**Ministry of education and science of Ukraine**

**Ivano-Frankivsk national technical university of oil and gas**

**Computer Systems and networks department**

 **M.O. Slabinoha**

**FRONTEND DEVELOPMENT AND WEB-DESIGN**

###### **LABS MANUAL**

PART 1

######

**Ivano-Frankivsk**

**2021**

UDC 004.514

С 47

| Reviewer: |   |
| --- | --- |
| A.R. Voronych | PhD, Docent, Docent of computer systems and networks department, Ivano-Frankivsk national technical university of oil and gas |

*Recommended by methodical university council*

*( protocol # of )*

M.O. Slabinoha

С 47. Frontend development and web-design: labs manual (part 1)/ M.O. Slabinoha – Ivano-Frankivsk, IFNTUOG, 2021. – 22 p.

 МВ 02070855- -2021

The labs manual on the discipline "Frontend development and web design" are developed in accordance with the syllabus of the discipline and the working curriculum.

Designed to prepare bachelors in specialty 123 - "Computer Engineering". Lecture notes can be used by full-time and part-time students.

 © M.O. Slabinoha

 © IFNTUOG, 2021

UDC 004.514

С 47

| Reviewer: |   |
| --- | --- |
| A.R. Voronych | PhD, Docent, Docent of computer systems and networks department, Ivano-Frankivsk national technical university of oil and gas |

*Recommended by methodical university council*

*( protocol # of )*

M.O. Slabinoha

С 47. Frontend development and web-design: labs manual (part 1)/ M.O. Slabinoha – Ivano-Frankivsk, IFNTUOG, 2021. – 22 p.

The labs manual on the discipline "Frontend development and web design" are developed in accordance with the syllabus of the discipline and the working curriculum.

Designed to prepare bachelors in specialty 123 - "Computer Engineering". Lecture notes can be used by full-time and part-time students.

 © M.O. Slabinoha

 © IFNTUOG, 2021

Head of computer systems and networks department S.I. Melnychuk

Agreed:

Member of the expert review

university commission V.V. Bandura

Normocontroller H. Ya.Tomashivska

Leading librarian STL H. M. Matsyuk

CONTENTS

DISCLAIMER…………………………………………………………………..5

LAB 1…...……………………………………………………………………....6

LAB 2……...…………………………………………………………………....9

LAB 3………...………………………………………………………………..13

LAB 4…………...……………………………………………………………..16

LAB 5……………....………………………………………………………….18

LAB 6………………....……………………………………………………….20

REFERENCES………………………………………………………………...22

##

## DISCLAIMER

## This lab manual are just compilation of different sources that explain the topic of subject in the best way the author can imagine. The author of this document is grateful for the tremendous work of creators of these materials and doesn’t mean to violate their exclusive copyright on their creation.

## This will be used with all the referencing responsibility for teaching the students and making IT education better. The author doesn’t support the plagiarism in any possible way. All the references are mentioned in the corresponding section

## LAB 1

CREATING THE WEBSITE USING SAAS SERVICES

**Work goal:** to learn how to create the websites using the SaaS services (i.e., Wix).

With the advent and rapid development of the Internet, many new opportunities have opened up to users, including the ability to communicate. Now you can discuss different topics in forums and get valuable tips, talk about yourself on the blog, find old friends through social networks, and more.

A much wider field of activity opens users to having their own site. This is an opportunity to express yourself, your hobbies, your job, your company. But creating even a simple website design or functionality requires a basic knowledge of HTML and JavaScript, image processing skills, and web document placement on Internet servers.

Today, it is quite popular to use site designers that provide tools for creating sites without proper foundation. Existing services have their own features and benefits, but one of the best projects in their field is the free Wix.com service.

Wix.com is a functional website builder from scratch that will be a great help for users who are not experienced in creating web documents. The whole process is simple, easy and intuitive - with Drag and Drop you can easily drag and drop web page elements, and a selection of settings will help you modify the basic template according to the task of the site.

For beginners, Wix.com provides help and training support to help you create your own site. You can ask questions in the forum.

