~~Fundamentals of Faith by Tanu Gupta~~

~~Vite + React~~

HTML and HTML5 Basics

<h1> <h2>, <h3>, ... <h6> with closing tags.

<!-- -->

<main></main> <header></header> <footer></footer>

<img src="" alt="">

<nav></nav>

<a href="" target=""> (target="_blank" opens a new tab)

for internal links

Make dead links using # symbol

<input type="" name="" value="" placeholder="">

<form action="" ></form>

<button type="submit"></button>

<input type="text" required> will make the input field required.

```
<label for="indoor">
<input id="indoor" type="radio" name="">
</label>
```

For a radio group name should be same for all radios.

```
<label for="loving">
    <input id="loving" type="checkbox" name="">
</label>
```

For a checkbox group name should be same for all checkboxes.

→ If you omit the "value" attribute the default value submitted is on. Hence it is important to include "value" attribute in radios & checkboxes.

→ You can set a checkbox or radio button to be checked by default as follows:-

<input type="radio" name="" checked>

<!DOCTYPE html> makes it HTML5 document.

<html></html>

<head></head>

<body></body>

link, meta, title and style tags go inside <head> tag.

Basic CSS:-

style can be inline, internal or external stylesheet
- internal → inside <head>
- inline → inside element tags.

Style attribute's properties:-

```
style = "color: blue;"
```

```
<style>
  h2 {
    color: red;
  }
</style>
```

Classes

```
<h2 class="blue-text"></h2>
```

```
<style>
  .blue-text{
    color: blue;
  }
</style>
```

```
font-size: 30px;
font-family: sans-serif;
```

Google fonts. (Also can use by downloading & installing)

Generic font families:-

- monospace
- serif
- sans-serif

Font degrading

If one font isn't available it will select the another one provided.

font-family: Helvetica, Sans-serif.

↓ if this isn't available this is selected.

width: 500px or width: 100%

Borders

```
border-color: red;
border-width: 5px;
border-style: solid;
border: 5px solid red;
border-radius: 10px (rounded corners)
border-radius: 50%. (circular border)
```

Element's background-color using:-

```
background-color: green;
```

"id" attribute:-

```
<h2 id="a">
#a{
    css
}
```

Padding :-

padding : 10px (All sides)

padding: 10px 20px
↓ top bottom ↓ left right

padding: 10px 20px 30px
↓ top ↓ left right ↓ bottom

padding: 10px 20px 30px 40px
↓ top ↓ right ↓ bottom ↓ left

padding-top:
padding-bottom:
padding-right:
padding-left:

Margin :-

margin : 10px (All sides)

margin : 10px 20px
↓ top bottom ↓ left right

margin : 10px 20px 30px
↓ top ↓ left right ↓ bottom

margin: 10px 20px 30px 40px
↓ top ↓ right ↓ bottom ↓ left

margin-top:
margin-bottom:
margin-left:
margin-right:

If you set element's margin to negative value, the element will grow larger.

Attribute Selectors :-

Example :-

```
[type='radio'] {
    margin: 20px 0px 20px 0px;
}
```

Absolute vs Relative units:

px, in, mm (absolute)

em, rem (relative to parent element's size)

Multiple classes

<h1 class="" "" "" "" ""></h1>

→ Inline styles override all CSS declarations in style element.

The most powerful method to override CSS and make sure its always this style that is applied is by using a keyword "!important"

color: red !important;
will override all styles.

Hexadecimal representation of colours.

color: #0f3219
↓ ↓ ↓
red green blue

In short #ffffff can be written as #fff
↓ ↓ ↓
r g b

RGB values to rep colours

from rgb(0,0,0) to rgb(255,255,255)
↓ ↓
black white.

CSS Variables:-

To create a CSS variable, you just need to give it a name with two hyphens in front of it and assign a value.

Like,

```
--penguin-skin: gray;
```

To use it, use var(variablename).

Ex:- background: var(--penguin-skin);

Fallback value to CSS variable in case it fails to fetch the variable second value is used.

Ex: background: var(--penguin-skin, black);

Pseudo-class selectors

:root

:root is a pseudoclass selector that matches the root element of the document, usually html element. By creating your variables in :root, they will be available globally and can be accessed for any other selector in the style sheet.

```
:root {
  --penguin-skin: gray;
}
```

Use a media query to change a CSS variable for varying screen sizes etc:

```
@media (max-width: 350px) {
  :root {
  }
}
```

Applied Visual Design

text-align: justify;
text-align: center;
text-align: right;
text-align: left

```
img {
    height : 20pt
}
```

```
img{
    width : 220pt
    or
    width : 30%. etc
```

  = font-weight: bold;
<u> </u> = text-decoration: underline;
  = font-style: italic;
<s> </s> = text-decoration : line-through;
(for striking text)

<hr> (horizontal line)
 (line break)

rgba (r, g, b, a)
↓ ↓
0-255 0-1 (alpha value from 0-1 ; i.e; 1 fully opaque, 0 fully transparent)

Box Shadow :- (create shadows of elements)

box-shadow : offset-x offset-y blur-radius spread-radius (optional) color;

Example to create multiple box-shadows :-

box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6pt 6px rgba(0,0,0,0.23);

opacity: 0.7 (from 0-1)
1 fully opaque
0 fully transparent

Text-transform:

text-transform: uppercase (TRANSFORM ME)
lowercase (transform me)
capitalize (Transform Me)
initial (Use the default value)
inherit (text-transform value from parent element)
none (default: uses the original text)

font-size: 10px (sets the font size)
font-weight: 100 (sets how thick or thin characters
200 are in a section of text)
300
400
⋮

line-height: 25px (vertical space b/w the lines)

:hover (also a pseudo-class selector for hovering state on links)

```
:hover {
  color: red
}
```

or

```
a:hover {
  color: red
}
```

or

```
p:hover {
  color: red
}
```

etc.

