**HTML**

Basic code of html: peas shift+!+enter

Structural Elements

These elements just define the structure of html page.

They don’t provide the browsers and search engine like google and bing, what kind of data is being rendered by the html page.

For example: <div>, <span> etc.

On the other hand, semantic elements define the structure as well as give meaning to the content of html page.

They help the browsers and search engine like google and bing, what kind of data is being rendered like which part is for navigation, which part is the main content, which part is the footer etc.

For exam: <header>, <nav> , <main> etc.

**HTML Elements**

<form> ,<input> ,<textarea> ,<lable> ,<fieldset> ,<legend> ,<select>, <optgroup>, <option> ,<button> ,<datalist> ,<output>

**HTML5 Input Types:**

Button, checkbox, color, data, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel , text, time, url, week.

**HTML5 Input Attributes:**

Autocomplete, autofocus, form, formaction, formenctype, formnovalidate, formtarget, height and width, list, min and max, multiple, pattern(regexp), placeholder, required, step

Required: the input field must be fill-up before submit the form. Without fill-up you can’t submit the form.

Maxlength: maxlength is the length of the password.

Min and Max:

<input class="input-item" type="number" name="age" min="18" max="46">

Google logo: <https://53.fs1.hubspotusercontent-na1.net/hubfs/53/image8-2.jpg>

Facebook logo:

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSXtoQVLNAnKvtCbd3OZJRFZBY2XAzorW37VfLVvaDAX95ZwMr2XCL7vpg4ikIZxBbvlvU&usqp=CAU.jpg

github:

<https://banner2.cleanpng.com/20180711/xzq/kisspng-github-computer-icons-github-icon-5b45aecdf2f3c2.8525743315312933899951.jpg>

linkedin:

<https://w7.pngwing.com/pngs/977/831/png-transparent-linkedin-icon-hd-logo.png>

**What is SVG?**

#SVG stands for Scalar Vector Graphics

#SVG is markup language for describing two dimensional based vector graphics.

#SVG images and their related behaviors are defined in XML files. These files can be created and edited with any text editor and with drawing software like Adobe premiere pro.

To design SVG we use the <svg> element

The HTML <svg> element is a container for SVG Graphics it defines a new coordinate system to design these graphics.

It is used as the outermost element of SVG documents, but it can also be used to embed a SVG fragment inside an HTML document.

**CSS**

Different States for Links

1. Link- this is the state where link is not visited.
2. Visited- This is the state when link is visited.
3. Hover-This is the state when mouse is hovered on the link.
4. Active- This is the state when the link is clicked.

HTML positioning elements

1. Static
2. Relative
3. Absolute
4. Fixed
5. Sticky

Static:

1. Default position value
2. It positions elements in the normal order on the webpage.

Relative:

1. It helps to position html elements relative to their normal position
2. It provides access to top,right,bottom and left properties.

Absolute:

1.It helps to position html element relative to the parent element.

2. the parent element needs to have a position other static or else the absolute element is positioned relative to the main html element.

Fixed:

1. It helps to position html elements relative to the main html element. This results in elements being fixed at the defined position.
2. The position fixed elements are not affected by the scroll position of the webpage.

Sticky:

1. It is a combination of position relative and position fixed. It requires a condition (top, right, bottom, left ) and based on this condition it toggles between position relative and fixed.

