ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Last lecture reminder

CSS describes how HTML elements should be styled and display on our browsers

We learned about some basic features with css:

- How we apply style to different elements
- What are the css selectors
- How to add different colors using rgb() and hex
- How to import and different fonts from the browser
- What is the css box model
- What are the css size units
- How to style containers and buttons using pure css

Display - Inline and Block

As we talked about before every HTML element has its own display configuration

For example:

- is a block element - meaning when we insert a the page will create a new block for it
- is an inline element - meaning when we insert a the page will add it to the current block

Using the display property we can change the HTML default configuration

- **display: inline -** will show this element as an inline element
- **display: block -** will show this element as a block element

Positioning

The position property specifies the type of positioning method used for an element

There are five different position values:

static - element that is not positioned in any special way, We can’t position it using TBLR (Top, Buttom, Left, Right)

relative - element that we can position using TBLR

fixed - positioned relative to the view port

absolute - positioned relative to its parent element that is positioned “relative”

sticky - positioned based on scroll position

Z-index

The **z-index** property specifies the stack order of an element.
An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements).

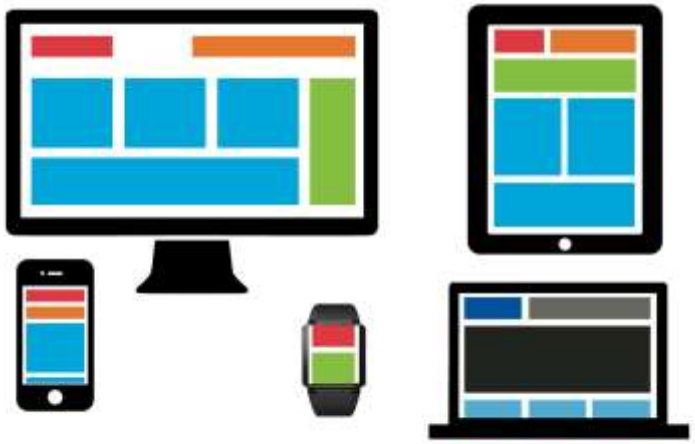
If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Responsive Design

Responsive web design makes your web page look good on all devices

Responsive web design uses only HTML and CSS, meaning that its not using any javascript for achieving this

Our web pages should not leave out information to fit smaller devices,
but rather adapt its content to fit any device



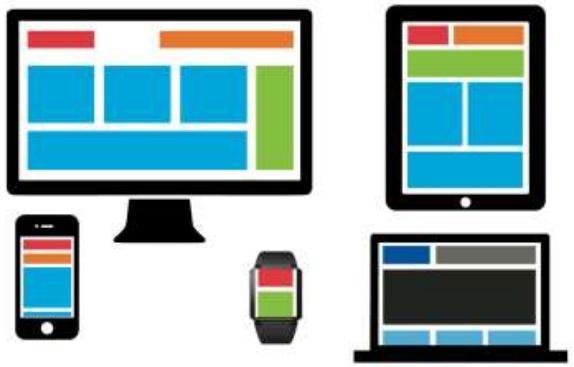
Note: Nowadays it's a necessity to build responsive layouts!

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Responsive Design

What should we use to make our websites responsive:

- **Using the width Property -** If the CSS width property is set to 100% the image will be responsive and scale up and down
- **Using the max-width Property -** If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size
- **Use fluid width as opposed to fixed**
- **Media queries -** Different css styling for different devices
- **Rem units over px -** rem using the font size of the root element, making it more responsive
- **Mobile first method -** First we will design our smaller screens and then we will define what we want to show on the larger screens



Media Queries

The @media rule is used in media queries to apply different styles for different media types/devices

Using the @media rule we can apply “breakpoints” that allow us the configure the relevant style for each device

Media queries can be used to check many things, such as:

- width and height of the viewport (the available screen)
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering responsive web design

How to apply Media Queries

Simple steps to create media query:

1. create a media query using the @media sign in our style sheet
2. open () next to the @media sign and write your condition (the breakpoint)
3. open {} next to your condition and write a valid css syntax (element selector, style properties and values)

Once your page will hit the condition you added to the media query, the style you added will be apply automatically and override your previous style configuration

```
@media(max-width: 500px){
    body{
        background-color: red;
    }

}
```

Example - Media queries

Flexbox Introduction

The Flexbox Layout module aims at providing a more efficient way to lay out, align and distribute space among items in a container

A flex container expands items to fill available free space or shrinks them to prevent overflow

