HyperText Markup Language

HTML&CSS

Lesson 7

Responsive Design

Contents

Lesson materials are attached to this PDF file. In order to get access to the materials, open the lesson in Adobe Acrobat Reader.

1. What is responsive design?

In today's world there are lots and lots of devices we use to go online: computers, tablets, smartphones, TVs, and even watches. And it will not be convenient for the user to browse a web page if its layout on the smartphone is the same as on the laptop: interface elements will be too small, or there will be a horizontal scrolling. The user is likely to leave such a site if the responsive design is not applied to the latter.

Let's look at an example of how the famous Google website would display if it did not have a responsive design.

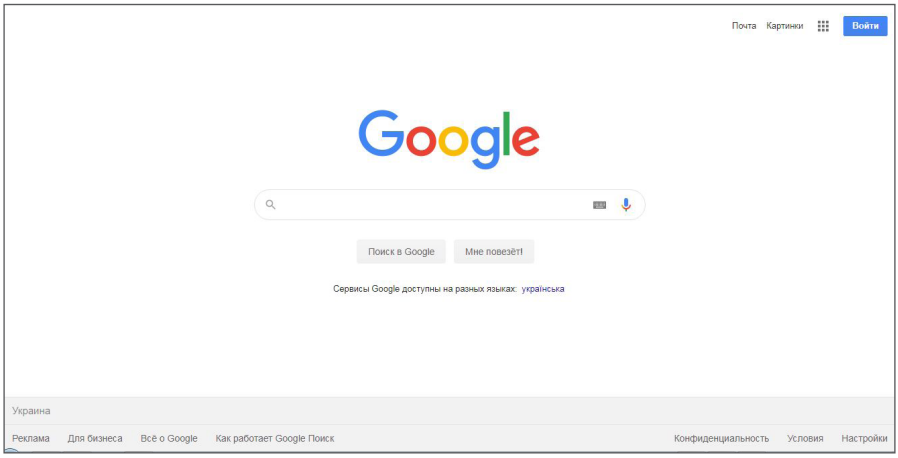

Figure 1. This is how Google looks on a laptop

In the Figure 1 above, all elements are conveniently arranged and take up the entire visible part of the browser window on the laptop.

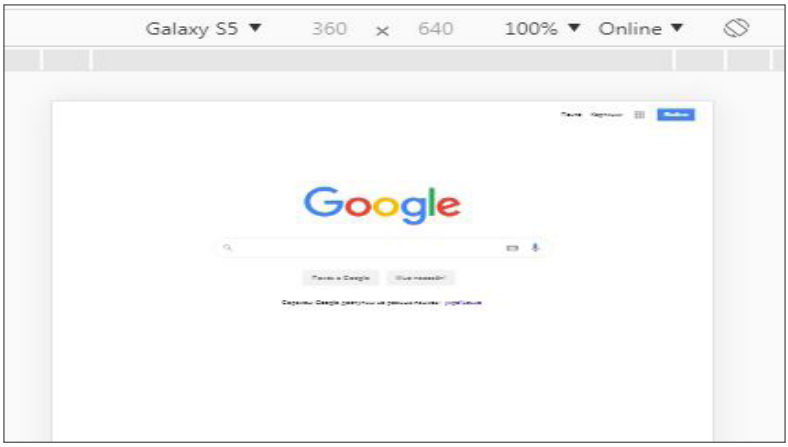

Figure 2. This is how non-responsive Google looks on Galaxy S5

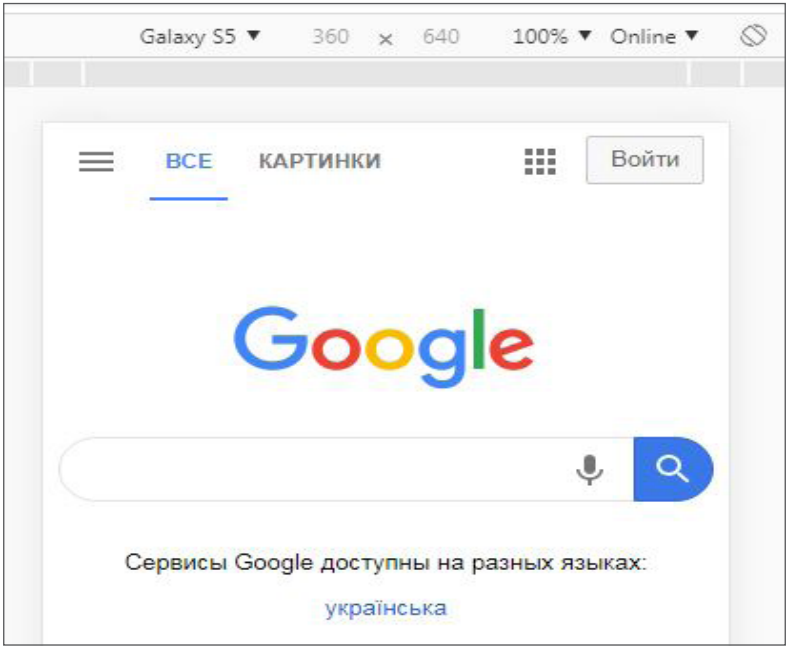

Figure 3. This is how responsive Google looks on Galaxy S5

In the Figure 2, all elements on the page are tiny and inconvenient for view or use.

The last Figure 3 shows how the responsive design has rebuilt the search engine page: all elements are conveniently arranged again, and their sizes correspond to the device.

Responsive design is a design that provides the correct display of a web page on various devices and smoothly changes page elements when resizing the browser window.

In the recent past it was customary to create a separate «mobile» version of the website, which was actually a copy of the content with a different layout of elements, and it led to problems in promoting such sites. After the responsive design appeared in the world of web development, layout versions for various devices were no longer required since we got a convenient arrangement of elements on mobile devices while having the same content.

Advantages of responsive design:

- There is no need to create a separate version of the website for a specific device;
- Universal presentation of web pages for various devices;
- All pages are accessible at the same url address, which eliminates problems in website promotion;
- Easy-to-use web page interface regardless of the gadget.

Disadvantages of responsive design:

- Slow website loading due to its weight; Full version of the website is loaded regardless of the device;
- Layout of web pages is more complex because all nuances of display on various devices should be taken into account;

- Inability to "disable" mobile display on a device with a small screen, you need to open the web page on a different device with a larger screen;
- Website testing takes more time and effort.

