    line-height: 14px;
    color: orange;
}
```

both p **paragraphs** and li **list** items should use the same font size, line height, and color.

Collectively, this whole chunk of code (selectors and bracketed properties) is called a **CSS rule**.

List of CSS properties - https://www.w3schools.com/cssref/default.asp

CSS – Why called Casading Style Sheets?

- It's because **selector matches** **cascade from the top down**. When more than one selector applies to an element, the **later** rule generally **overrides** the **earlier** one.

- E.g., The rules for p are applied first, but then the rules for p.highlight override the less specific p rules.

```
p {
    color: blue;
}

p.highlight {
    color: black;
    background-color: yellow;
}
```

More specific => overrides p rule

A regular paragraph

A highlighted paragraph

- The rules follow the **specificity** of selector. The **p.highlight** selector overrides the p rule even if it were listed first, because it is **more specific**. If two selectors have the **same specificity**, the **later one** will be applied.

CSS – Selectors

- D3 uses CSS-style **selectors** to **identify elements** on which to operate
- Selectors identify specific elements to which styles will be applied.
  - **Type selectors** - match DOM elements with the **same name**.

```
h1          /* Selects all level 1 headings          */
p           /* Selects all paragraphs                */
strong      /* Selects all strong elements           */
em          /* Selects all em elements               */
div         /* Selects all divs                      */
```

em refer to the font size => 2em means 2 times the size of the current font

  - **Descendant selectors** - match elements that are contained by ("descended from") another element.

```
h1 em       /* Selects em elements contained in an h1 */
div p       /* Selects p elements contained in a div  */
```

CSS – Selectors (Class and ID)

- **Class selectors** - match elements of any type that have been assigned a specific class. Class names are preceded with a **period**.

```
<main class="inline-block-center">

.inline-block-center {
  text-align: center;
}
```

```
.caption    /* Selects elements with class "caption"  */
.label      /* Selects elements with class "label"    */
.axis       /* Selects elements with class "axis"     */
```

- Target elements with **multiple classes** by **stringing** the classes together.

```
.axis.x        /* Could target an x-axis        */
.axis.y        /* Could target a y-axis         */
```

- **.axis** could be used to apply styles to **both axes**, for example, whereas **.axis.x** would apply only to the **x-axis**.

CSS – Selectors (Class and ID)

- **ID selectors** - match **single** element with given ID - preceded with a **hash** mark.

```
#myHeader {
Text-align: center;
}
```

```
#header     /* Selects element with ID "header"     */
#nav        /* Selects element with ID "nav"        */
#export     /* Selects element with ID "export"     */
```

```
<h1 id="myHeader">My Header</h1>
```

Class name can be used by **multiple** HTML elements, **ID** name only used by **one** element within page

CSS – Properties and Values

- Groups of **property**/**value** pairs **cumulatively form the styles**:

Selector

```
{
    margin: 10px;
    padding: 25px;
    background-color: yellow;
    color: pink;
    font-family: Helvetica, Arial, sans-serif;
}
```

- **Colors** can be specified in several different formats, such as:

Named colors: `orange`

Hex values: `#3388aa` or `#38a`

RGB values: `rgb(10, 150, 20)`

RGB with alpha transparency: `rgba(10, 150, 20, 0.5)`

Color Picker: https://www.w3schools.com/colors/colors_picker.asp

Referencing Styles – Internal (Embed)

1) **Embed the CSS in your HTML** – everything will be in one file.

- In the document **head**, include all CSS code **within a style element**.

Specify style type

Property: value;

Selector

```
<html>
    <head>
        <style type="text/css">
            p {
                font-size: 24px;
                font-weight: bold;
                background-color: red;
                color: white;
            }

        </style>
    </head>
    <body>
        <p>If I were to ask you, as a mere paragraph, would you say that I
        have style?</p>
    </body>
</html>
```

If I were to ask you, as a mere paragraph, would you say that I have style?

Referencing Styles – External

2) **Reference external stylesheet from HTML** – CSS saved in a plain-text file with a **.css** suffix e.g., style.css.

- Use a **link element** in the **head element** to reference the external CSS file.

Specifies the **relationship** between the current document and the linked document

Linked document is a stylesheet

Specifies the **path/location** of the linked document
- e.g., shows in same directory as html file

```
<html>
<head>
  <link rel="stylesheet" href="5_CSS_styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS file – 5_CSS_styles.css

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

This is a heading

This is a paragraph.

Referencing Styles – Inline

3) **Attach inline styles** – attach style rules **inline** directly to elements in HTML.

- Add a **style attribute** to any **element**. Then include the CSS rules within the double quotation marks.

Style attribute **CSS rules – attached to element – no need for selectors**

```
<p style="color: blue; font-size: 48px; font-style: italic;">Inline styles
are kind of a hassle</p>
```

Inline styles are kind of a hassle

- Inline styles are attached **directly to elements** => **no need for selectors**.
- **Messy and hard to read**, but useful for giving **special treatment to a single element**, when that style information doesn't make sense in a larger stylesheet.

CSS Specificity

- **Specificity** determines **which CSS rule is applied** . Usually the reason why CSS-rules don't apply to some elements, although you think they should.
- If **two selectors** apply to the **same element**, the one with **higher specificity wins**.
- There are four distinct categories which define the specificity level of a given selector: **inline styles**, **IDs**, **classes**, **attributes**, **and elements**.
- When selectors have an **equal** specificity value, the **latest** rule is the one that counts.
- When selectors have an **unequal** specificity value, the **more specific** rule is the one that counts.
  - **Inline styles** (**e.g., style = "font-weight: bold;"**) => **highest** specificity.
  - **ID** selectors **(e.g., #example)** => **use IDs to increase the specificity**.
  - **Class** selectors (**e.g., .example**) and **Attribute** selectors (**e.g., a[target="blank"]**) have **same** specificity.
  - **Element** selectors (e.g., h1) => **lowest** specificity

CSS Specificity

Attribute selectors (**e.g., a[target="blank"]**)

```
<style>
a[target="_blank"] {
  background-color: yellow;
}
</style>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
```

The link with target="_blank" gets a yellow background:

w3schools.com disney.com wikipedia.org

Equal specificity: the **latest rule** counts

```
<style>
h1 {background-color: yellow;}
h1 {background-color: red;}
</style>
```

This is heading 1

ID selectors have a higher specificity than **class/attribute** selectors => first rule more specific than the other two, and will be applied

Contextual selectors are more specific than a single **id** selector

```
<html>
<head>
<style>
div#a {background-color: green;}
#a {background-color: yellow;}
.class1 {background-color: blue;}
</style>
</head>

<body>
<div class="class1" id="a">This is a div</div>
</body>
```

This is a div

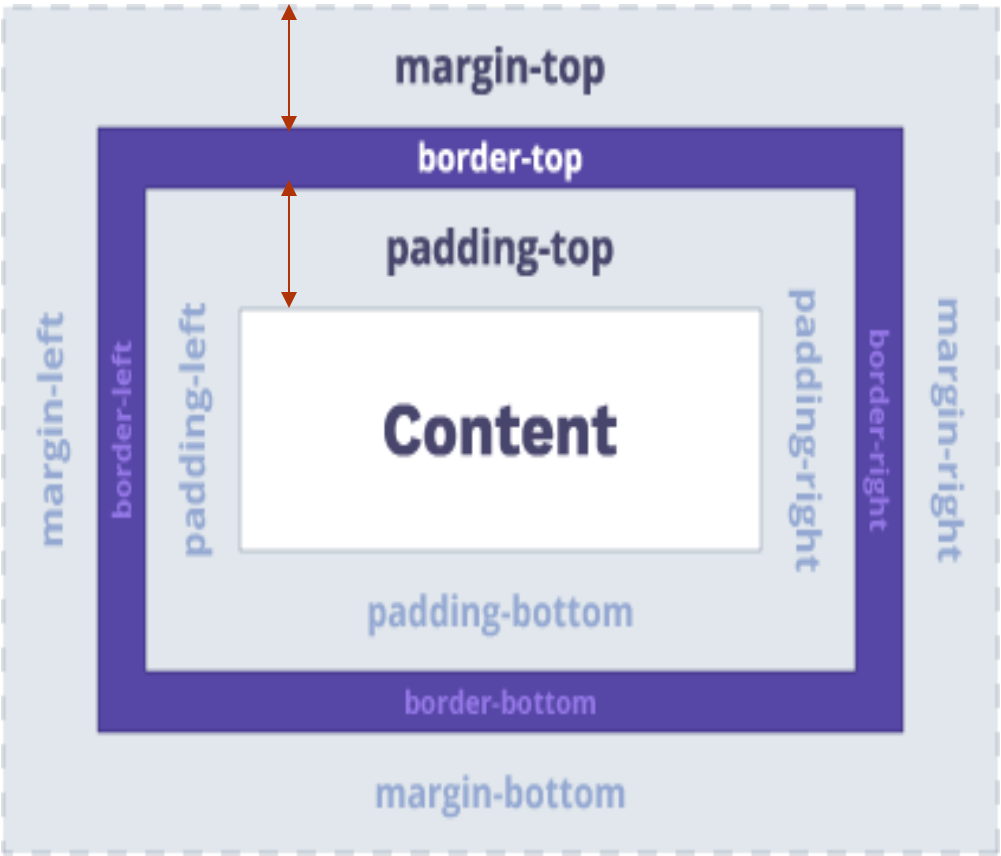
A class selector beats any number of **element selectors** - a class selector such as .intro beats h1, p, div, etc

```
<html>
<head>
<style>
.intro {background-color: yellow;}
h1 {background-color: red;}
</style>
</head>
<body>

<h1 class="intro">This is a heading</h1>

</body>
</html>
```