The main page of the site lists the main features:

- Use of the designer does not require programming knowledge and other special skills, the provided tools allow you to develop Flash or Html5 site of various complexity.

- Hundreds of unique designs for different project formats (from portfolio to business card site) or option to create a site from scratch.

- Creating mobile sites, Facebook pages or blogs.

- Many settings for changing images, colors and fonts of the site provide the ability to change the basic template.

- All elements of the web page are easy to edit, move and customize.

- Support for SEO options for search engine indexing.

- Basic Designer Options are free. Fees are provided to extend the functionality of the site.

Step 1: Register and choose a template

In order to use the Wix.com constructor, you need to go through a simple registration procedure, with only an e-mail and password. Then the user can immediately create a site by selecting the appropriate theme:

Wix.com offers a wide variety of themes for design: from photography, design, art to business, services and restaurants. Each theme contains professional templates, broken down by category. For example, the Music and Entertainment theme includes site templates for actors, theater, cinema, dance, events, musicians, and more. Often, the patterns overlap in different categories.

On the same page, you can organize your designs separately with Flash and Html5, as well as view the latest or most popular works. Near each layout there is a "Browse" button that lets you see what a future internet project will look like when it's finished.

If the user wants to apply their own design, there is a Blank option where you can select the layout of the elements and menus on the future site, the rest of the elements are added manually.

STEP 2. Edit and customize the template

After selecting the basic layout for the site, you need to click the "Edit" button to proceed with making your own changes to the finished design. The constructor enters edit mode, with many options and settings for the user.

You can change the finished template in one of the following ways:

Click on the item that needs change.

Use the left side menu, which consists of four items. Each item contains submenus that can be switched and edited.

The left side menu has four items:

Pages - Options designed to work with site pages. They can be moved, edited, added and deleted. The hierarchy of pages is supported as a random menu. You can also specify META data for each of the pages on the site.

Design - options allow you to change the background of the site, the colors, and fonts.

Add - a section with widgets that can be placed on a web page. These include text, photos, galleries, media, forms and lines, buttons and menus, social networks, e-commerce and applications.

Customization - here you can set a site address, add favorites, set SEO parameters for search engines, customize statistics.

Below is an example of how to change options in the selected template. Three different windows with parameters are opened at once - first general, then detailed.

Installing widgets does not require programming knowledge, plugin installation, etc. In the appropriate menu item, select the appropriate widget and simply drag the item to the page. Depending on your settings, the widget may be displayed on one or all pages of sites.

Features that are not in the constructor can be added using HTML code. But even without that, there are enough elements available for a standard full-fledged site. All elements of the page are provided in customized settings.

Step 3. Finish. Own domain and premium rates

To the top of the Wix.com editor are buttons to save changes and view created pages. Publication allows you to host a site on a server and make it accessible to all Internet users. Because Wix.com is free, therefore, a domain address restriction has been introduced that looks like:

http: //your\_login.wix.com/site\_name

This is certainly not the best solution, so it makes sense to connect your domain to the project. Unfortunately, this will require you to switch from a free account to premium rates. There are several paid offers on Wix.com:

Benefits of a paid account

- Obtaining a quality, fully customizable design.

- Added an administrative page with settings and the ability to modify the content of the site easily, just without specific knowledge.

- There is SEO functionality and many useful widgets.

- You can do the site support yourself or contact Wix.com Technical Support.

- As a bonus for premium rates, the domain is provided free of charge. Coupons are also available for advertising on Google and Facebook, which is important for starting your own online business.

Wix.com offers great deals for web developers. You can become a partner and make money from the income you get from your own site. For creative people, they offer a designer Wix.com. They get access to the Wix Pro Directory where the service administration directs clients looking for contractors.

**Work order**

- Check out Wix.com. You can use a different constructor to do the job.

- Choose a theme-appropriate template and create a full website in 3-4 pages.

- See the administrative panel.

- Edit texts, pictures, headings, background. Site Language - English!

- Add / remove existing services to the site.

- To publish a site on the Internet.

**The contents of the report**

- Name and purpose of laboratory work.

- Provide domain name, title and short description of the site you created

- Screenshots of the sequence of creating a site with a brief description of each of the steps.