But responsive design is required even with its disadvantages because the number of mobile users and the variety of gadgets grow by the day.

2. Principles of Responsive Design

1. **Use relative units.** In order to make elements change smoothly based on the width of the browser window, we should use relative units for width, height, margins and paddings, font size. Relative units are %, em, ex, vh, vw, vmin.
2. **Use boundary values for the container width.** Content stretched to the entire width of the window looks good on a small device, but the same display on a widescreen device will cause discomfort. This is why it is recommended to use boundary values for width/height in absolute values, namely in pixels. The following properties are used for this: min-width/min-height and max-width/max-height.
3. **Use a grid structure.** CSS3 has various tools that allow you to build a flexible structure for the element arrangement. These tools are Flexbox and Grid Layout.
4. **Use media queries** to rebuild the display of elements on the page. To make the pages not just smoothly rescale the content based on the width of the browser window but rebuild the content to fit the screen for convenient view, we need to set breakpoints. Breakpoints are a physical parameter of a device based on which the current display is determined. Breakpoints are set with media queries.
5. **Begin layout with the mobile layout** and gradually move towards widescreen or vice versa. It is customary to begin layout of a responsive web page with small devices

since they have fewer elements, and the content display is simple and concise, and gradually move to larger screen sizes. But you can go the other way around, moving from a widescreen device to a mobile layout.

6. **Use system fonts.** Of course, you will use various fonts in your design. You should also remember that a font downloaded from the Internet slows down the loading of your page. System fonts are loaded instantly, which significantly speeds up the web page loading.
7. **Hide or replace elements on various devices.** Elements that are not informative (like a banner advertising a third-party resource) are often hidden on mobile devices or put outside the screen so that one can open them at any convenient time (like product filters).
8. **Adapt images and videos.** In order to make pages load faster, use images adapted to the current layout. For instance, smaller background images for mobile devices. Whenever possible, use vector images instead of raster.

If you want to see examples of responsive design, go to Media Queries (mediaqueri.es). It has a gallery of responsive design websites.

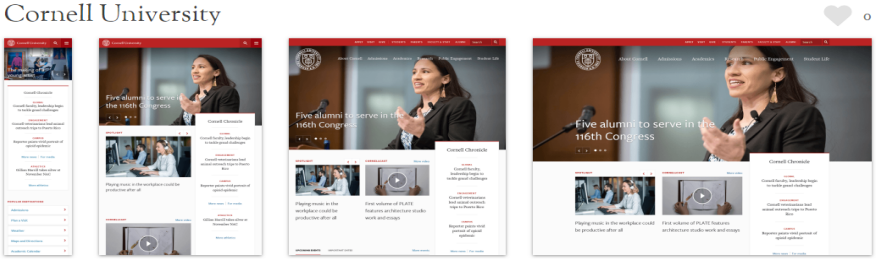

Figure 4

The Figure 4 shows that website elements are rebuilt based on the device and its size.

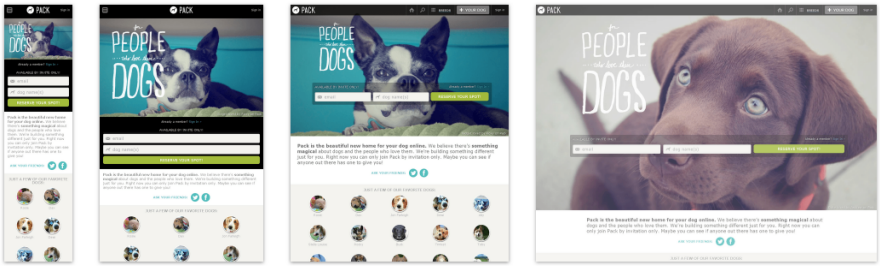

Figure 5

Interface elements rebuild to fit the device for more convenient use, and images load for faster website loading.

3. The viewport Meta Tag

A **viewport** is a visible area of the browser window into which the latter fits a web page. Each browser determines the viewport width in its own way. For instance, Safari has it as 980 pixels, and IE as 1024 pixels. On average, the width is about 1000 pixels because it is generally believed that web pages are designed for desktops.

The process of page display consists of the following stages: the browser receives a page from the server, the viewport sets its width and then proportionally resizes the page until it is the size of the device's screen.

In order to prevent the browser from rescaling the page taking its width as the viewport's default width, use the viewport meta tag.

Meta tags indicate information for browsers and search engines. The viewport meta tag tells the browser the scale in which the visible part of the page should be displayed on various devices.

The table 1 has the list of parameters and their values that the viewport meta tag can take.

Figure 1

Parameter	Value	Description
width	An integer in pixels or a device-width value that is equal to the screen width in CSS pixels at a scale of 100%	Sets the viewport width. Width in CSS pixels is not a physical resolution of the screen but the size regulating the size of a pixel. It is necessary to make elements look the same at a high pixel density.

Parameter	Value	Description
height	An integer in pixels or a device-height value equal to the screen hight in CSS pixels at a scale of 100%	Sets the viewport height.
initial-scale	A real number from 0.1 and higher	Sets the initial zoom level of the initial size of the viewport. (1.0 — no scaling).
user-scalable	no/yes	Disallows/allows the user to scale the page.
minimum-scale	A real number from 0.1 and higher	Sets the minimum zoom level of the viewport. (1.0 — no scaling).
maximum-scale	A real number from 0.1 and higher	Sets the maximum zoom level of the viewport. (1.0 — no scaling).

In order for the browser to understand that the page is adapted for various devices, we need to write the viewport meta tag with the following values:

```
<meta name="viewport" content="width=device-width,
      initial-scale=1">
```

The values specified in the content attribute are default values. They should be changed if your website should have, say, the minimum width of at least 450 pixels. Then the value of the meta tag

```
<meta name="viewport" content="width=450,
      initial-scale=1">
```

will actually set this minimum width for the viewport. If the screen width is greater than 450 pixels, the browser will expand the viewport rather than zoom it in.

The second example is used when the scale settings should remain the same when switching the orientation:

```
<meta name="viewport" content="initial-scale=1,
      maximum-scale=1">
```

There is currently no established standard for the tag <meta name="viewport"> in the CSS3 specification.