This is a heading

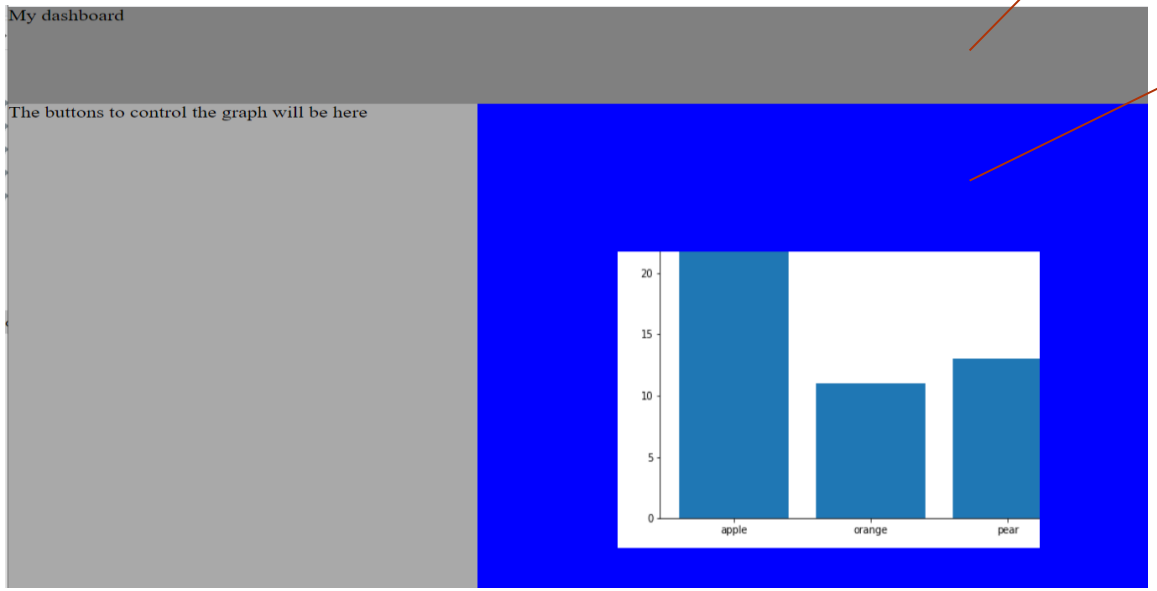
CSS Alignment - Attributes

Margin is the spacing **around** an element's **border**, **Padding** is the spacing **between** the element's **border** and the element's **content**.



Simple Dashboarding using CSS

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Dashboard</title>
        <script type="text/javascript" src="d3.js"></script>
        <link rel="stylesheet" href="style_align.css">
    </head>

<body bgcolor="#170061">
    <div id="topbar">
      My dashboard
    </div>
    <div id="graph_div">
      <img src="fav_fruits.png" alt="CI6221 Class Favorite Fruits" class="img-class">
    </div>
    <div id="panel_div">
      The buttons to control the graph will be here
    </div>
</body>
```



CSS

```css
body, html {
    margin:0;
    padding:0;
    height:100%;
}

#topbar {
    width:100%;
    height:100px;
    background:grey;
    margin:0;
}

#graph_div {
    width:60%;
    height:100%;
    float:right;
    background:blue;
}

#panel_div {
    width:40%;
    float:left;
    background:darkgrey;
    height:100%;
}
```

% of the width of the containing element

100% available (full) width

100 pixel **fixed** height

60% available width

100% available height after topbar 100px

Position to right

40% of width

Position to left

100% available height after topbar 100px

Class Exercise - CSS

- Open any text editor – e.g., Notepad++

- Edit the codes in following HTML files to make them work:
  - 4_CSS_Internal.html
  - 5_CSS_External.html and 5_CSS_styles.css
  - 6_CSS_Inline.html

- Create a simple dashboard using CSS – see files in Dashboard folder

```html
<!DOCTYPE html>
<html>
<head>

body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

1

```html
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

2

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="5_CSS_styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

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

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

</body>
</html>
```

```c
  background-color: powderblue;

h1 {
  color: blue;
}
p {
  color: red;
}
```

3

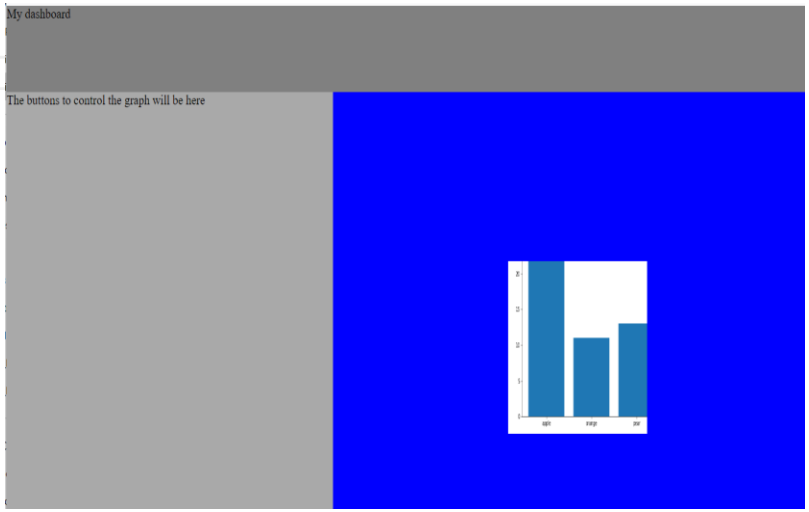
```c
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

4

```html
<!DOCTYPE html>
<html>
<body>

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

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

</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Dashboard</title>
        <script type="text/javascript" src="d3.js"></script>
        <link rel="stylesheet" href="style_align.css">
    </head>

<body bgcolor="#170061">
    <!-- <div id="topbar"> -->
      My dashboard
    </div>
    <!-- <div id="graph_div"> -->
      <img src="fav_fruits.png" alt="IN6221 Class Favorite Fruits" class="img-class">
    </div>
    <!-- <div id="panel_div"> -->
      The buttons to control the graph will be here
    </div>
</body>
```

Remove comments



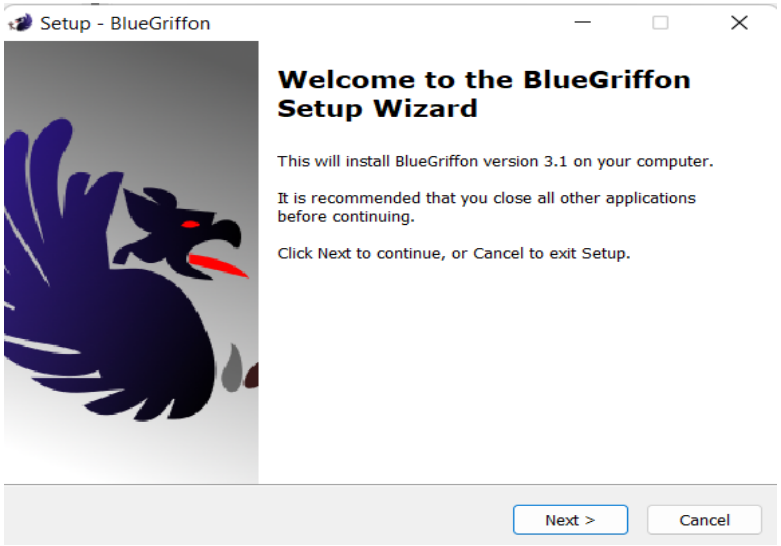

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Dashboard</title>
        <script type="text/javascript" src="d3.js"></script>
        <link rel="stylesheet" href="style_align.css">
    </head>

<body bgcolor="#170061">
    <div id="topbar">
      My dashboard
    </div>
    <div id="graph_div">
      <img src="fav_fruits.png" alt="IN6221 Class Favorite Fruits" class="img-class">
    </div>
    <div id="panel_div">
      The buttons to control the graph will be here
    </div>
</body>
```