- In conclusion, evaluate the usability of the constructor and the administrative panel.

**Questions**

1. What SaaS services for creating the website do you know?
2. What is the website template?
3. How to publish the website using Wix?

## LAB 2

CREATING THE WEBSITE USING BASIC HTML

**Work goal:** to learn how to create the basic webpage using HTML and basic inline CSS styling.

HTML stands for Hypertext Markup Language. It allows the user to create and structure sections, paragraphs, headings, links, and blockquotes for web pages and applications.

HTML is not a programming language, meaning it doesn’t have the ability to create dynamic functionality. Instead, it makes it possible to organize and format documents, similarly to Microsoft Word.

When working with HTML, we use simple code structures (tags and attributes) to mark up a website page. For example, we can create a paragraph by placing the enclosed text within a starting <p> and closing </p> tag.

HTML was invented by Tim Berners-Lee, a physicist at the CERN research institute in Switzerland. He came up with the idea of an Internet-based hypertext system.

Hypertext means a text that contains references (links) to other texts that viewers can access immediately. He published the first version of HTML in 1991, consisting of 18 HTML tags. Since then, each new version of the HTML language came with new tags and attributes (tag modifiers) to the markup.

According to Mozilla Developer Network’s HTML Element Reference, currently, there are 140 HTML tags, although some of them are already obsolete (not supported by modern browsers).

Due to a quick rise in popularity, HTML is now considered an official web standard. The HTML specifications are maintained and developed by the World Wide Web Consortium (W3C). You can check out the latest state of the language anytime on W3C’s website.

The biggest upgrade of the language was the introduction of HTML5 in 2014. It added several new semantic tags to the markup, that reveal the meaning of their own content, such as <article>, <header>, and <footer>.

HTML documents are files that end with a .html or .htm extension. You can view then using any web browser (such as Google Chrome, Safari, or Mozilla Firefox). The browser reads the HTML file and renders its content so that internet users can view it.

Usually, the average website includes several different HTML pages. For instance: home pages, about pages, contact pages would all have separate HTML documents.

Each HTML page consists of a set of tags (also called elements), which you can refer to as the building blocks of web pages. They create a hierarchy that structures the content into sections, paragraphs, headings, and other content blocks.

Most HTML elements have an opening and a closing that use the <tag></tag> syntax.

Below, you can see a code example of how HTML elements can be structured:

<div>

 <h1>The Main Heading</h1>

 <h2>A catchy subheading</h2>

 <p>Paragraph one</p>

 <img src="/" alt="Image">

 <p>Paragraph two with a <a href="https://example.com">hyperlink</a></p>

</div>

* The outmost element is a simple division (<div></div>) you can use to mark up bigger content sections.
* It contains a heading (<h1></h1>), a subheading (<h2></h2>), two paragraphs (<p></p>), and an image (<img>).
* The second paragraph includes a link (<a></a>) with a href attribute that contains the destination URL.
* The image tag also has two attributes: src for the image path and alt for the image description.

HTML tags have two main types: block-level and inline tags.

* Block-level elements take up the full available space and always start a new line in the document. Headings and paragraphs are a great example of block tags.
* Inline elements only take up as much space as they need and don’t start a new line on the page. They usually serve to format the inner contents of block-level elements. Links and emphasized strings are good examples of inline tags.

The three block level tags every HTML document needs to contain are <html>, <head>, and <body>.

* The <html></html> tag is the highest level element that encloses every HTML page.
* The <head></head> tag holds meta information such as the page’s title and charset.
* Finally, the <body></body> tag encloses all the content that appears on the page.