**Handling Overflows**

**\*Visible**

**\*Hidden**

**\*Scroll**

**\*Auto**

**Shadow 2px 2px 2px red**

**Here 2px horizontal**

**2px vertical**

**2px baring**

 text-shadow: 2px 2px 2px red, 2px 6px 2px green;

**First shadow (2px 2px 2px red)**

* **2px → move right by 2 pixels (horizontal shift)**
* **2px → move down by 2 pixels (vertical shift)**
* **2px → blur radius (makes the shadow a little soft)**
* **red → shadow color is red**

**Second shadow (2px 6px 2px green)**

**2px → move right by 2 pixels (horizontal shift)**

**6px → move down by 6 pixels (vertical shift — more than before)**

**2px → blur radius again**

**green → shadow color is green**

**single selectors**

**HTML:**

**P{**

**Font size:24px;**

**Color:blue;**

**}**

**ID:**

**#cover-pic {**

**Width:400px;**

**Height:400px;**

**}**

**Class:**

**.message {**

**Font size:24px;**

**Color:red;**

**}**

**Combinators:**

**Combinators define relationship between two selectors**

1. **Descendant Selector:’ ‘**
2. **Child Selector: ‘>’**
3. **Adjacent sibling selector:’+’**
4. **General sibling Selector:’~’**

**Descendant Selector**

**Syntax: Selector\_a selector\_b;**

**For eg: div p{…};**

**Selectors all the elements matching selector\_b which are descendants of the elements matching selector\_a.**

**Adjacent Sibling Selector**

**Syntax: selector\_a+ selector\_b;**

**For eg, div + .menu-items**

**Selects the element matching selector\_b which is right next to the elements matching selector\_a. In another words the adjacent siblings.**

**General Sibling Selector**

**Syntax: Selector\_a ~selector\_b;**

**For eg, div~h3**

**Selects all the elements matching selector\_b which are the siblings of the elements matching selector\_a.**

**Pseudo classes:**

**Menu:link {**

**Color:black;**

**}**

**Menu:visited{**

**Color:green;**

**}**

**Menu:hover{**

**Color:red;**

**}**

**Menu:active{**

**Color:blue;**

**}**

**Pseudo Elements:**

**Pseudo Element is used to style a specific part of an HTML element.**

**::after , ::before, ::first-letter, ::first-line , ::selection**

**::first-line**

1. **It is used to add styles to the first line of some text. This text could be paragraphs (<p>), headings (<h1>-<h6>) etc.**
2. **This pseudo-element only works with block level elements.**

**::Selection**

1. **It is used to add styles to the selected text.**
2. **In another words, it is used to style the text highlight.**

**::before and ::after**

1. **::before is used to add some content before the content html element.**
2. **::after is used to add some content after the content of html element.**
3. **This content could be Text, Image or Blank.**

**Responsive Web Design:**

**Responsive design refers to the idea that your website should display equally well for all screen sizes from widescreen monitors to mobile phones.**

**Viewport Meta Tag:**

**The viewport determines how content is scaled and sized when it is rendered on different devices.**

**We have a viewport meta tag to control the width and scaling of the viewport.**

**Without this viewport meta tag, mobile devices render pages at desktop screen width, and then scales down the pages to fit the mobile screens.**

**Media Queries:**

**Media queries are simple filters that can be applied to CSS styles.**

**They make it easy to change styles based on the characteristics of the device rendering the content. These characteristics could be display type, width, height, orientation, and even resolution.**

**Most of the cases you will just be working with the device width condition.**

**@media screen and (max-width:600px) { //Styles for device with max width 600px;**

**}**

**Syntax for Media Queries:**

**P{**

**Font-size:24px;**

**Color:red;**

**}**

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

**P{**

**Font-size: 16px;**

**Color: blue;**

**}**

**}**

**@media screen and (min-width:992px) and (max-width:1200px) {**

**//Styles for device with width ranging from 992px to 1200px**

**}**

**@media screen and (max-width:600px) {….}**

**The above media query handles devices with width less than 600px.**

**For eg, Mobile phones.**

**@media screen and (min-width:600px) {….}**

**This media query handles devices with width greater than 600px. For eg, Large Screen Mobiles and Tablets in portrait Mode.**