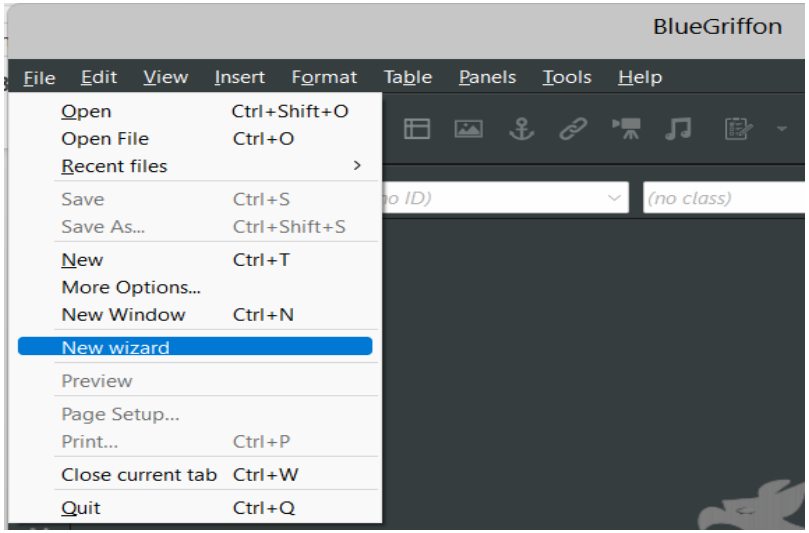
WYSIWYG HTML Editor

- BlueGriffon is an open-source WYSIWYG editor => download from http://bluegriffon.org/

- First create a working folder on your hard drive where all the pages and associated content can reside.

- Create a sample web page using the new document wizard.

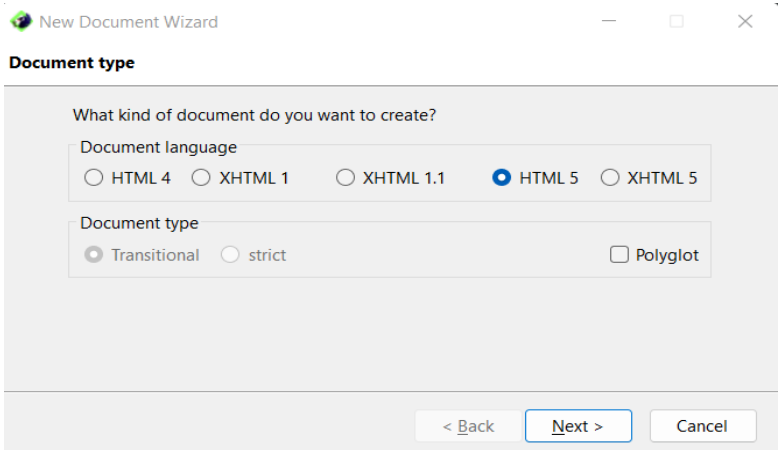


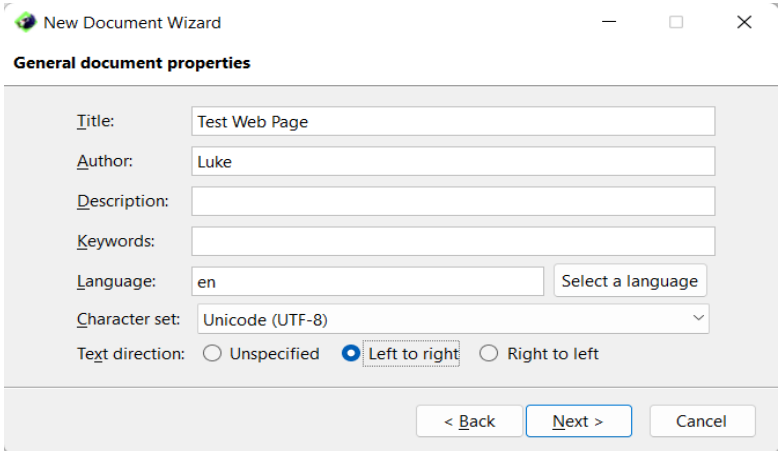


WYSIWYG HTML Editor

- Create a document of the desired HTML dialect.

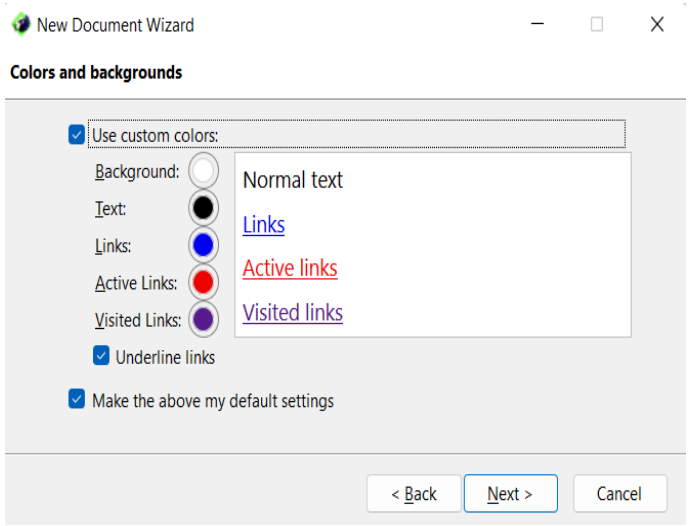
- Next set the main **metadata** of the document: **Title, Author, Description, Keywords, Main Language of document**, **Character** set of document, **Writing direction** of document.



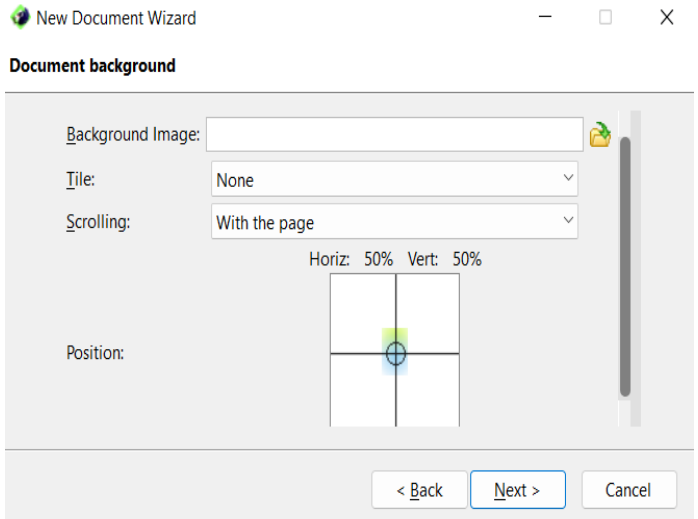


WYSIWYG HTML Editor

- Set the **default colors** of the document: its background color, the foreground (text) color and the color of links. All choices made here will be applied to the document through CSS styles contained in a stylesheet embedded into the new document.