<html>

 <head>

 <!-- META INFORMATION -->

 </head>

 <body>

 <!-- PAGE CONTENT -->

 </body>

</html>

* Headings have 6 levels in HTML. They range from <h1></h1> to <h6></h6>, where h1 is the highest level heading and h6 is the lowest one. Paragraphs are enclosed by <p></p>, while blockquotes use the <blockquote></blockquote> tag.
* Divisions are bigger content sections that typically contain several paragraphs, images, sometimes blockquotes, and other smaller elements. We can mark them up using the <div></div> tag. A div element can contain another div tag inside it as well.
* You may also use <ol></ol> tags for ordered lists and <ul></ul> for unordered ones. Individual list items must be enclosed by the <li></li> tag. For example, this is how a basic unordered list looks like in HTML:

<ul>

 <li>List item 1</li>

 <li>List item 2</li>

 <li>List item 3</li>

</ul>

Many inline tags are used to format text. For example, a <strong></strong> tag would render an element in bold, whereas <em></em> tags would show it in italics.

Hyperlinks are also inline elements that require <a></a> tags and href attributes to indicate the link’s destination:

<a href="https://example.com/">Click me!</a>

Images are inline elements too. You can add one using <img> without any closing tag. But you will also need to use the src attribute to specify the image path, for example:

<img src="/images/example.jpg" alt="Example image">

If you want to learn more HTML tags, consider checking our complete HTML cheat sheet (which is also available for download).

Stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once. For example, a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.

While CSS is great for creating text styles, it is helpful for formatting other aspects of Web page layout as well. For example, CSS can be used to define the cell padding of table cells, the style, thickness, and color of a table's border, and the padding around images or other objects. CSS gives Web developers more exact control over how Web pages will look than HTML does. This is why most Web pages today incorporate cascading style sheets.

**Work order**

1. Create the document called about-me.html

2. Write down the simple structure of the document

3. Use title, h1, meta description and meta charset to add the topic of the page and it's basic parameters.

4. Add your photo (or random image) to the page.

5. Use the list structure (ul or ol) to do the list of your interests.

6. Make your usual time schedule using the table with three column (number, activity, time).

7. Make a list of your favorite websites to visit using the links/

8. Structure your page using minor headings (h2-h6)

9. Prettify your text using simple tags or stylings for it.

10. Put a background for your page

**The contents of the report**

- Name and purpose of laboratory work.

- Provide the website source code

- Provide the screenshots of the developed websites.

**Questions**

1. Name the basic HTML tags
2. How to use CSS inline?
3. How to use the background image?

## LAB 3

CREATING THE MULTIPAGE WEBSITE WITH LAYOUT

**Work goal:** to learn how to use display options and divs to create website layouts.

Float is a CSS positioning property. To understand its purpose and origin, we can look to print design. In a print layout, images may be set into the page such that text wraps around them as needed. This is commonly and appropriately called “text wrap”. Here is an example of that.

In page layout programs, the boxes that hold the text can be told to honor the text wrap, or to ignore it. Ignoring the text wrap will allow the words to flow right over the image like it wasn’t even there. This is the difference between that image being part of the flow of the page (or not). Web design is very similar.

In web design, page elements with the CSS float property applied to them are just like the images in the print layout where the text flows around them. Floated elements remain a part of the flow of the web page. This is distinctly different than page elements that use absolute positioning. Absolutely positioned page elements are removed from the flow of the webpage, like when the text box in the print layout was told to ignore the page wrap. Absolutely positioned page elements will not affect the position of other elements and other elements will not affect them, whether they touch each other or not.

Setting the float on an element with CSS happens like this:

#sidebar {

 float: right;

}

There are four valid values for the float property. Left and Right float elements those directions respectively. None (the default) ensures the element will not float and Inherit which will assume the float value from that elements parent element.

Aside from the simple example of wrapping text around images, floats can be used to create entire web layouts.

While floats can still be used for layout, these days, we have much stronger tools for creating layout on web pages. Namely, Flexbox and Grid. Floats are still useful to know about because they have some abilities entirely unique to them, which is all covered in this article.

Floats are also helpful for layout in smaller instances. Take for example this little area of a web page. If we use float for our little avatar image, when that image changes size the text in the box will reflow to accommodate.

This same layout could be accomplished using relative positioning on container and absolute positioning on the avatar as well. In doing it this way, the text would be unaffected by the avatar and not be able to reflow on a size change.

Float’s sister property is clear. An element that has the clear property set on it will not move up adjacent to the float like the float desires, but will move itself down past the float. Again an illustration probably does more good than words do.

