CSS Navigation Bar

Topics Covered

- Navigation bar in CSS
    - Vertical Navigation Bar
    - Horizontal Navigation Bar
- Responsive Web Design
    - Viewport
    - Media Queries
    - Fluid Grids
    - Flexible Visuals

Topics in Detail

Navigation Bar in CSS

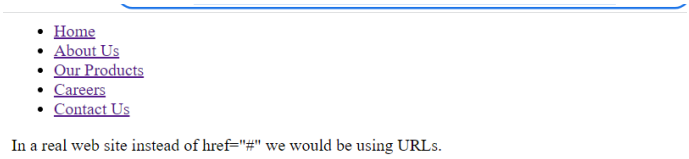
- Navigation bar comes under **Graphical User Interface.**
- It helps the webpage user to access information by **navigating** to other sections of the website using **links**.
- Usually, the navigation bar is on the **top** of the web page as a **horizontal list of links**.
- It can be placed **below the header or logo**, but it must always be placed **before** the main content.
- A website with **easy-to-use** navigation bar allows the user to visit any section easily and quickly.
- **Standard HTML** is the base for the Navigation Bar.
- Basically, **Navigation Bar** is a **list of links**.

Example

In this code let us try to list the menu by using <ul> and <li> tags.

```
<html>
<!--  <header>…
<body>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
    <p>In a real web site instead of href="#" we would be using URLs.</p>
</body>
</html>
```

Output



- Navigation Bar should not have list markers so we set ***list-style-type: none;*** to remove the bullets.
- To remove browser default settings, we set ***margin: 0;*** and ***padding: 0;***

Types of Navigation Bar

- Vertical Navigation Bar
- Horizontal Navigation Bar

Vertical Navigation

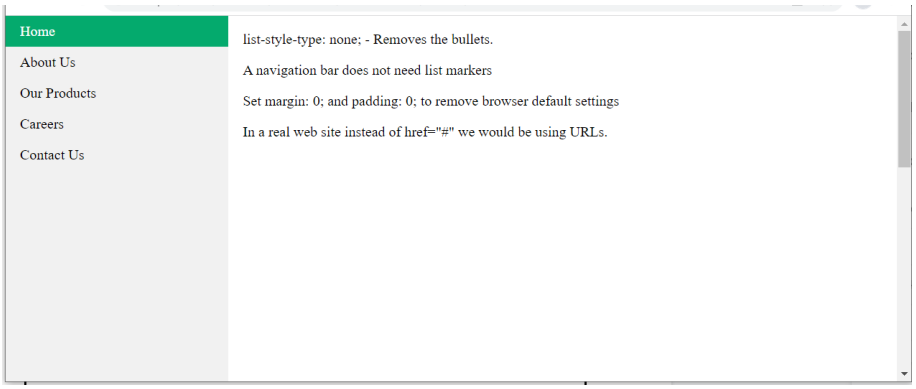
HTML

```
<body>
    <ul>
        <li><a class="active" href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
    <div style="margin-left:25%;padding:1px 16px;height:1000px;">
        <p>list-style-type: none; - Removes the bullets.</p>
        <p> A navigation bar does not need list markers</p>
        <p>Set margin: 0; and padding: 0; to remove browser default settings</p>
    <p>In a real web site instead of href="#" we would be using URLs.</p></div>
</body>
```

Style

```
body {margin: 0;}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}

li a.active {
  background-color: #04AA6D;
  color: white;
}

li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
```

Output



- ***display: block;*** - We display the links as block elements which allows us to specify the width, padding, margin, height, etc. In the Vertical Navigation Bar, the menu should be a part of a web page.
- ***width: 25%;*** - By default block elements take up the full width available. So We have to specify the width. The width can be set in the <ul> tag as well.

```
li a {
  display: block;
  color: #000;
  width: 25%;
  padding: 8px 16px;
  text-decoration: none;
}
```

- Set the navigation bar **background color** to **gray** and when the user moves the cursor over the menu the background color of the link has to change to **dark gray**.

```
li a:hover {
  background-color: #555;
  color: white;
}
```

- **Active/Current Navigation Link**
  - We have added an "**active**" class to the current link so that the user will get to know the current web page he is viewing.

```
li a.active {
  background-color: #04AA6D
  color: white;
}
```

- Set ***overflow: auto;*** to enable scroll if the side nav has more content.