4. Media Queries

A media query is, in fact, a conditional construct that requests characteristics from the device that displays the web page and executes a set of style rules if the received characteristics of the device correspond to the ones set in the condition of the current media query.

Media queries are used in responsive design when it is necessary to apply various CSS styles for display on different devices taking into account their characteristics. For example, the Figure below shows how the page is rebuilt from device to device. Notice the search box. On the desktop, it is in the upper right corner of the page, remaining in the same place when displaying search results. On the mobile device, this element is at the bottom of the page because it is the most convenient place for quick entry of the information you need.

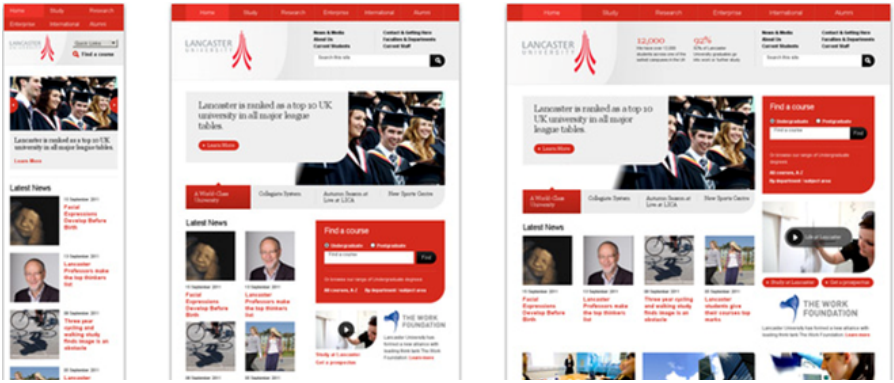

Figure 6

A media query consists of the device type (optional) and technical characteristics of this device.

Media queries can be applied in the following ways:

1. Using the link tag:

```
<link rel="stylesheet" media="screen and
      (min-width:900px)" href="width_900.css">
```

2. Using the @import at-rule:

```
@import url(width_900.css) screen and (min-width:900px);
```

3. Using the @media at-rule specified inside the style tag or in the style file:

```
@media screen and (min-width:900px){

   /* styles for the specified type of the device and its
      characteristics
   */

}
```

Device types:

- all — all devices (default);
- print — print preview mode;
- screen — monitor screens;
- speech — speech synthesizers.

CSS2 had a larger list of device types. It included projectors, TVs, tablets, and other devices that are considered obsolete in the modern standard.

Below is the table 2 with device characteristics that are a mandatory part of a media query.

Figure 2

Name	Description
aspect-ratio	The width to height ratio. For instance, (aspect-ratio: 12/5).
color	The number of bits per color component.
color-gamut	Checks the color gamut supported by the device: rgb(color-gamut:srgb), p3(color-gamut:p3), BT.2020(-color-gamut:rec2020).
color-index	Checks whether the device uses a color chart. Value: an integer.
grid	Checks if the output device is grid or raster. If the output device is a grid (for instance, ATM or a telephone screen with one fixed font), the value will be 1. Otherwise, 0.
height	Height of the viewport. Set in absolute or relative units.
monochrome	The number of bits per pixel of a monochrome device. Set with an integer.
orientation	Device orientation: orientation: portrait or orientation: landscape.
overflow-block	Describes the behavior of the device when content overflows the initial containing block in the block axis. Values: overflow-block:none, overflow-block:scroll, overflow-block: optional-paged, overflow-block: paged.
overflow-inline	Describes the behavior of the device when content overflows the initial containing block in the inline axis. Values: overflow- inline:none, overflow-block:scroll.
resolution	Screen resolution: the number of dots (pixels) per inch (dpi) or centimeter (dpcm).
scan	Checks the device rendering process: scan:interlace and scan:progressive.
update	Checks the possibility to update content after it has been rendered (like CSS animation): update:none, update:slow, and update:fast.
width	Viewport width. Set in absolute or relative units.

A media query may include a wide-ranging check of device characteristics. A combined media query is created with logical operators:

1. and combines several media functions into one media query. The query is executed only if all media functions correspond to the device characteristics. For example:

```
@media (min-width:320px) and (max-width:480px){}
```

   This query will be executed if the screen width is in the range from 320 to 480 pixels.

2. comma — combines multiple media queries into one rule. Each query is processed separately from the others, so the styles are applied if at least one query matches the device characteristics. For example:

```
@media screen and (aspect-ratio: 16/9),
       screen and (aspect-ratio: 16/10){}
```

   This query is executed for screens with the aspect ratio of width to height equal to 16/9 or 16/10.

3. not is used to invert a media query. For example:

```
@media not (color){}
```

   This query will be executed for devices that do not support colors.

4. only is used for a style only if the entire query corresponds. For instance, it hides styles for older versions of browsers.

Examples of generating media queries taking into account the device's type and characteristics:

1. width/height are the most popular options for media queries that read the device's width and height:

Device and Parameters	Query
smarphones with the screen width in the range from 320 to 480 pixels	@media only screen and (min-width:320px) and (max-width:480px) { /* styles */ }
smartphones with the maximum screen height equal to 600 pixels	@media only screen and (max-height:600px) {/* styles */ }

2. orientation (landscape and portrait) checks the device's orientation if there is any and enabled:

Device and Parameters	Query
smartphones with landscape orientation	@media screen and (orientation: landscape) { /* styles */ }
Preview on a printer with portrait orientation	@media print and (orientation: portrait) {/* styles */ }

3. color checks if the device can display various colors:

Device and Parameters	Query
devices that do not support many colors, such as a black and white printer	@media not (color){ /* styles*/ }
devices that support a 4-bit color or less	@media (max-color: 4) {/* styles */}

4. proportions check the proportions of the screen, namely the screen's width to height ratio:

Device and Parameters	Query
Widescreen monitors with a proportional width to height ratio of 16/9	@media screen and (aspect-ratio: 16/9){ /* styles*/ }
Square monitors with the proportional width to height ratio of 1/1	@media screen and (aspect-ratio: 1/1){ /* styles*/ }

5. **resolution** checks the resolution of the device — the number of dots per inch or centimeter:

Device and Parameters	Query
Devices with the resolution of 150dpi (dots per inch)	@media (max-resolution: 150dpi){ /* styles*/ }

6. **complex media** queries are used when you need to check several parameters or apply the same styles to different devices and their technical characteristics:

Device and Parameters	Query
the device has the minimum height of 680 pixels, or it is a screen device in a portrait mode	@media (min-height: 680px), screen and (orientation: portrait){ /* styles */ }

Let's consider how media queries are implemented through the example of a small web page.