**@media screen and (min-width:768px) {….}**

**This media query handles devices with width greater than 768px.**

**For eg, Tablets in Landscape mode.**

**Introduction to Flex Layout:**

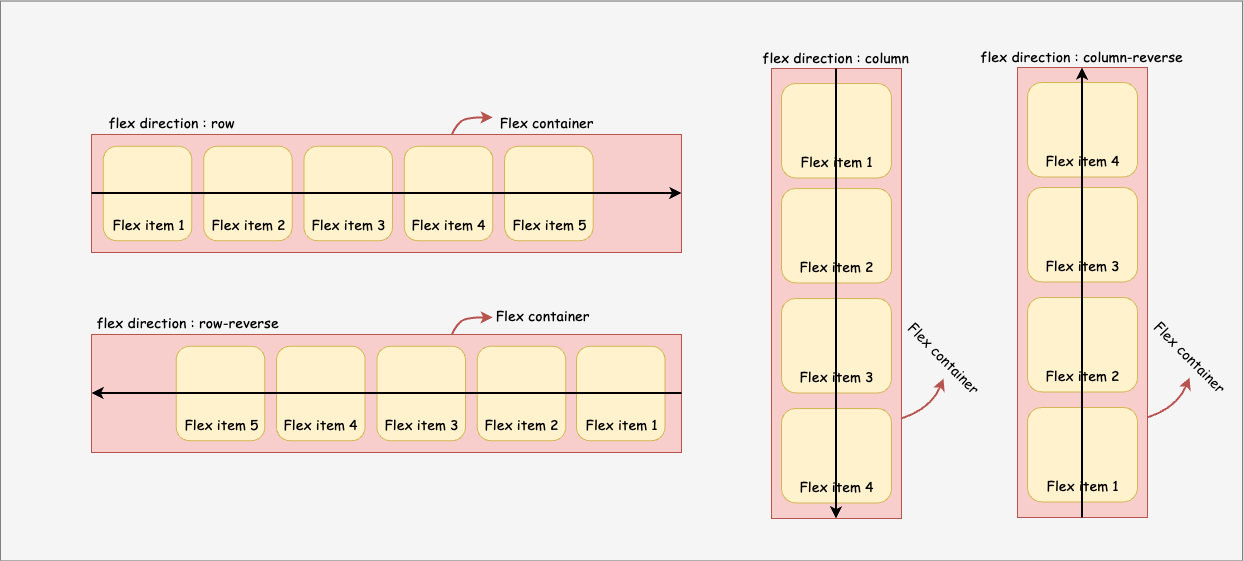
**The flex layout gives container the ability to alter its items’ width and height to best fill the available space.**

**A flex container expands items to fill available free space, or shrinks them to prevent overflow. This makes it perfect for designing layouts for which the size is unknown or flexible.**

**Flex layout Direction :  
Horizontal and vertical**

**Main -axis and cross-axis for direction-row**

**Main-axis is in the direction of flex-direction value whereas, cross-axis is the perpendicular to the flex-direction value.**



**Property: align-items**

**It is used to align items along the cross-axis.**

**It works only when flex is wrapped around multiple lines.**

**Transform property:**

**The transform property allows you to visually manipulate an element by translating, rotating, scaling and skewing.**

**It is used to translate or move elements across x-axis, y-axis and z-axis.**

1. **x-axis-translateX()**
2. **y-axis-translateY()**
3. **z-axis-translateZ()**

**Animation property:**

**These properties can be used to animate many css properties such as color, background-color, width,height, border-radius etc.**

**Each animation needs to be defined with the @keyframes rule which is then called with the animation property. Each @keyframes rule defines what should happen at specific moments during the animation where, 0% is the beginning of the animation and 100% is the end.**

**Animation-name 🡪 animation-duration🡪 animation-timing-function🡪animation-delay🡪 animation-direction🡪animation-iteration-count🡪 animation-fill-mode🡪animation-play-state.**

**Grid layouts**

**It is the very first css module created specifically to solve the layout problems.**

**It offers a grid-based layout system, with rows and columns, making it easiers to design web pages**