In the above example, the sidebar is floated to the right and is shorter than the main content area. The footer then is required to jump up into that available space as is required by the float. To fix this problem, the footer can be cleared to ensure it stays beneath both floated columns.

#footer {

 clear: both;

}

Clear has four valid values as well. Both is most commonly used, which clears floats coming from either direction. Left and Right can be used to only clear the float from one direction respectively. None is the default, which is typically unnecessary unless removing a clear value from a cascade. Inherit would be the fifth, but is strangely not supported in Internet Explorer. Clearing only the left or right float, while less commonly seen in the wild, definitely has its uses.

One of the more bewildering things about working with floats is how they can affect the element that contains them (their “parent” element). If this parent element contained nothing but floated elements, the height of it would literally collapse to nothing. This isn’t always obvious if the parent doesn’t contain any visually noticeable background, but it is important to be aware of.

As anti-intuitive as collapsing seems to be, the alternative is worse. Consider this scenario:

If the block element on top were to have automatically expanded to accommodate the floated element, we would have an unnatural spacing break in the flow of text between paragraphs, with no practical way of fixing it. If this were the case, us designers would be complaining much harder about this behavior than we do about collapsing.

Floats often get beat on for being fragile. The majority of this fragility comes from IE 6 and the slew of float-related bugs it has. As more and more designers are dropping support for IE 6, you may not care, but for the folks that do care here is a quick rundown.

* Pushdown is a symptom of an element inside a floated item being wider than the float itself (typically an image). Most browsers will render the image outside the float, but not have the part sticking out affect other layout. IE will expand the float to contain the image, often drastically affecting layout. A common example is an image sticking out of the main content push the sidebar down below.

Quick fix: Make sure you don’t have any images that do this, use overflow: hidden to cut off excess.

* Double Margin Bug – Another thing to remember when dealing with IE 6 is that if you apply a margin in the same direction as the float, it will [double the margin](http://www.cssnewbie.com/double-margin-float-bug/). Quick fix: set display: inline on the float, and don’t worry it will remain a block-level element.
* The 3px Jog is when text that is up next to a floated element is mysteriously kicked away by 3px like a weird forcefield around the float. Quick fix: set a width or height on the affected text.
* In IE 7, the Bottom Margin Bug is when if a floated parent has floated children inside it, bottom margin on those children is ignored by the parent. Quick fix: using bottom padding on the parent instead.

**Work order**

1. Create the project folder.

2. Inside the project folder, create folder "css" to keep your styles and img for the images.

3. Inside project folder, create four html documents: index.html, gallery.html, contact.html and hobbies.html

4. Using guides from <https://www.w3schools.com/html/html_layout.asp> page, make the following layout for all the pages



5. Create the header, navigation and footer panels for all the pages you've created. Header should contain website name and page name, and should have the background. Navigation panel should interconnect all pages of the websites with links. Footer should contain the links for your social networks accounts and any general information you want. These three panels should have same content for all of the pages.

6. Keep all styles for sections above in one CSS file located in css folder.

**The contents of the report**

- Name and purpose of laboratory work.

- Provide the website source code

- Provide the screenshots of the developed websites.

**Questions**

1. Explain the advantages and disadvantages of float.
2. What is the common page layout?
3. How to achieve the proper layout with display options.

## LAB 4

USING BOOTSTRAP AS AN EXAMPLE OF GRID LAYOUT

**Work goal:** to learn how to use grid layout possibilities of the Bootstrap Framework.

It is an open-source and free CSS framework, which helps in directing a 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. This framework helps in faster web development and supports developers in creating responsive web pages faster.

### History of Bootstrap

Twitter Blueprint was the first name of Bootstrap and was developed at Twitter by Mr. Mark Otto and Jacob Thornton. It got released as an open-source product in August 2011 on GitHub. The framework was mainly designed for encouraging the reliability and uniformity of web pages across internal tools. Before Bootstrap's existence, for making responsive sites and interface development, various external libraries were used, which brought inconsistency and gave rise to the heavy maintenance burden.