Create a page that displays the main elements: header, main split into three columns, footer. Write tag names and their positions for the desktop layout as content.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content=
          "width=device-width, initial-scale=1.0">
    <title>Media</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <header>header top</header>
        <main>
            <aside>aside left</aside>
```

```
            <section>section center</section>
            <aside>aside right</aside>
        </main>
        <footer>footer bottom</footer>
    </div>
</body>
</html>
```

This is what we got:

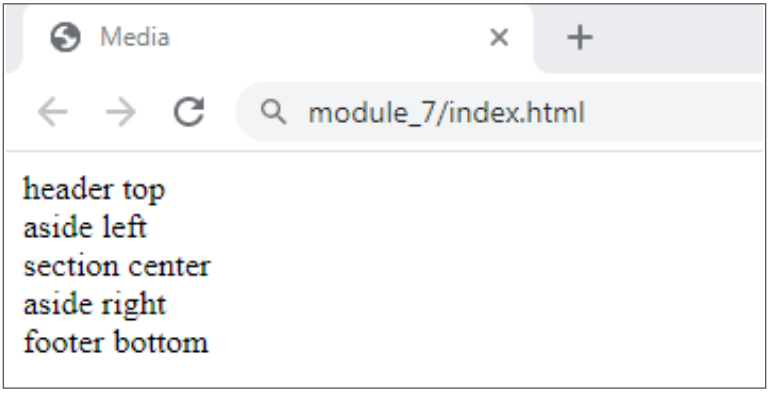

Figure 7

The Figure 7 shows that all elements are arranged one below the other because they are block tags. Let's get down to styles.

```
/* refer to all elements to remove default paddings
   and margins assigned by the browser
*/
{
    padding: 0;
    margin: 0;
}
```

```
/* we will use a combination of fixed and fluid layout
   for the main container
*/
.container{
    /* set the main width equal to the width
       of the browser's viewport
    */
    width: 100vw;
    /* set the minimum width down to which the container
       can be scaled; if the device's width is smaller,
       a horizontal scroll will appear
    */
    min-width: 320px;
    /* set the maximum width up to which the container
       can be scaled; if the device's width is bigger,
       margins will appear on the sides
    */
    max-width: 1200px;
    /* specify horizontal margins in order
       for the container to be always in the center
    */
    margin: auto;
    /* if there is not enough content so that the page
       could completely fill the browser's window height,
       set the minimum height
    */
    min-height: 100vh;
    /* if there is enough content for a vertical scroll
       to appear, set the height equal to the content
    */
    height: auto;
    /* for easier arrangement of elements on the page,
       use the flexbox technology
    */
    display: flex;
    /* change the direction of the main axis to vertical */
    flex-direction: column;
}
```

```
/* set a different background color to each container
   element for better clarity
*/
header{
    background-color: pink;
    /* set the header size along the main axis */
    flex-basis: 10vh;
}

/* main is the main element on a page, it is both flex
   parent and flex container
*/
main{
    /* to fill the entire space along the main axis
       with the main element, write the flex property
    */
    flex: 1 1 auto;
    display: flex;
}

main>aside{
    background-color: cyan;
    /* set the size of the left side along the main axis
       in percent
    */
    flex-basis: 20%;
}

main>section{
    background-color: coral;
    /* to fill the entire space along the main axis
       with the main element, write the flex property
    */
    flex: 1 1 auto;
}

main>section+aside{
    background-color: lime;
```

```
    /* set the size of the left side along the main axis
       in percent
    */
    flex-basis: 20%;
}

footer{
    background-color: gray;
    /* set the footer size along the main axis */
    flex-basis: 5vh;
}
```

The display for a widescreen monitor is as follows now:

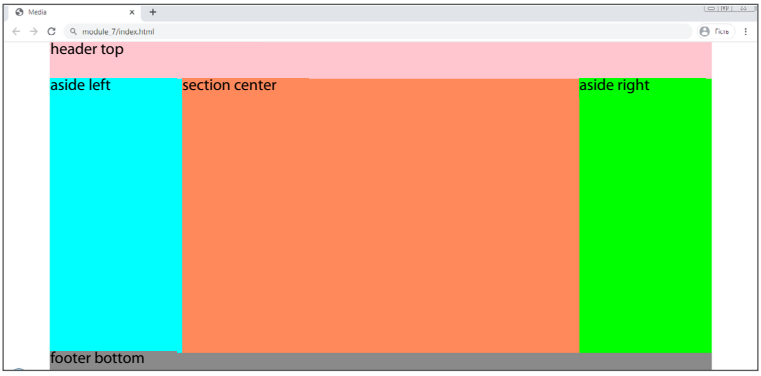

Figure 8

Let's move on to media queries. Because we began with a widescreen layout, let's now go down in size and end with a mobile device. For this, let's create two media queries to display on square screens and smartphones.

```
/* query for the screens with the maximum width
   of 960px and less*/
@media screen and (max-width:960px){
```

```
    main>section+aside{
        /* hide the element on the right on all devices
           whose width is 960px and less
        */
        display: none;
    }

    main>aside{
        /* set the size of the left side along the main
           axis in percent
        */
        flex-basis: 30%;
    }

    main>section{
        /* set the size of the central part along the main
           axis in percent
        */
        flex-basis: 70%;
    }
}

/* query for screens with the maximum width of 570px
   and less
*/
@media screen and (max-width:570px){
    main{
        /* change the direction of the main axis
           of the element that displays the main content
         */
        flex-direction: column;
    }

    main>aside{
        /* set a size to the visible element named aside
           along the main axis
        */
```

```
        flex-basis: 20vh;
    }

    main>section{
        /* set a size to the main content along
           the main axis*/
        flex-basis: auto;

        /* change the order of element display so that
           the content is higher */
        order:-1;
    }