Note: Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

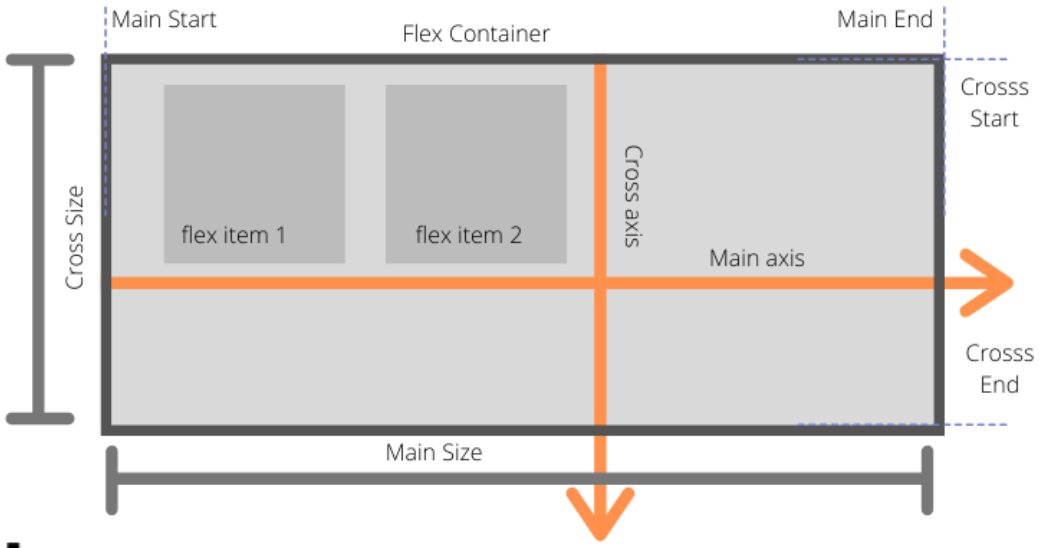
Flexbox

“flex” is a value for the display property (like display: inline, display: block that we learned before)

Using Flexbox we can align elements horizontal and vertical (before we could do it only vertical)

How is it working:

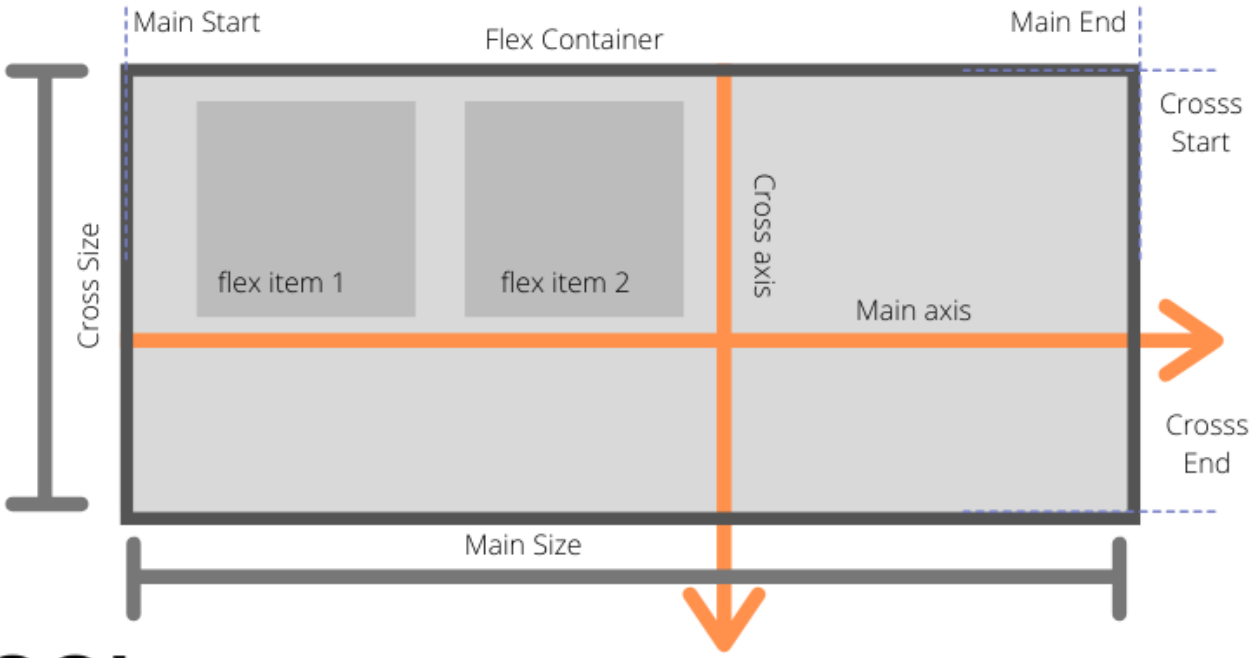
- If we configure our container to be display: flex we are creating a **“flex” container**
- All the elements inside this container (child elements) are called **“flex items”**