- We can make the Vertical Navigation Bar **stick** to one position even on a **scroll** by setting ***position: fixed;***

```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}
```

Horizontal Navigation Bar

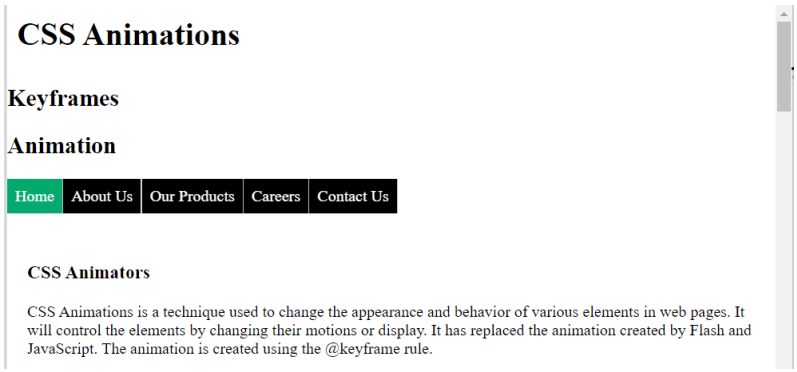
HTML

```html
<body>
  <h1 style="padding:10px">CSS Animations</h1>
  <h2>Keyframes</h2>
  <h2>Animation</h2>
    <ul>
        <li><a class="active" href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
   <div style="padding:20px;margin-top:10px;height:1500px;">
    <h3>CSS Animators</h3>
<p>CSS Animations is a technique used to change the appearance
and behavior of various elements in web pages.
It will control the elements by changing their motions or display.
It has replaced the animation created by Flash and JavaScript.
The animation is created using the @keyframe rule.</p>

<p>It has two parts,
CSS Properties (describe the animation of the elements)
keyframes (specific time intervals at which the animations have to occur)
When the animation is created in the @keyframe rule, it must have
a selector otherwise, the animation will have no effect.</p>

<p>@keyframe
Keyframes are the foundation of CSS Animations.
It will control the intermediate steps in a CSS animation sequence.
It defines the display of animation at the corresponding
stages in the whole duration.</p>
</div>
</body>
```

Style

```css
<style>
  body {margin:0;}
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    top: 0;
    position: sticky;
    width: 100%;
  }

  li{
    float: left;
    border-right:1px solid #bbb;
  }
  li a {
    display: block;
    color: white;
    text-align: center;
    text-decoration: none;
    padding: 8px;
    background-color: black;
  }
  li a.active {
    background-color: #04AA6D;
    color: white;
  }

  li a:hover:not(.active) {
    background-color: #555;
    color: white;
  }
  </style>
```

Output



- To create a horizontal navigation bar there are two ways.
- They are
    - Inline List Items
    - Float List Items

Inline List Items

- Horizontal Navigation Bar is created by specifying <li> element as **inline**.

```
li{
  display: inline;
}
```

- The line breaks before and after each item in the list is removed to display in one line.
- elements are displayed as **block** by default.

Float List Items

- Horizontal Navigation Bar is created by specifying <li> element as ***float: left;***.
- Setting ***float: left;*** will make the block elements float next to each other.
- ***display: block;*** - We display the links as block elements which allows us to specify the width, padding, margin, height, etc.

```
li{
  float: left;
}

li a {
  display: block;
  padding: 8px;
  background-color: black;
}
```

- Set the navigation bar **background color** to **black** and when the user moves the cursor over the menu the background color of the link has to change to **gray**.

```
li a {
  display: block;
  color: white;
  text-align: center;
  text-decoration: none;
  padding: 8px;
  background-color: black;
}
li a:hover {
  background-color: #555;
  color: white;
}
```

- **Active/Current Link Navigation**
  - We have added an "**active**" class to the current link so that the user will get to know the current web page he is viewing.
  - The current Link will appear in **dark green** color.

```
li a.active {
  background-color: #04AA6D;
  color: white;
}
```

- Adding ***border: right;*** property to divide the links

```
li{
  float: left;
  border-right:1px solid #bbb;
}
```

- We can make the Navigation Bar **stay** in one position either top or bottom even when **scrolled** by setting ***position: fixed;***

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  top: 0;
  position: fixed;
  width: 100%;
}
```

- Fixed position might **not work properly** on **mobile devices**.
- **Sticky Navbar**
  - Sticky elements **switch** between **relative** and **fixed** position depending on the position of the **scroll**.

- It behaves **relative** until it reaches a specific position and later behaves **fixed**.