}
```

So, we got two more types of layout. For square monitors:

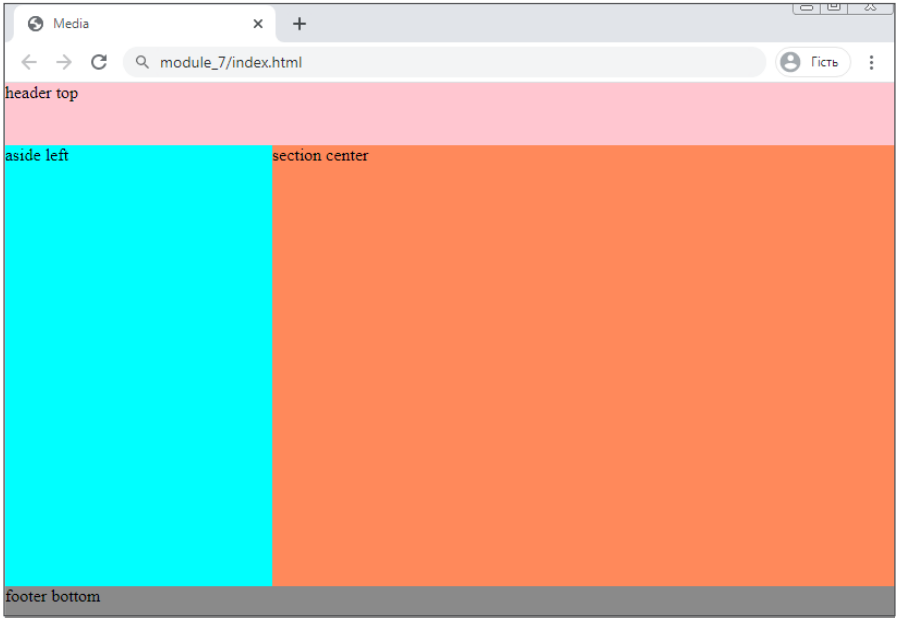

Figure 9

And for smartphones:

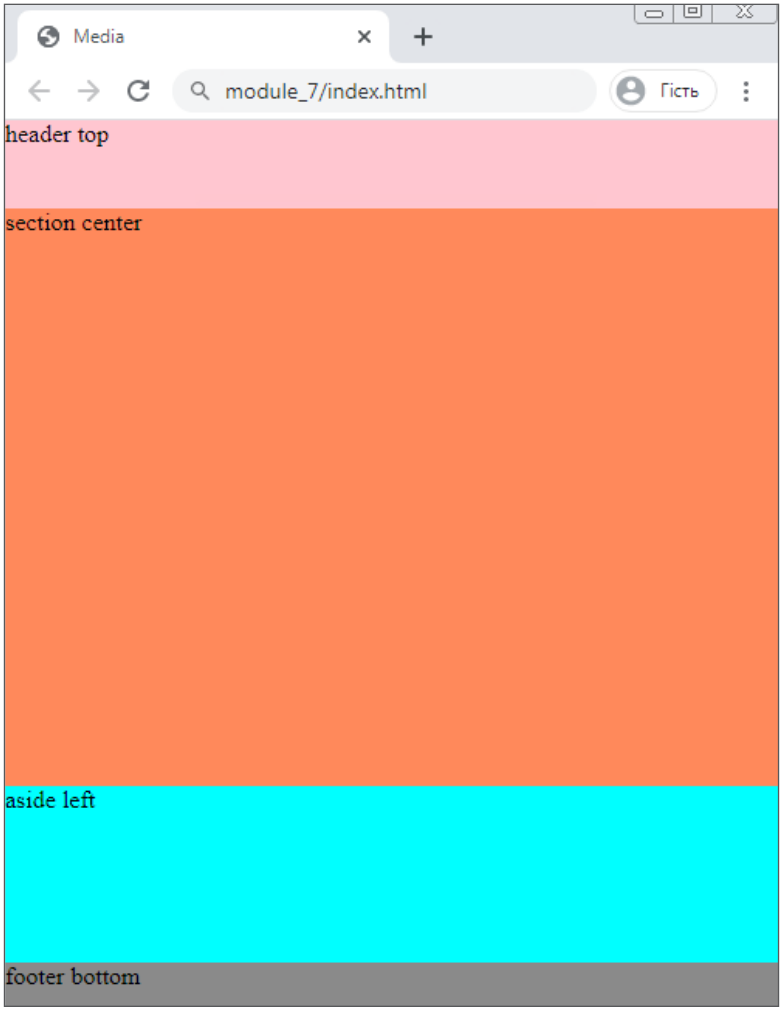

Figure 10

▷ You can view the full code in the files attached to the lesson or in CodePen at https://codepen.io/MethCont/pen/WNrrVNW.

Let's consider an example of creating a small page of a news site. Wide screens will display content in three columns, square screens in two, and mobile ones in one column.

The Figure 11 provides the layout on a widescreen monitor.

Figure 11

Let's create an index.html page with the following content:

1. The header includes the website name and the main menu.
2. The main part of the website consists of:
   - ▷ a column with the latest news on the left,
   - ▷ main content — an article with a title, image, and social media icons in the center,
   - ▷ a column of quotes on the right.
3. A website footer has a copyright sign and the current year.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content=
         "width=device-width, initial-scale=1.0">
    <title>Media</title>
    <!-- add the Bitter font from fonts.googleapis.com -->
    <link href="https://fonts.googleapis.com/
                css2?family=Bitter&display=swap"
          rel="stylesheet">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
       <header>
          <h1>Media</h1>
          <nav>
             <a href="">Home</a>
             <a class="active" href="">News</a>
             <a href="">Gallery</a>
             <a href="">Contact</a>
          </nav>
       </header>
       <main>
          <aside>
             <div>
                <h3>Last News</h3>
                <article class="news">
                   <strong>business</strong>
                   <h2>
                     <a href="">Conseptur eos
                        ipsam possimus</a>
                   </h2>
                  <em>by Modi Cumque</em>
                   <p>Mollitia modi cumque architecto
                      commodi illum, conseq sed nihil
                      error  aliquam  accusamus.
                   </p>
                   <strong>2 hours ago</strong>
                </article>
```

```
            <article class="news">
               <strong>art</strong>
               <h2>
                  <a href="">Lorem architecto
                     commodi illum</a>
               </h2>
               <em>by Timus Onihil</em>
               <p>Dolore ipsam possimus mollitia
                  modi cumque architecto commodi
                  illum, consequuntur!</p>
               <strong>4 hours ago</strong>
            </article>
         </div>
      </aside>
      <section>
         <article>
            <h2>Conseptur eos ipsam possimus</h2>
            <em>by Modi Cumque</em>
            <img src="images/urbanization.jpg" alt="">
            <p>Facilis minus alias consequuntur
               esse autem ducimus quaerat, delectus
               obcaecati quidem voluptatibus corrupti
               ex, aliquam, error quisquam dicta ut.
               Eum quasi quidem fugit corporis
               labore velit voluptatibus consectetur,
               laudantium cupiditate?</p>
            <p>Tempora earum iure corporis iste
               ea voluptates impedit doloribus
               minima, error nisi.Eum cum sunt
               inventore nesciunt quod dicta?
               Nostrum consectetur odit sit nulla
               ad odio debitis rem sapiente animi
               impedit aut tenetur autem vitae
               inventore numquam dolore iste quos,
               architecto sed? </p>
            <p>Repellat quisquam numquam a
               inventore dolores, modi quod autem
```

```
            voluptatibus! Lorem ipsum dolor sit
            amet consectetur adipisicing elit.
            Architecto, et. Tempora, doloremque
            inventore? Minima fugiat quis quas
            ipsam voluptates adipisci?</p>
          <div class="social">
            <a class="fb" href=""></a>
            <a class="tw" href=""></a>
            <a class="inst" href=""></a>
          </div>
        </article>
      </section>
      <aside>
        <div>
          <h3>Osed accusamus</h3>
          <figure>
            <img src="images/unnamed.png" alt="">
            <figcaption>
              <p>Consectetur eos sunt, dolore
              ipsam possimus mollitia modi cumque
              architecto commodi illum!</p>
              <em>by Modi Cumque</em>
            </figcaption>
              </figure>
              <figure>
                <img src="images/unnamed.png" alt="">
                <figcaption>
                  <p>Odio in, delectus commodi
                     sapiente nostrum ratione
                     porro laborum facere quasi
                     cumque!</p>
                  <em>by Timus Onihil</em>
                </figcaption>
              </figure>
          </div>
        </aside>
    </main>
```

```
        <footer>
            <p>&copy; 2020</p>
        </footer>
    </div>
</body>
</html>
```