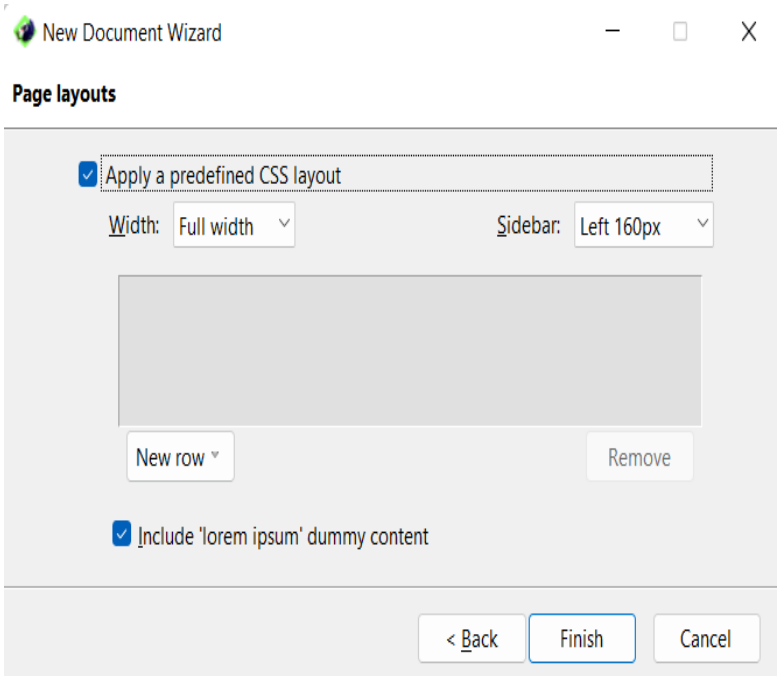
- Apply a **background image** to the whole document through the new page. You can select a background image, define how it will be repeated over the document, say if it should scroll with the document or remain fixed and finally finely set its position in the document's background.



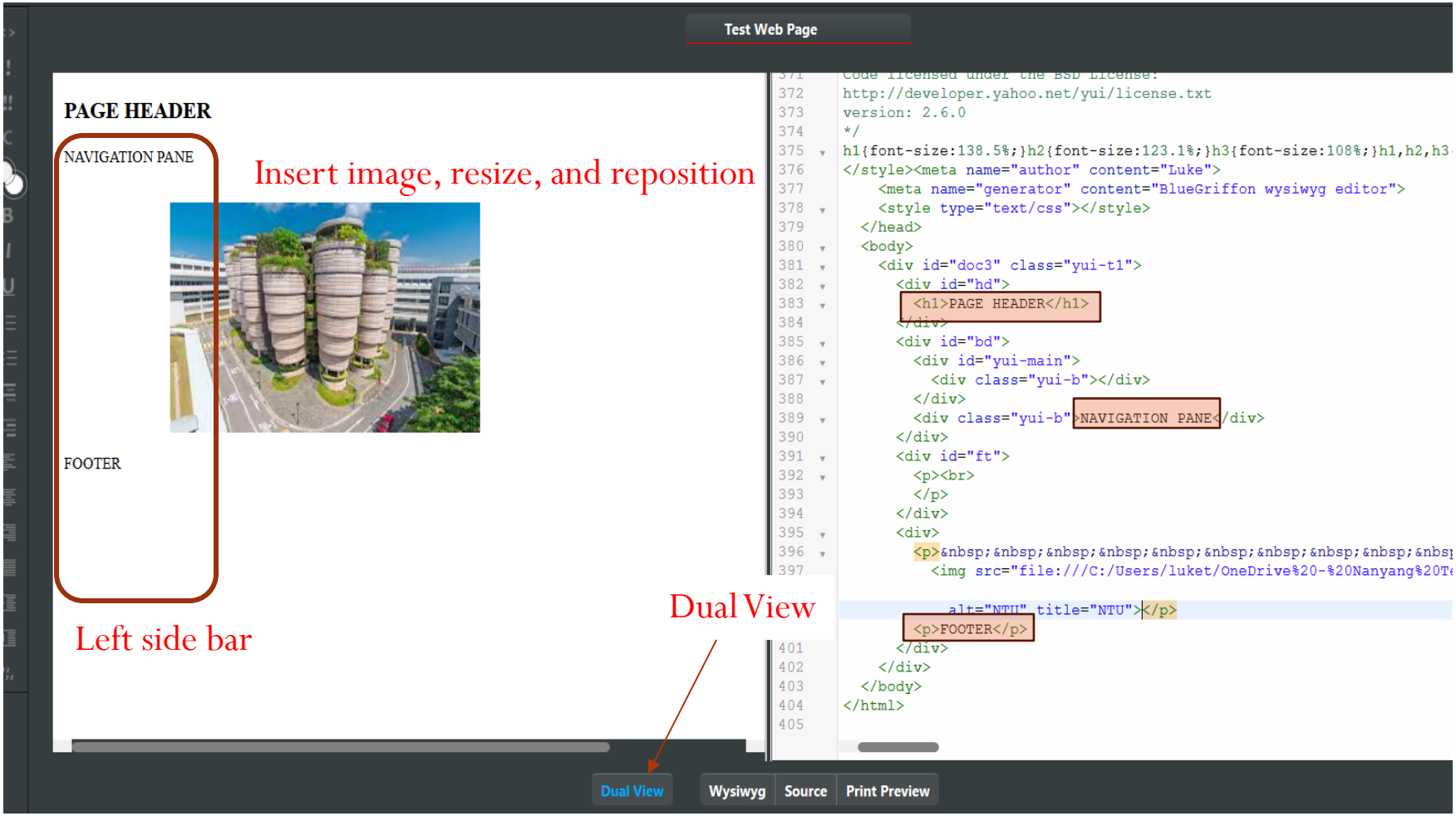
WYSIWYG HTML Editor

- Define the **width** of the main content area of the document and indicate if it should contain a **sidebar** and where (left or right).

- The last checkbox allows to populate the document with **dummy content** – provide visually editable layout of the new document => select the created dummy content and replace it with your own.

