**HTML & CSS Course**

03. Typography

* Books – “The elements of typographic style”, “The elements of Typographic style applied to the web”
* Typography is the art and technique of arranging type to make written language, it is the visual elements of the written word
* Website **readability** is a measure of how easy it is for visitors to **read** and **understand** online text
* **Keys to Readable Typography:**
* User-friendly headers
* White Space - scannable and consistent text
* Emphasis of Important Elements
* Good Margins to avoid walls of text
* Scannable Text
* Consistency
* Organization of Information
* CSS font properties:
* font-family/ font-space:

e.g font-family: Arial, Helvetica, but “Times New Roman” in quotes due to the fact it consists of several words with white space between, when 2 font families are given, if one is not available, then the other one is automatically used)

e.g. font-face specifies custom font and the src path @font-face {

font-family: “Open Sans”;

src: url(“/fonts/opensans.woff”) format(“woff”);

}

Google fonts – free fonts fonts.google.com > download family > style.css define @font-face

* font-size/ line-height/letter-spacing – 1px = 0.75pt = 1/96 inch, pt/px/em(relative to the next parent element)/rem(relative to the html parent(default for html is16px))
* font-weight (e.g. bold) – thin(100), light(300), normal(400), bold(700), or value [100-900]
* font-style (e.g. italic) – normal or italic
* font-variant (e.g. small caps)
* text -align: left/right/center/justify – defines the horizontal alignment
* text -decoration: none/line-through/overline/underline
* text -indent – when text starts with space on the line (един пръст място)
* text -overflow: clip(…) – when text is overflowing the box, this defines how the hidden text is displayed
* text-transform: capitalize(first letter of each word)/uppercase(all)/lowercase(all) – specifies how to capitalize text
* word-break: normal(breaks at space symbol)/break-all(breaks at the end of the line)
* text-shadow
* text-color
* background-color: HEX/RGB/<named color>/transparent
* Text-shadow: <horizontal> <vertical> <blur> <color>, ex. text-shadow: 2px 4px 10px red;
* cursor: ponter/move/none/col-resize;
* outline: <width/style/color>; ex: outline: 4px dotted red;
* When more than one font-family is registered through @font-face{…} then later they are identified by referring to their stated font-weight, i.e.:

@font-face{ @font-face{

font-family: “Roboto”; font-family: “Roboto”;

src: url(“./fonts/Roboto-Bold.ttf“; src: url(“./fonts/Roboto-Normal.ttf“;

**font-weight: 700; font-weight: 400;**

} }

h1{

font-family: Roboto;

**front-weight: 700;**

text-align: center;

}

* **Font Awesome** provides vector icons, emojis, etc.
* Add the following link inside **<head>**
* Or import Font Awesome in the **CSS** file
* Choose an icon 🡪 copy the **<i>** element 🡪 paste it in your HTML file

<head>

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.10.2/css/all.css">

</head>

<body>

<a href="#"><i class="fa fa-home fa-fw"></i>Home</a>

</body>

@import 'https://use.fontawesome.com/releases/v5.10.2/css/all.css';

04. **CSS BOXING**

HTML elements contain:

* Content
* Padding
* Border
* Margin

Block vs Inline HTML elements

* Block – starts on a new line and fills up all the space left and right.

Ex: main, header, article, section, fieldset, nav, ul, ol, li, form, h1-h6, p, div

* Inline – don’t start on a new line and appear on the same line as the content and tags beside them, margins can be added only on right and left

Ex: a, label, map, span, strong, em, I, img, textarea, input, button, select

* Inline-block – we can set margin and padding to all four sides, we have to declare the element as display: inline-block. Commonly these elements are used to create nav links horizontally

Width:

* Auto
* Em/rem/px
* % from the parent element
* Min-width: 300px; - at least 300px but can be more
* Max-width:500px; - up to 500px but can be less

Height:

* Auto
* Em/rem/px
* % from the parent element
* Overflow: visible/auto/scroll/hidden – how to display content longer than the height; overflow-x: (only horizontal scroll)/overflow-y: auto(only vertical scroll)
* Max-height: if the max height is larger that the elements actual height, the max height has no effect and an automatic scroller appears
* Min-height:

Padding vs Margin:

* Padding creates space inside the element, inside the box border
* Margin creates space outside of the element, between the elements
* Values are set clockwise, top, right, bottom, left, i.e. margin/padding: 20px, 20px, 20px, 20px

Or when only 2 values they would be top & bottom, left & right, i.e. margin/padding: 20px, 20px;

Border:

* Border: {width} {style} {color}
* Width = px. Em, rem, %
* Style: solid/dashed/dotted/double/groove/inset/ridge/outset/none
* Color: word/hex/rgb
* Border-width:
* Border-style:
* Border-color:
* Border-radius:
* Border-top-left-radius:
* Border-bottom-style
* Border-left-color:

Box-sizing:

* Box-sizing: content-box; the size of the content has priority and the other elements of the container expand from there
* Box-sizing: border-box; the border determines the size of the container and the content inside adjusts to it
* Universal box-sizing; saves time to write the same thing again and again. Can be set once like this:

**html {**

**box-sizing: border-box;**

**}**

**\*,**

**\*:before,**

**\*:after {**

**box-sizing: inherit; // or box-sizing: border-box**

**}**

05. Position & Grid

Properties: position, z-index(layering), location (top, bottom, left, right)

Position:

* static – default position
* relative – other properties are calculated relative to the parent, the element position is kept in the layer
* absolute – the element is no longer part of that z-layer, the other elements are positioned regardless of an absolute element, it starts appearing more like an inline element. This element’s top and bottom, left and right are calculated to the next non-static parent element
* fixed – position according to the view port, it is used for navigation bar
* sticky – the element behaves as position relative or static, but one we reach with scrolling the page the top/bottom/left values we’ve set, it becomes fixed

Layering:

* Z-index is used on non-static elements
* it shows the priority of the elements – which should appear on top
* by default the z-index is 0
* used and applied on elements with position property that is non-static
* -1 is behind the parent element, while 1+ is in front

Float:

* Used for styling images, so that the text is wrapped around the element
* The text after the image is the one that wraps around it

Grid:

* Used for creating layout
* display: grid;
* grid-template-column: 300px 300px; => 2 columns 300px each
* grid-template- column: repeat(2, 300px); => 2 columns 300px each
* grid-template- column: 100px repeat(2, 300px); => 1 column 100px and 2 columns 300px each
* grid-template- column: 300px 1fr; => 1 column 300px and the rest is 1 fraction
* grid-template- column: repeat(3, 1fr); => 3 equal columns or 3 equal fractions
* gap: 24px;

06. Flexbox

* display:
* flex => Flexbox property is given to the parent element, which will arrange the child elements inside by default as inline-block, regardless of their usual state (ex. p is a block element but when in a .container{display: flex;} it would be in one line with the other child elements)
* display: inline-flex => the parent element (.container) will behave like an inline-block element with regard to the other elements around it
* flex-direction:
* row & row-reverse; 1 2 3 \_\_\_\_ & \_\_\_\_\_3 2 1
* column & column-reverse;
* flex-wrap:
* nowrap; - elements appear on a single line no matter what
* wrap; - elements will be distributed on multiple lines if needed
* wrap-reverse; - elements will be in the correct order but rows will be in reverse order, i.e. 5 6 / 1 2 3 4
* flex-flow: combination between flex-direction and flex-wrap; default is: row wrap;
* justify-content: defines how items are aligned horizontally within the flexbox container
* flex-start; / flex-end; / center; / space-between; / space-around;
* align-items: defines how items are aligned vertically
* flex-start; / flex-end; / center; / baseline; / stretch;
* align-content:
* flex-start; / flex-end; / center; / space-between; / space-around;