All elements are arranged on the page in the order of their description, like in the Figure 12.

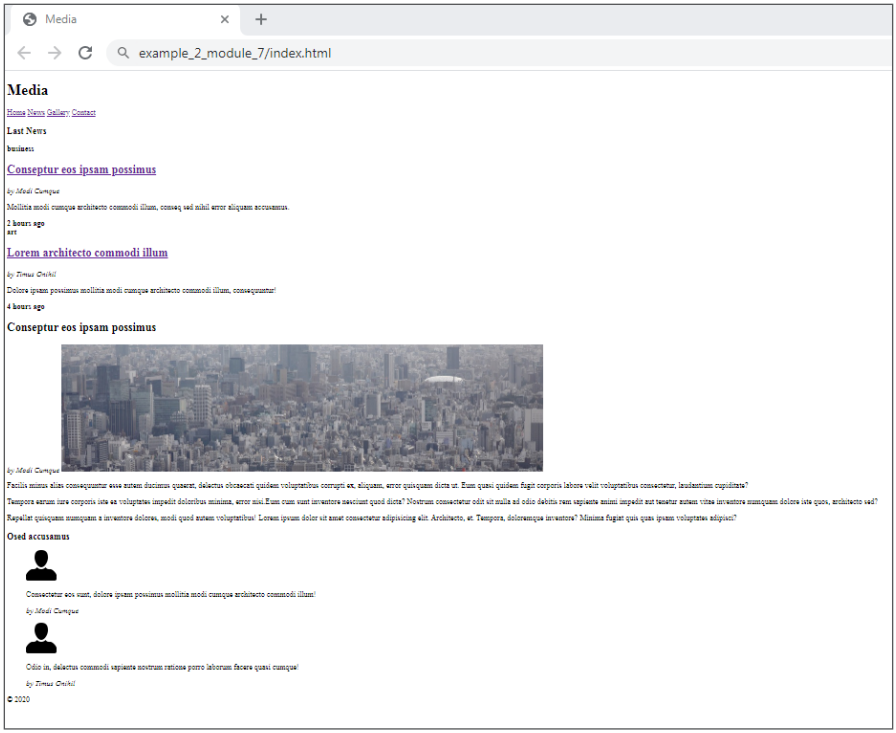

Figure 12

Let's create a style file. First, style the main layout, in our case, it is a layout for wide screens.

```
/* refer to all elements to remove default paddings
   and margins assigned by the browser
*/
*{
    padding: 0;
    margin: 0;
}

html {
    /* set the main font size in order to use
       the relative unit rem
    */
    font-size: 16px;
}

body{
    /* hide the content that exceeds the width
       of the browser window
    */
    overflow-x: hidden;
    /* set the main font of the content, let it be
       a system font to make the page load faster
    */
    font-family: Helvetica, Tahoma, Verdana, sans-serif;
    /* the rem unit sets the size relative to the font
       size of the <html> element
    */
    font-size: 1rem;
    color: rgb(75, 75, 75);
}

h1,h2,h3,a{
    /* the font for titles and links is 'Bitter' added
       through the <link> tag
    */
    font-family: ‘Bitter’, serif;
    font-weight: 400;
```

```
    color: rgb(105, 105, 105);
}

/* the font for titles and links is 'Bitter' added through
   the <link> tag
*/
.container{
    /* set the main width equal to the width
       of the  browser's viewport
    */
    width: 100vw;
    /* set the minimum width down to which the container
       can be scaled; if the device's width is smaller,
       a horizontal scroll will appear
    */
    min-width: 320px;
    /* set the maximum width up to which the container
       can be scaled; if the device's width is bigger,
       margins will appear on the sides
    */
    max-width: 1200px;
    /* specify horizontal margins in order for
       the container to be always in the center
    */
    margin: auto;
    /* if there is not enough content so that the page
       could completely fill the browser's window height,
       set the minimum height
    */
    min-height: 100vh;
    /* if there is enough content for a vertical scroll
       to appear, set the height equal to the content
    */
    height: auto;
    /* for easier arrangement of elements on the page,
       use the flexbox technology
    */
```

```
    display: flex;
    /* change the direction of the main axis to vertical */
    flex-direction: column;
}

/* set a different background color to each container
   element for better clarity
*/
header{
    /* set the header size along the main axis */
    flex-basis: 10vh;
    /* website's header elements are arranged from left
       to right
    */
    display: flex;
    /* set an indent between elements */
    justify-content: space-between;
    /* paddings on the left and right depend
       on the browser's window width
    */
    padding: 0 3vw;
}

header h1{
    /* align the content of the child element vertically */
    align-self: center;
}

header nav{
    /* align the content of the child element vertically */
    align-self: center;
    /* for easier arrangement of links, use the flexbox
       technology again
    */
    display: flex;
}
```

```
/* set styles for links in the main menu */
header nav a{
    display: block;
    margin-left: 2vw;
    padding: 0.5vw 1vw;
    /**/
    font-size: 1.1rem;
    text-decoration: none;
    border: 2px solid transparent;
}

header nav a:hover,
header nav a.active{
    border-bottom: 2px solid rgba(105, 105, 105, 0.5);
}

/* main is the main element on a page, it is both
   flex parent and flex container
*/
main{
    /* to fill the entire space along the main axis with
       the main element, write the flex property
    */
    flex: 1 1 auto;
    display: flex;
    /* child elements are arranged from left to right */
    justify-content: space-between;
    padding-top: 2vh;
}

main>aside{
    /* set the size of the left side along the main axis
       in percent
    */
    flex-basis: 20%;
}
```

```
main>aside>div{
    /* set paddings as a percentage of the width
       of the parent element
    */
    padding: 3% 10%;
}

main>aside>div>*{
    border-bottom: 2px solid rgba(105, 105, 105, 0.3);