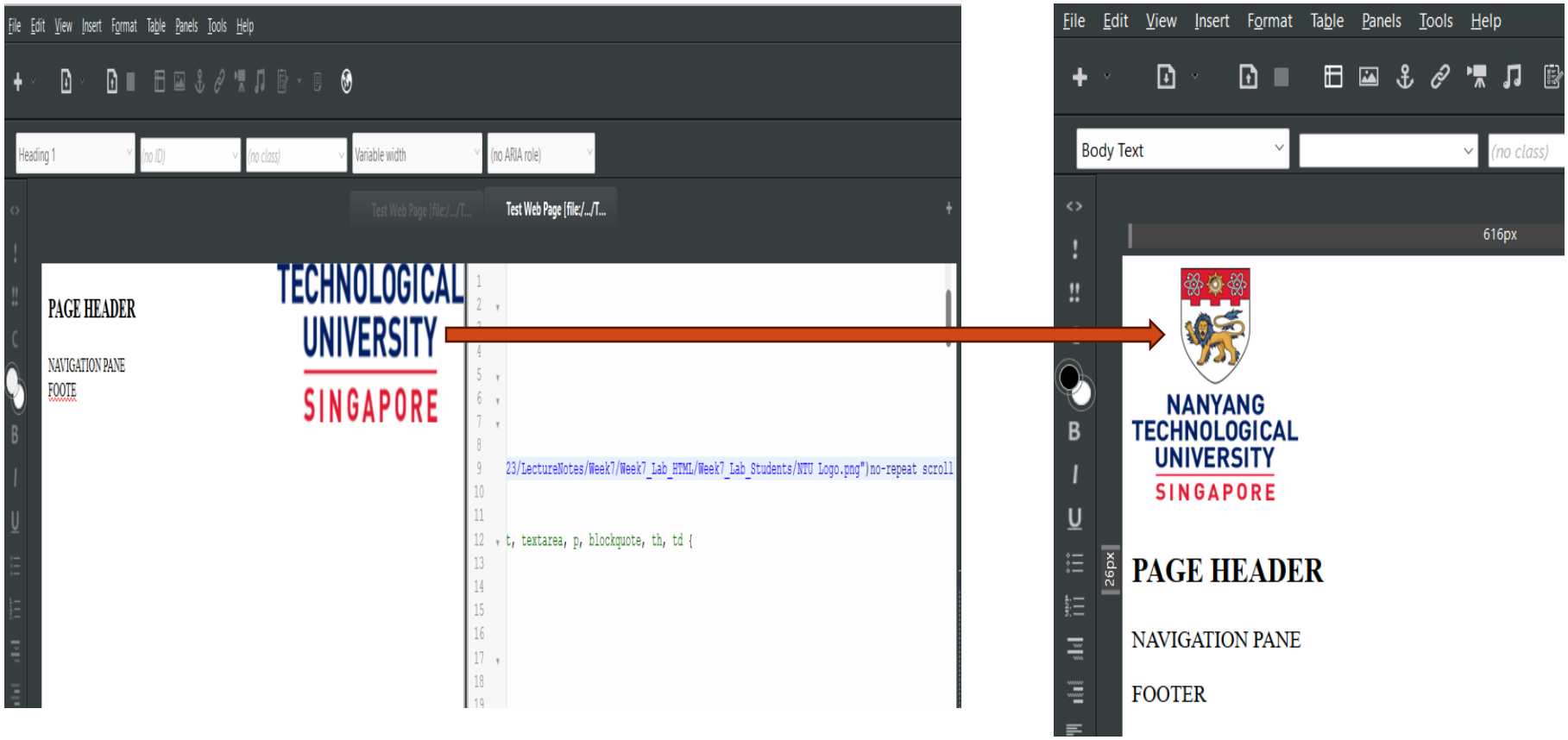


WYSIWYG HTML Editor



In-Class Exercise

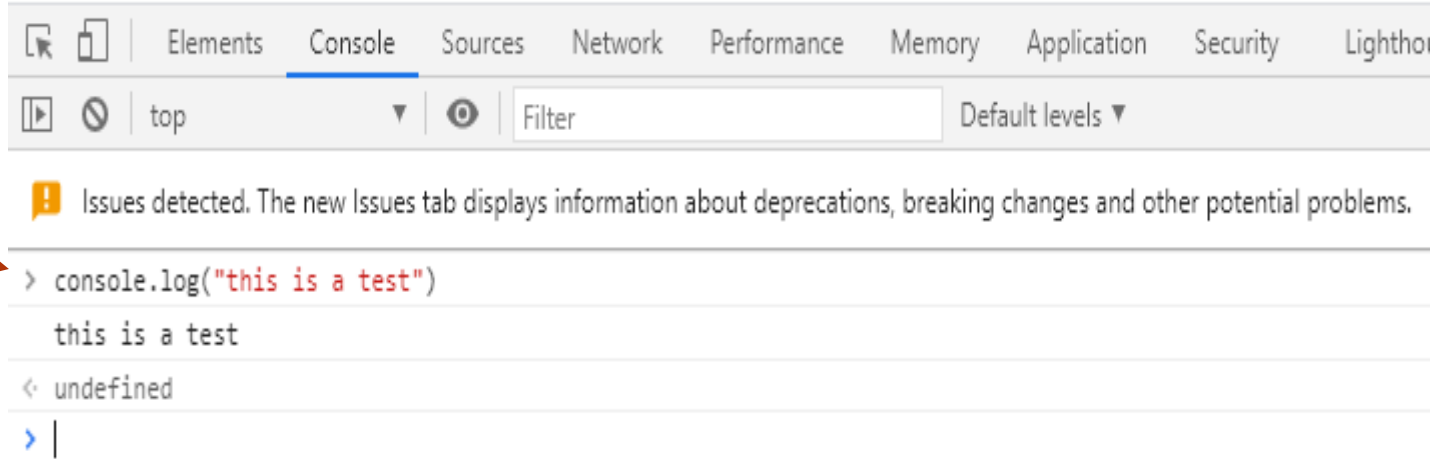
- Use **Blue Griffon** to align the image to the TOP LEFT of the webpage for (**4_First Web Page.html**).



Introduction to JavaScript

- JavaScript is the **scripting** language (**interpreted line by line** and not compiled) that can **make pages dynamic by manipulating the DOM** after a page has loaded in the browser.
- JavaScript code can be written in a text file, and then **loaded to the browser** in a web page **unlike Java codes** which need to be compiled to machine codes.
- JavaScript code can also be **written directly into the browser**. Use the JavaScript **console** for debugging (**Press F12** => Console Tab).

JavaScript code written on Chrome



- E.g., print values out to the console => **console.log("something");**

Introduction to JavaScript

- **JavaScript** – simple example

```
<!DOCTYPE html>
<html>
<body>

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

</body>
</html>
```

getElementById() – one of many JavaScript **methods**

innerHTML property sets or returns HTML content of an element.

Upon clicking button element – **get element by ID "demo"** and **display date time** using Date() method

My First JavaScript

Click me to display Date and Time.

Sat Sep 05 2020 21:21:28 GMT+0800 (Singapore Standard Time)

Introduction to JavaScript

JavaScript can change HTML content.


```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!
</button>

</body>
</html>
```

JavaScript accepts both double and single quotes:

Change the style of an HTML element.


```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>

</body>
</html>
```

Introduction to JavaScript

Show/Hide HTML element.

To hide an element, set the style display property to "none".

```
document.getElementById("element").style.display = "none";
```

To show an element, set the style display property to "block".

```
document.getElementById("element").style.display = "block";
```

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display='block'">Click Me!</button>

</body>
</html>
```

Hidden element p

Display hidden element identified by id = 'demo'

JavaScript - <script> Tag

In HTML, JavaScript **function** is inserted between **<script>** and **</script>** tags.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

The **<body>** encapsulates the **contents** of the document, while the **<head>** part contains **meta** elements, i.e., information about the contents. This is (typically) title, encoding, author, styling, etc

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>JavaScript in Head</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.

Referencing External JavaScript

```
<!DOCTYPE html>
<html>
<body>

<h2>External JavaScript</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p>(myFunction is stored in an external file called "myScript.js")</p>

<script src="myScript.js"></script>

</body>
</html>
```

External JavaScript

A Paragraph.

Try it

External JavaScript

Paragraph changed.

Try it

myFunction is stored in an external file called "myScript.js"

"myScript.js"

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

In JavaScript, statements are concluded with a **semicolon**.

JavaScript – Variables and Arrays

- **Variables** are containers for data => `var number = 5;`
- The **equal** sign is an **assignment operator** - takes the value on the right (5) and **assigns** it to the variable on the left (number)
- **Variable types**:

```
var thisMakesSenseSoFar = true;  //I'm a boolean!
var thisMakesSenseSoFar = "true";  //I'm a string!
```

```
var number = 5;
```

- An **array** stores **multiple values** in a single **variable**.

Keeping track of related values in separate variables is inefficient:

```
var numberA = 5;
var numberB = 10;
var numberC = 15;
var numberD = 20;
var numberE = 25;
```

Hard brackets [] indicate an **array**, and each value is separated by a comma.

```
var numbers = [ 5, 10, 15, 20, 25 ];

numbers[0]  //Returns 5
numbers[1]  //Returns 10
```

Arrays can contain **any type of data**, not just integers:

myArray[0] = Date.now; **(Objects)**

myArray[1] = myFunction; **(Functions)**

myArray[2] = myCarsArray; **(Arrays)**

JavaScript **Array** is similar to Python **List**

JavaScript – Objects

- With more complex datasets use an **object** (indicated by **curly brackets** {})
- A **colon** : separates each **property** and its **value**, and a **comma** separates each **property/value pair**:

Object

Property

Value

```
var fruit = {
    kind: "grape",
    color: "red",
    quantity: 12,
    tasty: true
};
```

An object is a **collection of properties**, and a property is an association between a **name** (or ***key***) and a **value**.

A property's value can be a **function**, in which case the property is known as a **method**. e.g., Math.random()

```
fruit.kind      //Returns "grape"
fruit.color     //Returns "red"
fruit.quantity  //Returns 12
fruit.tasty     //Returns true
```

To reference each value, use **dot notation** then specify the name of the property:

- JavaScript **Object** is similar to **Python Dictionary**

JavaScript – Objects and Arrays

- Possible to **combine** these two **structures** to create **arrays** of **objects**, **objects** of **arrays**, or **objects** of **objects**.

Array of Objects

```
var fruits = [
    {
        kind: "grape",
        color: "red",
        quantity: 12,
        tasty: true
    },
    {
        kind: "kiwi",
        color: "brown",
        quantity: 98,
        tasty: true
    },
    {
        kind: "banana",
        color: "yellow",
        quantity: 0,
        tasty: true
    }
];
```

fruits[0]

fruits[1]

fruits[2]

[] means array, and {} means object.