### Why should developers use Bootstrap?

Here are some of the key usages of Bootstrap listed:

* Browser supportive: Every browser supports this Bootstrap Framework.
* Mobile-first approach: In the Bootstrap 3 framework, there is a preexisting mobile-first style all through the library and not as separate files.
* Simple and easy to start: If you know HTML and CSS, you can quickly start working with Bootstrap, and its documentation is provided on the official site.
* 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.
* Easy customization: It provides some built-in components and functionalities that are easy for customizing.
* Clean interface or Developers: Bootstrap framework provides a new and consistent result for building user interfaces in web pages.
* It is an open-source framework with web-based customization.

### Benefits of Bootstrap Framework

* It produces less cross-browser bugs.
* It is a consistent framework supported by all the browsers plus CSS based compatibility fixes.
* It is a lightweight and hence widely used framework for creating responsive sites.
* Looks, structure, and styles can be customized as per requirement.
* A simple and effective grid system.

**Work order**

Now, we should fill the website from lab 3 with content.

1. Add some information about you to the index.html (you can take this from lab2 website). Make your website good-looking with content like videos and images

2. On page contacts.html add form with fields "Name", "Subject", "Message" and "Send" button (you can use bootstrap classes).

3. On page hobbies.html pick 3 or more hobbies and create 3 blocks with photo and text describing each hobby.

4. On page gallery.html, add 8 or more photos, using rows and cols from Bootstrap, making page responsive.

Feel free to create something fresh and interesting! If you don't know how to implement that, just ask the teacher!

**The contents of the report**

- Name and purpose of laboratory work.

- Provide the website source code

- Provide the screenshots of the developed websites.

**Questions**

1. Explain the grid layout
2. Explain the usage of bootstrap on small screens
3. How to add bootstrap to your website?

## LAB 5

USING FLEX AS A MODERN WAY TO POSITION PAGES LAYOUT

**Work goal:** to learn how to use flex to create page layout.

As web design has progressed, there have been different standard ways to arrange elements on a website. CSS flexbox is a relatively new yet powerful way to create layouts and something every web developer and designer should be familiar with.

If you don’t know how to use it yet, this in-depth flexbox tutorial aims to change that. The post below will talk about what flexbox is, why it matters, and its underlying concept. After that, we will go over the CSS properties and values associated with flexbox in detail and finish up with an example of a use case.

Alright, we have a lot to cover and it’s going to be a bit technical, so put your CSS goggles on and let’s get going.

Flexbox stands for flexible box. It’s a layout module for CSS aimed at giving you an efficient way to arrange, organize, and size website elements to create highly adaptive designs.

Of course, the technology to place web components on a page is not new. Since the beginning of the Internet, web designers have used different ways to place images, text, and other content where they wanted it to be. However, these were either not suitable for responsive design (tables), were never intended for as a layout tool in the first place (float), didn’t allow you to define equal heights for elements (inline-block), or had other issues.

So, while designers and developers made do for a long time, there were still a bunch of design patterns that were either impossible or needed JavaScript to work. Common examples are vertical centering and equal columns, two of the holy grails of web design.

The way flexbox works is quite simple on the surface: You have a container (flex container) with children (any elements contained within, called flex items), which are placed along flex lines.

Lines and items can be manipulated in layout, size, spacing, and more along both the vertical and horizontal axis using a multitude of operators. This allows you to best take advantage of the available space and lets elements arrange themselves automatically according to it.

As mentioned, flexbox is a whole CSS module, not a single property. Therefore, it comes with a whole lot of its own operators, some for the parent container, some for its children.

To understand how they work, it’s important that you know the concepts and terminology of flexbox, which are displayed in the image above:

* main axis — This is the axis at which the items are laid out. Important: it can be both vertical or horizontal, depending on the flex-direction property.
* main-start, main-end — These represent the start and end point of where items are arranged.
* main size — This denotes either the width or height of the flex items, depending on the direction of the main axis.
* cross axis — The axis perpendicular to the main axis. Its direction, too, depends on how the main axis is defined.
* cross-start, cross-end — Start of and direction in which flex lines will be filled.
* cross size — Denotes the other dimension of flex items that is not defined by main size.
* writing-mode — Allows you to switch the direction of writing from left-to-right to right-to-left or even to vertical. It’s a work in progress with little to no browser support, however, it’s important to know for some of the properties further below.