    margin-bottom: 2vh;
}

main>aside>div>h3{
    text-transform: capitalize;
}

.news{
    font-size: 0.9rem;
    padding-bottom: 5px;
}

.news>*{
    display: block;
    padding-bottom: 7px;
}

.news>h2>a{
    font-size: 1.1rem;
    text-decoration: none;
}

.news>strong,
.news>em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}
```

```
.news>strong{
    text-transform: uppercase;
}

main>section{
    /* to fill the entire space along the main axis with
       the main element, write the flex property
    */
    flex: 1 1 55%;
}

main>section>article{
    padding: 1% 5%;
}

main>section>article>*{
    display: block;
    padding-bottom: 1.5vh;
}

main>section>article>h2{
    text-transform: uppercase;
}

main>section>article>em{
    color: rgb(155, 155, 155);
    font-size: 0.9rem;
}

main>section>article>img{
    /* set the image width to 100% of the width
       of the parent element
    */
    width: 100%;
}
```

```
.social{
    text-align: right;
}


/* add social media using an image consisting of three
   icons. In this case, one image loads instead of three,
   so the site loading speed is higher
*/
.social>a{
    /* change the link display from inline to inline-block */
    display: inline-block;
    /* the link size is equal to the size of one icon */
    width:32px;
    height: 32px;
    /* if the path to the image has no spaces or special
       characters, it can be written without quotes
    */
    background-image: url(images/social_icons.png);
    background-repeat: no-repeat;
    /* set the size so that only one icon is displayed */
    background-size: 300%;
    opacity: 0.7;
}


/* change the positioning of the background image
   according to the class for the social network
*/
.social>a.fb{
    background-position: 0 0;
}

.social>a.tw{
    background-position: 50% 0;
}
```

```
.social>a.inst{
    background-position: 100% 0;
}

main>section+aside{
    /* set the size of the left side along the main axis
       in percent
    */
    flex-basis: 20%;
}

main>section+aside figure *{
    display: block;
    padding-bottom: 7px;
}

main>section+aside figure img{
    opacity: 0.5;
    width: 32px;
    height:32px;
}

main>section+aside figure figcaption{
    font-size: 0.9rem;
}

main>section+aside figure figcaption em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}

footer{
    /* set the footer size along the main axis */
    flex-basis: 5vh;
    display: flex;
    flex-direction: column;
```

```
    justify-content: center;
    align-items: center;
}
```

The page look as follows now:

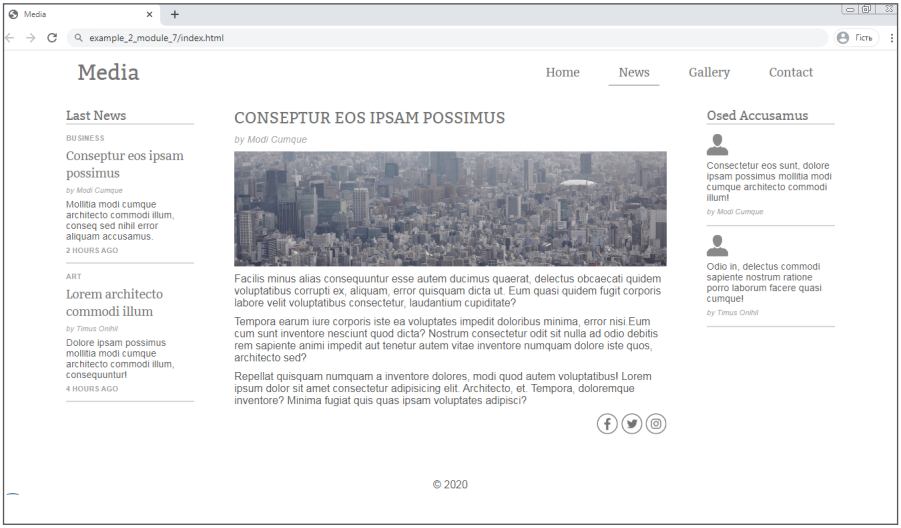

Figure 13

Let's create media queries:

```
/* query for screens with the maximum width of 960px
   and less
*/
@media screen and (max-width:960px){
    main>section+aside{
        /* hide the element on the right on all devices
           whose width is 960px and less
        */
        display: none;
    }
```

```
    main>aside{
        /* set the size of the left side along the main
           axis in percent
        */
        flex-basis: 30%;
    }
    main>section{
        /* set the size of the central part along
           the main axis in percent
        */
        flex-basis: 70%;
    }
}

/* query for screens with the maximum width of 570px
   and less
*/
@media screen and (max-width:570px){
    header{
        flex-basis: auto;
         /* change the direction of the main axis
            of the website header
         */
        flex-direction: column;
    }
    main{
        /* change the direction of the main axis
           of the element that displays the main content
        */
        flex-direction: column;
    }
    main>aside{
        /* set a size to the visible element named aside
           along the main axis
        */
        flex-basis: 20vh;
    }
```

```
    main>aside>div {
        padding: 3% 5%;
    }
    main>section{
        /* set a size to the main content along the main
axis*/
        flex-basis: auto;
        /* change the order of element display so that
           the content is higher
        */
        order:-1;
    }
}
```