Flexbox

Alignment properties:

- **align-items -** align the flex items vertically (along the cross axis)
- **justify-content -** align the flex items horizontally (along the main axis)
- **align-content -** align when extra space in cross axis



ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

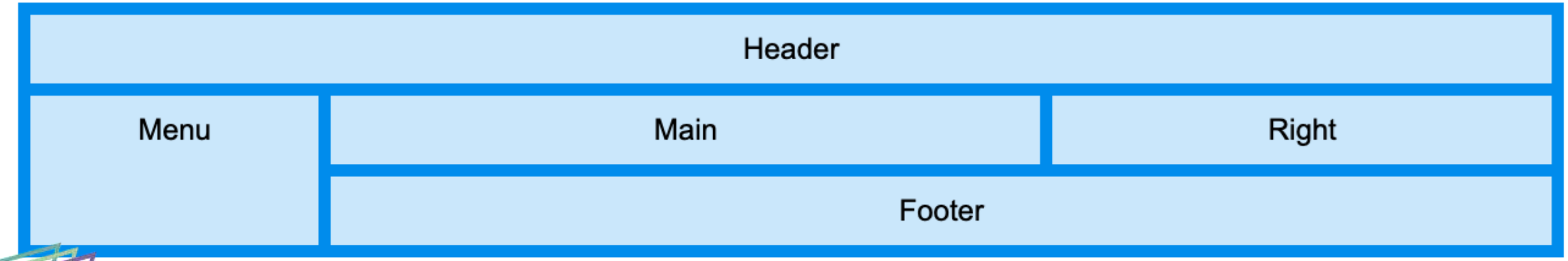
Example - Flexbox

CSS Grid Introduction

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning

A grid layout consists of a parent element, with one or more child elements (just like in the flexbox where we had a flex-container and flex-items inside of it)

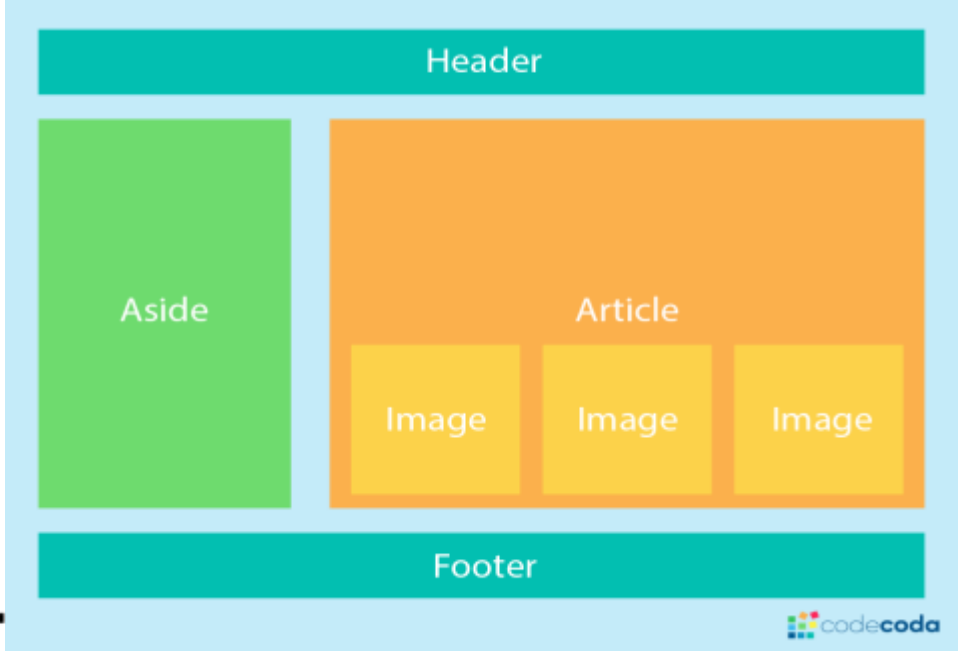
Note: The main difference between Flexbox and Grid is that Grid is supported two dimensional unlike Flexbox that its only 1 dimensional



CSS Grid

How is it working:

- If we configure our container to be **display: grid** we are creating a css grid
- All the elements inside this grid (child elements) are called **“grid items”**
- The **“grid-template-columns”** defines width and the number of columns
- The **“grid-template-rows”** defines width and the number of rows
- Because the css grid is two dimensional we can configure columns and rows for the same grid-container
- **grid-column-start** and **grid-row-start** - using this property we can span columns and rows



Example - CSS Grid

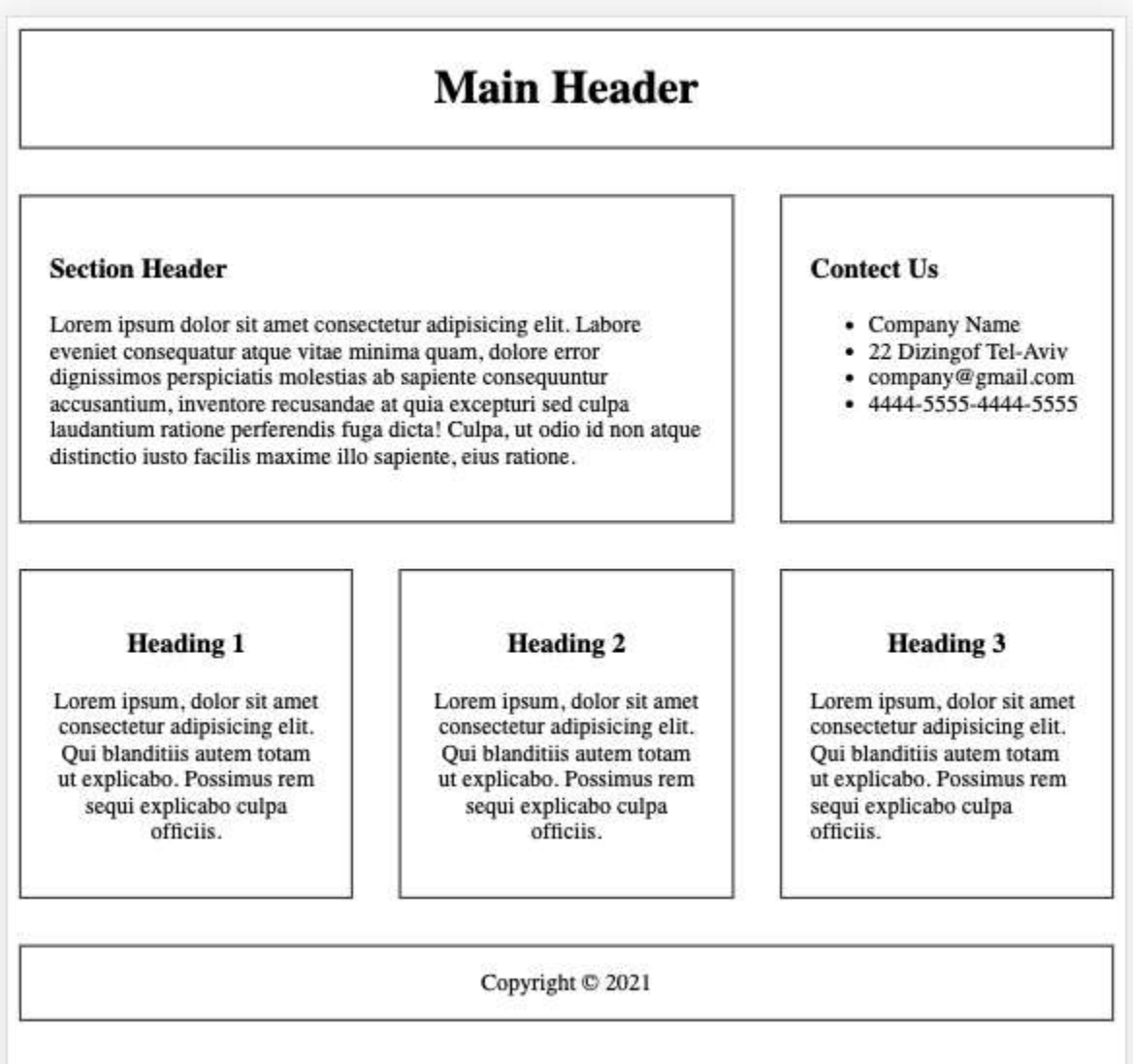
Class Exercise - CSS Grid

Class Exercise - CSS Grid

Instructions:

- You will be provided with basic HTML code with no styling
- Your mission is to create the provided web page layout using only css and css grid
- Your page should be responsive and fit to all devices viewport
- If you don’t know / remember how to do something you should look for a solution online

Class Exercise - CSS Grid



Class Exercise - CSS Grid Solution