```
fruits[0].kind      == "grape"
fruits[0].color     == "red"
fruits[0].quantity  == 12
fruits[0].tasty     == true
```

What we have in fruits[2].quantity?

JavaScript – Objects and Arrays

Objects of Arrays

```
var defaults = {
 backgroundcolor: '#000',
 color: '#fff',
 weekdays: ['sun','mon','tue','wed','thu','fri','sat']
};
```

Array in an object

Property of object

Property value = Array

defaults.weekdays[2] = ?

Objects of Objects

```
var pause_menu = {
    pause_button : { someProperty : "prop1", someOther : "prop2" },
    resume_button : { resumeProp : "prop", resumeProp2 : false },
    quit_button : false
};
```

Properties of object

Property value = Object

pause_menu.pause_button.someProperty = ?

JavaScript - Operators

Mathematical Operators

```
1 + 2       //Returns 3
10 - 0.5    //Retuns 9.5
33 * 3      //Returns 99
3 / 4       //Returns 0.75
```

```
console.log(3 + 4 * 5);
```

What is the result?

Comparison Operators

```
==  //Equal to
!=  //Not equal to
<   //Less than
>   //Greater than
<=  //Less than or equal to
>=  //Greater than or equal to
```

If the result is true, then true is returned.
If the result is false, false is returned.

```
3 == 3
3 == 5 → False
3 >= 3
3 >= 2
100 < 2 → False
298 != 298 → False
```

Logical Operators

Combine logical statements

```
&&   //AND
||   //OR
```

```
3 == 3 && 4 == 4 → True
3 == 3 && 4 == 5 → False
2 == 3 && 4 == 5 → False

3 == 3 || 4 == 4 → True
3 == 3 || 4 == 5 → True
2 == 3 || 4 == 5 → False
```

JavaScript – Control Structures

- An **if statement** uses **comparison** operators to determine if a statement is true or false:

```
if (3 < 5) {
    console.log("Eureka! Three is less than five!");
}
```

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

If else **shorthand**

```
condition ? expression_1 : expression_2
```

If condition true return **exp1** else return **exp2**

```
// only register if the age is greater than 18
var allowRegister = age > 18 ? true : false;
```

- Use **for loops** to **repeatedly** execute the same code, with slight variations.

```
for (var i = 0; i < 5; i++) {
    console.log(i);  //Prints value to console
}
```

Output => 0,1,2,3,4

```
var numbers = [ 8, 100, 22, 98, 99, 45 ];

for (var i = 0; i < numbers.length; i++) {
    console.log(numbers[i]);  //Print value to console
}
```

Output => 8, 100, 22, 98, 99, 45

length is a property of **array**. In this case, numbers contains six values, so **numbers.length resolves to 6**, and the loop runs six times.

JavaScript – Control Structures

- **While Loop** – creates a loop that is executed while a **specified condition is true**. The loop continue to run as long as condition is true - only stop when the condition becomes false.

```
<p>Click the button to loop through a block of code as long as i is less than 5.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var text = "";
  var i = 0;
  while (i < 5) {
    text += "<br>The number is " + i;
    i++;
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
```

Loop through as long as variable (i) is less than 5:

Click the button to loop through a block of code as long as i is less than 5.

Try it

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

The += operator adds on the multi-lines to text

JavaScript – Functions

- Functions take **arguments** or **parameters** as input, and then **return values** as output. Parentheses are used to **call (execute)** a function.

Assign variable a function

```
var calculateGratuity = function(bill) {
    return bill * 0.2;
};
```

argument

return value

Declares a new variable named **calculateGratuity** and store a function in the variable. In the parentheses, specify **bill as (input)**, a variable to be used by the function itself. When called, the function will take the input, multiply it by 0.2, and **return** the result as its output.

```
calculateGratuity(38);
```

=> 38 * 0.2

Parentheses call the function

In-Class Exercise - JavaScript

- Open any text editor – e.g., Notepad++

- Edit the codes in following HTML files to make them work:
  - 7_JS_DateTime.html
  - 8_JS_ChangeHTMLContent.html
  - 9_JS_ChangeStyle.html
  - 10_JS_ShowHiddenContent.html
  - 11_JS_Function.html

- Edit the file JS_External.html and myScript.js (given in JavaScripts_External folder) to make them work.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=windows-1252">
  </head>
  <body>
    <h2>Display Date and Time</h2>
    <p id="demo">JavaScript can display date and time</p>
    <button type="button" onclick='document.getElementById("demo").innerHTML = '>Click
      me to display Date and Time</button>
  </body>
</html>
```

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=windows-1252">
  </head>
  <body>
    <h2>Display Date and Time</h2>
    <p id="demo">JavaScript can display date and time</p>
    <button type="button" onclick='document.getElementById("demo").innerHTML = Date()'>Click
      me to display Date and Time</button>
  </body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo"). = "Hello JavaScript!"'>Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.ElementById('demo').style.fontSize='35px'">Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display=">Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display='block'">Click Me!</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
"The temperature is " + (77) + " Celsius";

function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
</script>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
"The temperature is " + toCelsius(77) + " Celsius";

function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
</script>

</body>
</html>
```

JavaScript Frameworks

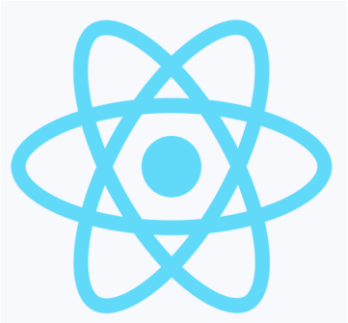
- A software framework is an abstraction in which software providing **generic functionality** can be selectively changed by additional user-written code.
- **JavaScript framework** is an application framework written in JavaScript where the programmers can manipulate the functions and use them for their convenience

Angular (Google)

Node.js

Server-Side Development

React.js (Facebook)



Vue.js

Bootstrap

Web Application Development

https://getbootstrap.com/

Introduction to Bootstrap

- Most websites share **similar structure**. The aim of **frameworks** is to provide a **common structure** so that developers don't have to redo from scratch and can reuse the code provided.
- Open-source and free **CSS framework** => responsive device-friendly mobile-first front-end web page development tool. Bootstrap includes the **CSS (Cascading Style Sheets)**, and an optional **JavaScript** supported design **template** (plug-ins) that deals with typography, implementation of **buttons**, **forms**, and various other components user interface.
- **Responsive design and looks**: Web pages designed using the Bootstrap framework has responsive CSS that can adjust to the screen size of large desktops, notebooks, tablets, and mobiles.
- **Relatively simple and easy to start**

Bootstrap – Environment Setup

- There are **two ways** of using Bootstrap:
- (1) **Bootstrap CDN** (Content Delivery Network) where the code is hosted externally => link and include in project.
- (2) **Download** the Bootstrap package from http://getbootstrap.com/ on local machine. Internet connectivity during project run is not required.

- The bootstrap package consists of:
  - **Bootstrap CSS**: a CSS framework.
  - **Bootstrap js**: a JavaScript / jQuery framework.

- Bootstrap needs **jQuery** for functioning. jQuery is a commonly used JavaScript library which simplifies the cross-browser compatible functionality.

- You can **download** and get the jQuery latest version from https://jquery.com/download/ => right-click Download the compressed, production jQuery 3.7.1 => save link as… (to save the .js file)

Bootstrap_setup_**CDN**.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2, minimum-scale=1">
</head>

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha3

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<!-- Latest compiled Bootstrap JS-->
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1
<body>

    <h1>This is page heading.</h1>
    <p>This is my first <strong>paragraph text</strong>.</p>

</body>
</html>
```

CSS

jQuery

Bootstrap JS

Bootstrap_setup_**Local**.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2, minimum-scale=1">
</head>

<!-- Bootstrap CSS -->
<link rel="stylesheet" type="text/css" href="bootstrap-4.6.0-dist/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="jquery-3.6.0.min.js"></script>

<!-- Latest compiled Bootstrap JS-->
<script src="bootstrap-4.6.0/js/bootstrap.min.js"></script>
<body>

    <h1>This is page heading.</h1>
    <p>This is my first <strong>paragraph text</strong>.</p>