Now the page looks as follows for square screens:

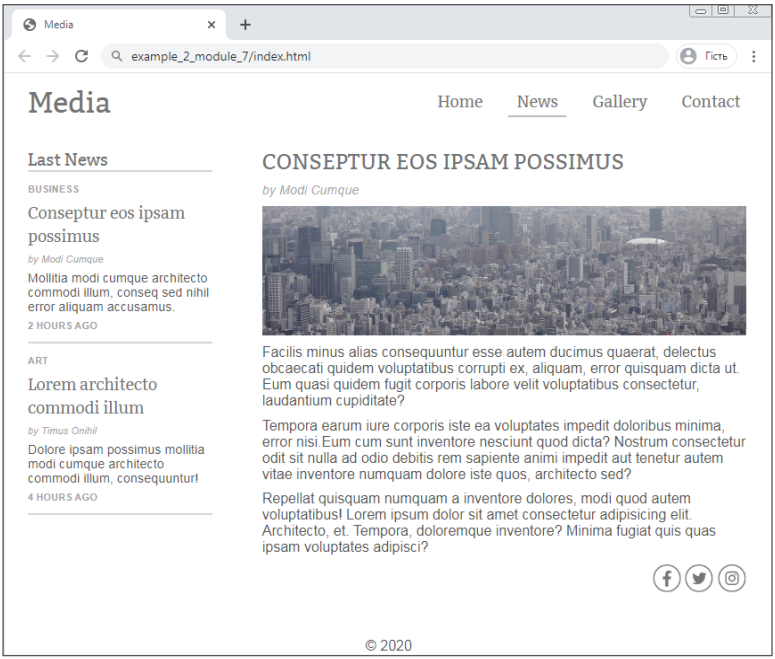

Figure 14

And as follows for mobile devices:

Figure 15

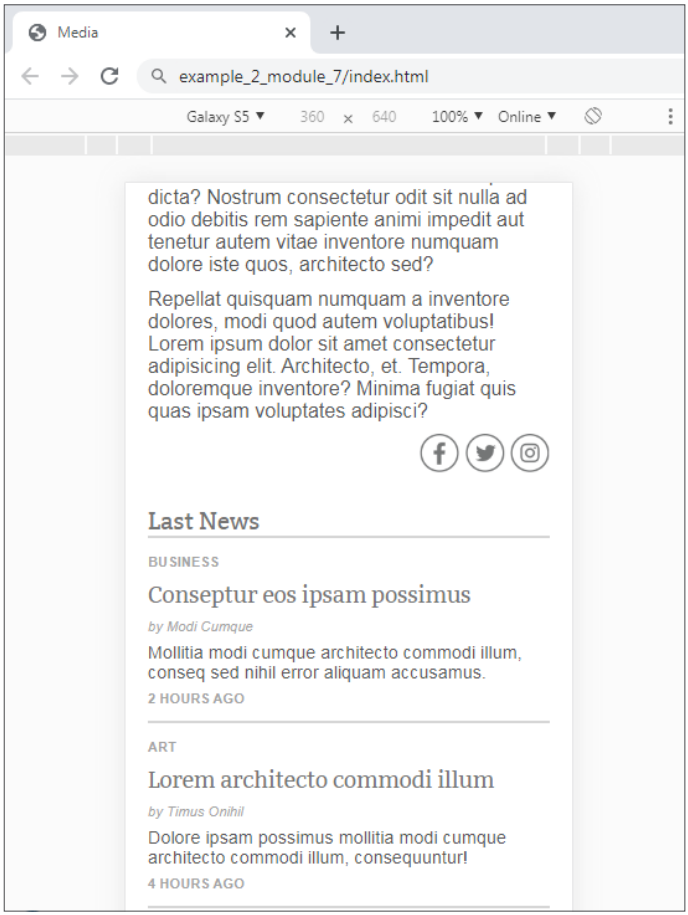

Figure 16

- **You** *can view the full code in the files attached to the lesson or in CodePen at https://codepen.io/MethCont/pen/bGEEXNZ.*

Homework

Make a page layout according to the design provided in the Figure below.

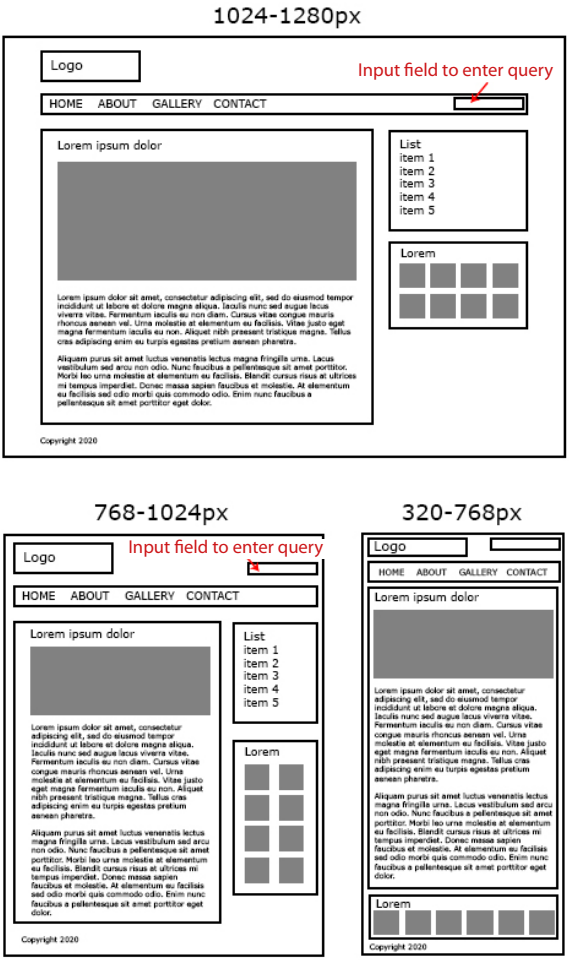

Figure 17

Lesson 7.
Responsive Design