As you can see, a lot about flexbox is rather abstract and not absolutely defined. Consequently, much of the CSS below is dependent on your setup.

While you can use flexbox to build entire web pages, however, that’s not the recommended use case. For larger layouts, consider Grid (more on that some other time). Flexbox, on the other hand, is most appropriate for small-scale layouts and applications, such as:

* navigation menus
* card layouts
* media items
* web forms

Flexbox was first proposed at the beginning of the past decade and recommended by the W3C for adoption in 2012. Since then, browsers have started supporting it and, by now, all modern browsers are able to deal with flexbox.

**Work task**

1. Redo the website you've did on Lab 4, by using flex

**The contents of the report**

- Name and purpose of laboratory work.

- Provide the website source code

- Provide the screenshots of the developed websites.

**Questions**

1. Explain the advantages of flex?
2. Which browsers support flex?
3. When we can use flex?

## LAB 6

USING MEDIA QUERIES FOR THE RESPONSIVE WEB DESIGN

**Work goal:** to learn how to use media queries to adjust the web page layout for different screens.

Media queries are a way to target browser by certain characteristics, features, and user preferences, then apply styles or run other code based on those things. Perhaps the most common media queries in the world are those that target particular viewport ranges and apply custom styles, which birthed the whole idea of responsive design.

/\* When the browser is at least 600px and above \*/

@media screen and (min-width: 600px) {

 .element {

 /\* Apply some styles \*/

 }

}

There are lots of other things we can target beside viewport width. That might be screen resolution, device orientation, operating system preference, or even more among a whole bevy of things we can query and use to style content.

Media queries are a powerful tool in your CSS toolbox with exciting hidden gems. But if you accomodate your design to every possible situation you’ll end up with a codebase that’s too complex to maintain and, as we all know, CSS is like a bear cub: cute and inoffensive but when it grows it will eat you alive.

That’s why I recommend following Ranald Mace’s concept of Universal Design which is “the design of products to be usable by all people, to the greatest extent possible, without the need for adaptation or specialized design.”

In “Accessibility for Everyone” Laura Kalbag explains that the difference between accessible and universal design is subtle but important. An accessible designer would create a large door for people on a wheel chair to enter, while a universal designer would produce an entry that anyone would fit disregarding of their abilities.

I know that talking about universal design on the web is hard and almost sound utopian, but think about it, there are around 150 different browsers, around 50 different combinations of user preferences, and as we mentioned before more than 24000 different and unique Android devices alone. This means that there are at least 18 million possible cases in which your content might be displayed. In the words of the fantastic Miriam Suzanne, “CSS out here trying to do graphic design of unknown content on an infinite and unknown canvas, across operating systems, interfaces, & languages. There’s no possible way for any of us to know what we’re doing.”

That’s why assuming is really dangerous, so when you design, develop and think about your products leave assumptions behind and use media queries to make sure that your content is displayed correctly in any contact and before any user.

**Work task**

1. Redo the website you've did on previous lab by adding media queries for different screens (minimum 320 px width)

**The contents of the report**

- Name and purpose of laboratory work.

- Provide the website source code

- Provide the screenshots of the developed websites.

**Questions**

1. Explain the usage of media queries
2. How the viewport works?
3. What is the minimum screen width to care about in modern design?

REFERENCES

1. CSS [Online]. Access mode: <https://techterms.com/definition/css>
2. What is HTML? The Basics of Hypertext Markup Language Explained [Online]. Access mode: <https://www.hostinger.com/tutorials/what-is-html>
3. All about floats[Online]. Access mode: <https://css-tricks.com/all-about-floats/>
4. Introduction to Bootstrap [Online]. Access mode: <https://www.w3schools.in/bootstrap4/intro/>
5. Flexbox tutorial for the beginners [Online]. Access mode: <https://torquemag.io/2020/02/flexbox-tutorial/>
6. A complete guide to media queries [Online]. Access mode: <https://css-tricks.com/a-complete-guide-to-css-media-queries/>