</body>
</html>
```

CSS

jQuery

Bootstrap JS

Bootstrap – Containers

- Containers are the **most basic layout element** in Bootstrap and are **required when using the default grid system**. Containers are used to contain, pad, and (sometimes) center the content within them.

Bootstrap comes with three different containers:

- .container, which sets a max-width at each responsive breakpoint
- .container-{breakpoint}, which is width then reach 100% until the **specified** breakpoint
- .container-fluid, which is width then reach 100% at **all** breakpoints

	Extra small <576px	**Small** ≥576px	**Medium** ≥768px	**Large** ≥992px	**Extra large** ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

Breakpoints

{breakpoint}	Description 1 pixel = 0.026 cm
sm	Works when the width of parent element is >= 567px.
md	Works when the width of parent element is >= 768px.
lg	Works when the width of parent element is >= 992px.
xl	Works when the width of parent element is >= 1200px.

Bootstrap – Grid Layout

The **grid** system is broken down into **twelve column-like segments** across the page. But there are situations where users do not want to use the **12 column** structure, and so the **column grouping**, i.e., the grid structure, can be formed in various other ways.

col-8								col-4			
col-4				col-4				col-4			
col-9									col-3		
col-12											
col	col	col	col	col	col	col	col	col	col	col	col

Five classes of the grid system:

1 pixel = 0.026 cm

col-	It is for other small devices where the screen size is less than 576px. => 14.9 cm
col-sm-	It is for small devices where the screen size is equal to or greater than 576px.
col-md-	It is for medium devices where the screen width equal to or greater than 768px.
col-lg-	It is for large devices where the screen width equal to or greater than 992px.
col-xl-	It is for extra-large devices where the screen width equal to or greater than 1200px.

The grid system will adjust its content to **rearrange the columns based on the screen size automatically**.

Bootstrap – Grid Layout

```
<body>
    <div class="container">
        <div class="row">
            <div class="col-sm-12 col-md-9 col-lg-9 col-xl-9 bg-contain bg-white border mb-4">
                <h1 class="h2 px-3 m-0 txt-rblue font-weight-semibold mb-4">Loan For Your Education</h1>
                <div class="text-size px-3 bg-white-transparent">
                    <p>Exclusive Funding your education could never be this easy. Online applications,
                    instant offer and get maximum funding for your education with loans of LoanCap.</p>
                    <p>We have worked out an exclusive deal for you with our partner LoanCap.</p>
                    <ul>
                        <li>Funding up to 100%</li>
                        <li>Fast Loan Appraval</li>
                        <li>Instant Offer</li>
                    </ul>
                </div>
                <div class="text-center pb-3">
                    <button type="button" name="submit_button" class="btn btn-primary radius-full
                    shadow-small" ng-disabled="processing">Apply Now</button>
                </div>
            </div>
            <div class="col-sm-12 col-md-3 col-lg-3 col-xl-3 d-none d-sm-block border">
                <div class="text-center">Advertisment</div>
            </div>
        </div>
    </div>
</body>
```

classes can be combined to create more dynamic and flexible layouts.

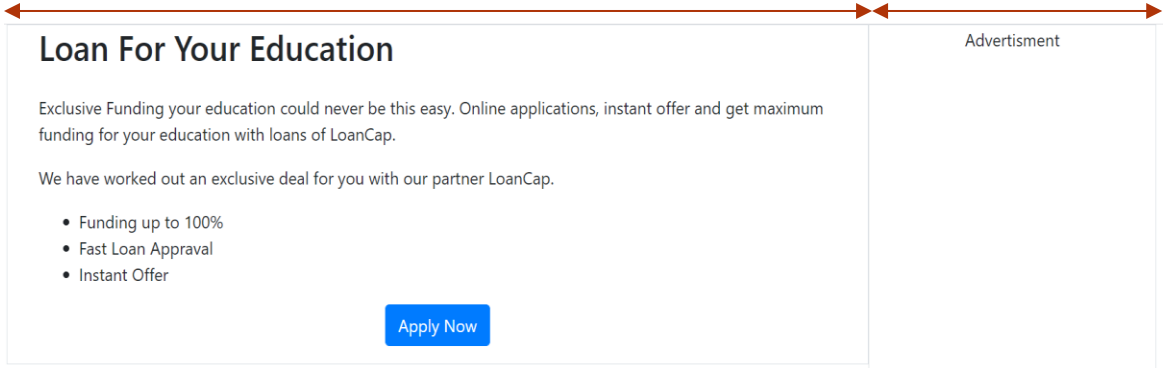
col-md-9

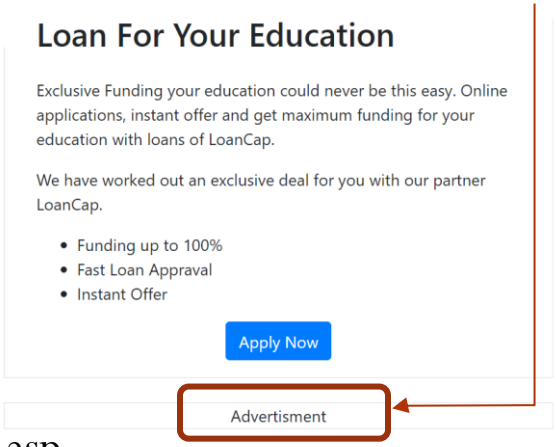
col-md-3

col-sm-12 => uses all 12 columns when screen is small

Col-9

Col-3





https://www.w3schools.com/bootstrap/bootstrap_ref_all_classes.asp

Bootstrap – Font Style

- By default, Bootstrap uses **1rem(16px)** as **font size** and the **line-height remains 1.5 relative to font size**. Default font-family is "**Helvetica**". Furthermore, all **<p> (paragraph)** elements have **margin-top set to 0** and **margin-bottom set to 1rem**.

```
<div class="container">
  <h1>h1 Bootstrap heading (2.5rem = 40px)</h1>
  <h2>h2 Bootstrap heading (2rem = 32px)</h2>
  <h3>h3 Bootstrap heading (1.75rem = 28px)</h3>
  <h4>h4 Bootstrap heading (1.5rem = 24px)</h4>
  <h5>h5 Bootstrap heading (1.25rem = 20px)</h5>
  <h6>h6 Bootstrap heading (1rem = 16px)</h6>
</div>
```

h1 Bootstrap heading (2.5rem = 40px)

h2 Bootstrap heading (2rem = 32px)

h3 Bootstrap heading (1.75rem = 28px)

h4 Bootstrap heading (1.5rem = 24px)

h5 Bootstrap heading (1.25rem = 20px)

h6 Bootstrap heading (1rem = 16px)

Bootstrap – Font Style

Display headings can be used when you need to display a significant, slightly **more opinionated title**.

```
<h1 class="display-1">Display Heading 1</h1>
<h1 class="display-2">Display Heading 2</h1>
<h1 class="display-3">Display Heading 3</h1>
<h1 class="display-4">Display Heading 4</h1>
```

Display Heading 1

Display Heading 2

Display Heading 3

Display Heading 4

Lead is used to add some **emphasis** to any paragraph content.

```
<p> This is normal Text without any lead class not emphasized. </p>
<p class="lead"> Example of Lead class with paragraph showing its use to emphasize text.
```

This is normal Text without any lead class not emphasized.

Example of Lead class with paragraph showing its use to emphasize text.

Bootstrap – Font Style

Class Name	Description
.font-weight-bold	For creating bold text.
.font-weight-bolder	For creating bolder text.
.font-italic	For creating italic text.
.font-weight-light	For creating lightweight text.
.font-weight-lighter	For creating lighter weight text.
.font-weight-normal	For creating normal text.
.small	For creating a smaller text which is comparatively 85% smaller than the size of the parent.
.text-break	This class helps in preventing long text from breaking design and layout.
.text-decoration-none	For removing the underline from any hyperlink.

Class Name	Description
.text-justify	For creating justified text.
.text-monospace	For creating mono-spaced text.
.text-nowrap	For creating no wrap text.
.text-lowercase	For creating lowercased text.
.text-reset	For resetting the text color or a link.
.text-uppercase	For creating uppercased text.
.text-capitalize	For creating capitalized text.

https://www.w3schools.com/bootstrap4/bootstrap_typography.asp

Bootstrap – Font Style Colors

Contextual text colors

```
<div class="container">
  <h2>Contextual Colors</h2>
  <p>Use the contextual classes to provide "meaning through colors":</p>
  <p class="text-muted">This text is muted.</p>
  <p class="text-primary">This text is important.</p>
  <p class="text-success">This text indicates success.</p>
  <p class="text-info">This text represents some information.</p>
  <p class="text-warning">This text represents a warning.</p>
  <p class="text-danger">This text represents danger.</p>
  <p class="text-secondary">Secondary text.</p>
  <p class="text-dark">This text is dark grey.</p>
  <p class="text-body">Default body color (often black).</p>
  <p class="text-light">This text is light grey (on white background).</p>
  <p class="text-white">This text is white (on white background).</p>
</div>
```

Contextual classes provide "meaning through colors":

This text is muted.

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

Secondary text.

This text is dark grey.

Default body color (often black).

Background colors