```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  top: 0;
  position: -webkit-sticky; /* Safari */
  position: sticky;
  width: 100%;
}
```

- Sticky Position is **not supported by IE**. For **Safari,** it requires a **-webkit- prefix**.
- For Sticky position to work we have to specify **top, right, bottom or left**.

Responsive Web Design

- Web Pages can be viewed on **any device** like mobile, laptop, tablet, etc. Regardless of the device, our web page should **look good** and **easy to use** for this we go for **Responsive Web Design**.
- Responsive Web Design uses only **HTML** and **CSS** to create a web page that looks good on all devices.
- While viewing on **smaller devices**, the web page should not leave any information. Instead, the content should **fit** any device.
- To **resize, hide, shrink, enlarge or move** the content using **HTML and CSS** to look good on any screen is called **Responsive Web Design**.

Viewport

- The **Visible area** of a web page is called **Viewport**.
- The viewport differs from device to device. **Mobile phones** have **smaller** viewports when compared to computers.
- Initially, Web pages are designed **only** for **computers** by having **static design** and **fixed size**.
- **Fixed-size** web pages are **too large** for the viewport of **mobiles and tablets**.
- **<meta>** tag in **HTML5** helped web designers by taking control over the **viewport**.

```html
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

  - **width=device-width** - Sets the web page width to screen-width of the device.
  - **initial-scale=1.0** - Sets the initial zoom level of the page when loaded by the browser for the first time.
- Elements with **large fixed widths** should not be used.
- The content **should not** rely on particular **viewport width** to render well.

- Page elements **should not** have **large absolute CSS width**. Instead, use **relative** width values such as **width: 100%. Large absolute values** cause the element to fall **outside** the viewport.

Ingredients of RWD

- Media Queries
- Fluid Grids
- Flexible Visuals

Media Queries

- Media Query is a technique in **CSS3**
- If a condition is true, the **@media** rule is used to include a block of CSS properties.
- **Media queries** are used to make images smaller on all mobiles but larger on other devices like ipads, desktops, etc.

```
body {
  background-color: lightgreen;
}

@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
```

- If the browser window is within 600px the background color is light blue or else the background color will be light green.

Flexible Grid / Fluid Grid

- The **rearrangement of columns** themselves to fit the screen size is possible only by **Flexible Grid**. These Flexible Grids are created using CSS.

Flexible Visuals

- Images will scale up and down if **width: 100%;** and **height: auto;** is set.

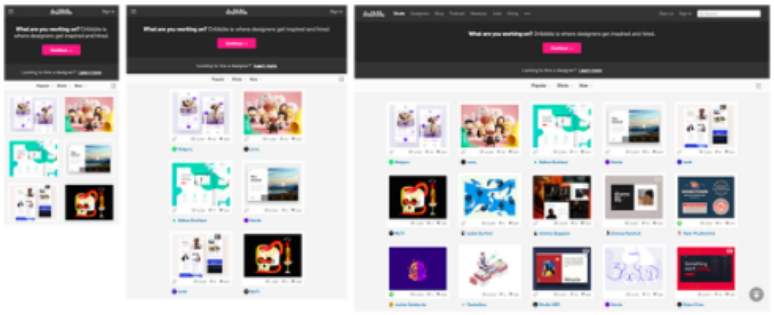
```
img {
  width: 100%;
  height: auto;
}
```

Real Time Examples of RWD

- **DropBox**

  - DropBox uses **fluid grid** and **flexible visuals** to achieve this responsive website.

- **Dribbble**



  - Dribbble website uses a **flexible grid** that condenses from 5 columns on computers to 2 columns on mobiles and tablets.