Change an element's relative position

* Block-level items automatically start on new lines while inline items sit within surrounding content.
* The default layout of elements is called "normal flow" of a document. CSS offers the position property to override it.

position: relative;

this allows you to specify how CSS should move it relative to its current position in the normal flow of page. It pairs with CSS offset properties - left, right, top, bottom. These say how much the element should be moved away from where it is normally positioned.

```
p{
    position: relative      this moves the element 10px
    bottom: 10px            above i.e., 10px away from the
}                           bottom of the element where it is
                            placed.
    etc.
```

Lock an element to its parent with absolute positing

position: absolute;

left:
right:
top:
bottom

If u want.

Lock an element to the browser window with fixed positioning (used generally for navbars or side links)

```
position: fixed;
top:
bottom: if u want.
left:
right:
```

Push elements left or right using float property:-

Floating elements are removed from the normal flow of the document and pushed either to left or right of their containing parent element. It's commonly used with the width property to specify how much horizontal space the floated element takes.

float: left float: right.

Change the position of overlapping elements with z-index property :-

When elements are positioned to overlap using (position: absolute | relative | fixed | sticky), the elements might overlap. However the 'z-index' property can specify the order of how elements are stacked on top of one another. It must be a whole number

and higher values of z-index property of an element more it higher in the stack than those with lower values

z-index: 1 z-index: 2

the second one will overlap the first one or is placed above the first one if they overlap.

Center a block element horizontally:-

margin: auto;

Images are inline elements so to center them you should to

display: block;
margin: auto;

Complementary colors, tertiary colors.

hsl() to set the colors.

hue saturation lightness. → (amount of white or black
0% black
100% white
50% normal color)

0deg to 360deg.
(red) (blue)

amount of gray in the color
(100% saturated has no gray in it
minimally " " almost completely gray in it)

hsl(300, 100%, 50%) gives magenta
hsl(0, 100%, 50%) gives red. etc.

~~Background property~~

'background' property's 'linear-gradient' function:

It fades from 1 colour to other colours.

background: linear-gradient(gradient-direction, color 1, color 2, color 3, ...);

Example:-

background: linear-gradient(45deg, red, yellow, rgb(204, 204, 255));

repeating-linear-gradient to create stripes:-

background: repeating-linear-gradient(
90 deg,
yellow 0px
blue 40px,
green 40px
red 80px
);

Stop values where two colours meet.

Adding background image

background: url(' ');

Transform Property

```
p{
  transform: scale(2)
}
```

will change the size of paragraph to twice big.

```
p:hover{
    transform: scale(2-1)
}
```

when hovered the paragraph grows larger.

```
p{
  transform: skewX(-32deg)   will skew on X-axis
  transform: skewY(40deg)    will skew on Y-axis
}
p{transform: rotate(30deg) will rotate.
}
```

:: before and :: after pseudo elements

These pseudo-elements are used to add something before or after a selected element.

For ::before & ::after pseudo elements to work properly, they must have a defined 'content' property. This property is used to add things like a photo or text to the selected element. For creating shapes though, content = "". is an empty string and required.

@keyframes and animation properties:

The animation properties control how the animation should behave and the @keyframes rule controls what happens during that animation.

animation-name : sets the name of animation which is used by @keyframes later.

animation-duration : 3s sets the length of animation to 3 seconds etc.

Ex:-

```
#anim {
    animation-name : colorful;
    animation-duration : 4s;
}
@keyframes colorful {
    0% {
        background-color : blue;
    }
    100% {
        background-color : yellow;
    }
}
```

0% to 100%.

En:

```
img:hover {
    animation-name : width;
    animation-duration : 500ms
}
```

```
@keyframes width{
    100%{
        width:40px;
    }
}
```

In the above code when hovered over an image the effect lasts for just 500ms but if you want this to persist till the time cursor is hovered on it then set, animation-fill-mode: forwards;

To repeat an animation set,
animation-iteration-count: 3; repeats the animation thrice

forever animation,
animation-iteration-count: infinite;

animation-timing-function property

It controls how quickly an animated element changes over the duration of animation.
There are four keywords,
animation-timing-function: ease; (starts slow, speeds up, ends slow)
" ease-out; (starts speed, ends slow)
" ease-in; (starts slow, ends speed)
" linear; (uniform speed throughout)

Another way to set animation-timing-function is simple cubic-bezier function or bezier curve.

there are 4 points in cubic-bezier p0, p1, p2, p3

p0 is always (0,0) p3 is always (1,1)

we have to set p1, p2

(x_1, y_1) (x_2, y_2) such as.

animation-timing-function: cubic-bezier(0.25, 0.25, 0.75, 0.75)

x_1 y_1 x_2 y_2.