```
<div class="container">
  <h2>Contextual Backgrounds</h2>
  <p>Use the contextual background classes to provide "meaning through colors".</p>
  <p>Note that you can also add a .text-* class if you want a different text color:</p>
  <p class="bg-primary text-white">This text is important.</p>
  <p class="bg-success text-white">This text indicates success.</p>
  <p class="bg-info text-white">This text represents some information.</p>
  <p class="bg-warning text-white">This text represents a warning.</p>
  <p class="bg-danger text-white">This text represents danger.</p>
  <p class="bg-secondary text-white">Secondary background color.</p>
  <p class="bg-dark text-white">Dark grey background color.</p>
  <p class="bg-light text-dark">Light grey background color.</p>
</div>
```

Background Text

Background colors do not set the **text color**, use together with **.text-*** class.

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

Secondary background color.

Dark grey background color.

Light grey background color.

Bootstrap – Table

```
<div class="container">
  <h2>Basic Table</h2>
  <p>The .table class adds basic styling (light padding and only horizontal dividers) to a table:</p>
  <table class="table">
    <thead>
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Email</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John</td>
        <td>Doe</td>
        <td>john@example.com</td>
      </tr>
```

thead – table header => group header content in table

tr – table row, **th** – header cell

tbody – table body, **td** – table data cell

Basic Table (basic styling)

The .table class adds basic styling (light padding and only horizontal dividers) to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
	Dooley	july@example.com

```
<div class="container">
  <h2>Striped Rows</h2>
  <p>The .table-striped class adds zebra-stripes to a table:</p>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Email</th>
      </tr>
    </thead>
```

.table-striped class adds zebra-stripes to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Bootstrap – Table

More interesting classes: **Contextual classes** can be used to color table **rows** (<tr>) or table **data cells** (<td>):

```
<div class="container">
  <h2>Contextual Classes</h2>
  <p>Contextual classes can be used to colo
  <table class="table">
    <thead>
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Email</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Default</td>
        <td>Defaultson</td>
        <td>def@somemail.com</td>
      </tr>
      <tr class="table-primary">
        <td>Primary</td>
        <td>Joe</td>
        <td>joe@example.com</td>
      </tr>
      <tr class="table-success">
        <td>Success</td>
        <td>Doe</td>
        <td>john@example.com</td>
      </tr>
      <tr class="table-danger">
        <td>Danger</td>
        <td>Moe</td>
        <td>mary@example.com</td>
      </tr>
```

Contextual Classes

Contextual classes can be used to color the table, table rows or table cells. The classes that can be used are: .table-primary, .table-success, .table-info, .table-warning, .table-danger, .table-active, .table-secondary, .table-light and .table-dark:

Firstname	Lastname	Email
Default	Defaultson	def@somemail.com
Primary	Joe	joe@example.com
Success	Doe	john@example.com
Danger	Moe	mary@example.com
Info	Dooley	july@example.com
Warning	Refs	bo@example.com
Active	Activeson	act@example.com
Secondary	Secondson	sec@example.com
Light	Angie	angie@example.com
Dark	Bo	bo@example.com

Bootstrap – Progress Bar Class

A **progress bar** shows how far along is a process.

```
<div class="bs-example">
    <!-- Progress bar HTML -->
    <div class="progress">
        <div class="progress-bar progress-bar-striped" style="min-width: 20px;"></div>
    </div>

    <!-- jQuery Script -->
    <script>
        var i = 0;
        function makeProgress(){
            if(i < 100){
                i = i + 1;
                $(".progress-bar").css("width", i + "%").text(i + " %");
            }
            // Wait for sometime before running this script again
            setTimeout("makeProgress()", 100);
        }
        makeProgress();
    </script>
</div>
```

Specify progress bar class

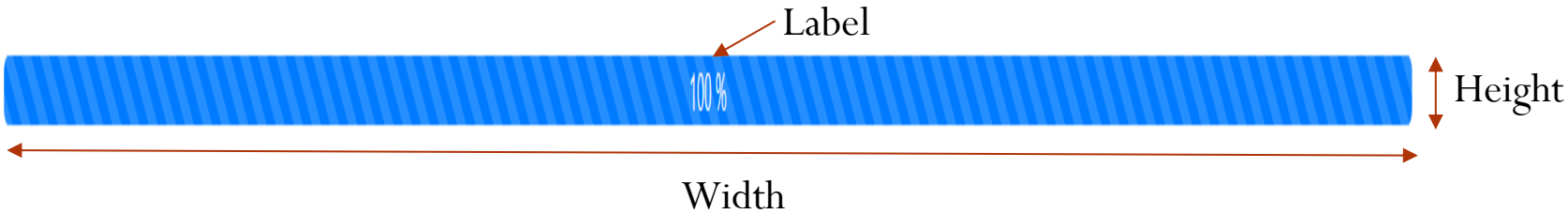
Specify width "progress" in bar

Specify label

$ => Javascript identifier

To create a default progress bar, add a .progress class to a container element and add the progress-bar class to its child element. Use the CSS width property to set the width of the progress bar:

```
<div class="progress-bar" style="width:40%;height:20px"></div>
```



Bootstrap – Carousel Class

Attribute used to mark a carousel as moving animation

```
<div id="demo" class="carousel slide" data-ride="carousel">

  <!-- Indicators -->
  <ul class="carousel-indicators">
    <li data-target="#demo" data-slide-to="0" class="active"></li>
    <li data-target="#demo" data-slide-to="1"></li>
    <li data-target="#demo" data-slide-to="2"></li>
  </ul>
  <!-- The slideshow -->
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="la.jpg" alt="Los Angeles" width="1100" height="500">
    </div>
    <div class="carousel-item">
      <img src="chicago.jpg" alt="Chicago" width="1100" height="500">
    </div>
    <div class="carousel-item">
      <img src="ny.jpg" alt="New York" width="1100" height="500">
    </div>
  </div>

  <!-- Left and right controls -->
  <a class="carousel-control-prev" href="#demo" data-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </a>
  <a class="carousel-control-next" href="#demo" data-slide="next">
    <span class="carousel-control-next-icon"></span>
  </a>
</div>
```

Specifies slides "items" in carousel

Specifies active "default" slide

Link active slide

Indicates number of slides, and which slide is current

Alternate text when image cannot be shown

Bootstrap attribute value => navigate slides

Add left arrow and icon

Bootstrap attribute

Add right arrow and icon

https://www.w3schools.com/bootstrap5/bootstrap_ref_js_carousel.asp

Other Bootstrap Classes

Pagination, Navigation, Tooltip, Popover, …

https://www.w3schools.com/bootstrap5/

In-class Exercise - Bootstrap

- Open 1_bootstrap_setup_CDN.html in a browser to see the setup of Bootstrap. Edit the 1_bootstrap_setup_Local.html file to enable Bootstrap for the file.

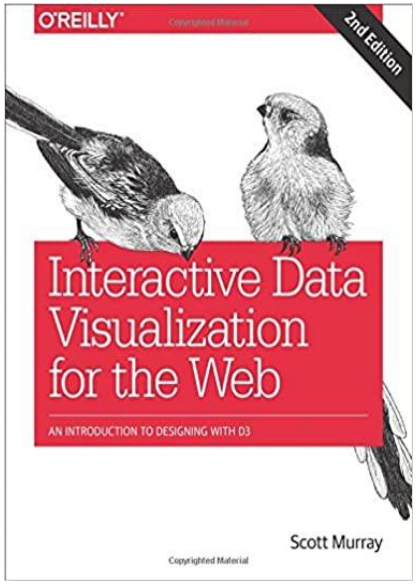
```
<!-- Latest compiled Bootstrap JS-->
<script src="bootstrap-4.6.0/js/bootstrap.min.js"></script>
<body>
```

- Open 2_bootstrap_grid.html in a browser and **adjust the size** of the browser to see the effects of changing columns when the browser size changes.

- Open the 3_bootstrap_carousel.html file to **run** the carousel slides feature.

References

- Murray, S. (2017). Interactive Data Visualization for the Web. An Introduction to Designing with D3, 2nd Ed. Inc.O'Reilly



- https://www.w3schools.com/