Day-4

Date
Page

Today Goal

- [x] CSS flexbox
- [x] CSS Grid
- [x] CSS Responsiveness

CSS Attributes Selectors

(i) css[attribute]

a[target] ⇒ [attribute = "value"]

Example: a[target = "_blank"]

(i) css [attribute = "value"] ⇒ a[target="_blank"]

(ii) css [attribute ~ = "value"] ⇒ [title ~ = "flower"]

value containing a specified word.

(iii) Css [attribute | = "value"] ⇒ [class | = "top"]

can be exactly specified value, followed by a hyphen

(iv) css [attribute ^ = "value"] ⇒ [class ^ = "top"]

Selects all value element attributes start ~~from~~ with top

(v) css [attribute $ = "value"] ⇒ [class $ = "test"]

selects elements whose attributes value ends with a specified value.

(vi) css [attribute * = "value"] ⇒ [class * = "te"]

selects elements whose ~~elements~~ attributes value contain a specified value.

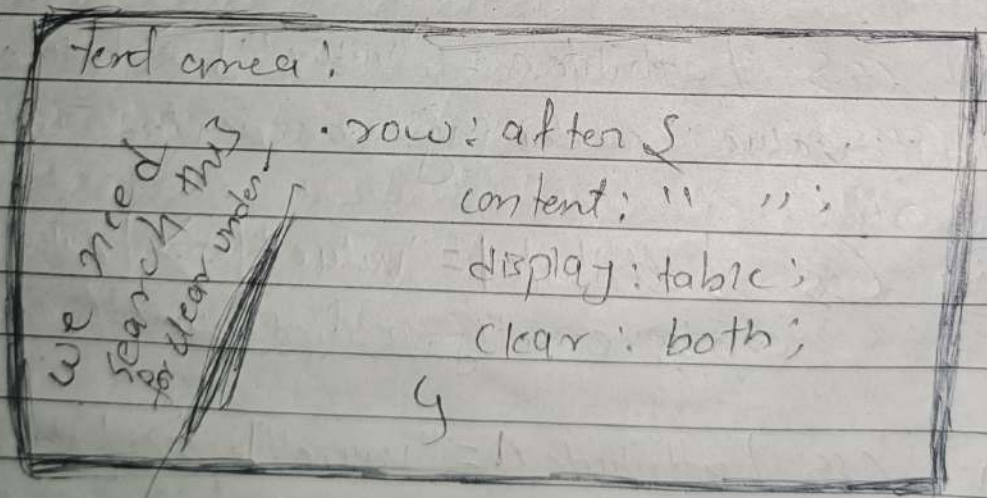
CSS Forms

we can select like this,
inputs [type = text]
inputs [type = password]
inputs [type = number]

(giving row on search/ input element)

- we can use the transition property for animation.

resize: none; resize property controls whether and how an element can be resized by the user.
- both
- horizontal
- vertical
- inherit



Css counter

Date
Page

Counter-reset: creates or resets a counter.
Counter-increment: Increaments a counter value.
Content: Inserts generated content
counter() or counters() function → Adds the value of a counter to an element.

```
body { counter-reset: section; }

h2::before {
    counter-increment: section;
    content: "Section" counter(section) ": ";
}
```

Css websites layout

Header, menus, content, footer

(i) Header
(ii) ~~Header~~ footer
(iii) Content
    ↳ 1-column ↔ often used for mobile browsers
    ↳ 2-column ↔ for tablets and laptops
    ↳ 3-column ↔ only for desktop
(iv) Footer

CSS Units

There are two types of length units
(i) absolute
(ii) relative

* Absolute Lengths

→ absolute lengths are fixed and a length expressed in any of these will appear as exactly that size.
→ not recommended to use
example :- cm, mm, in, px*, pt, pc

* Relative Lengths

em = Relative to the font-size of the element (2em means 2 times the size of the current size)
ex = Relative to the x-height of the current font
vw = Relative to 1% of the width of the viewport
vh = Relative to 1% of the height of the viewport
% = Relative to the parent element
vmax = Relative to 1% of viewport's * longer dimension
rem = Relative to the perent size

CSS specificity

Date
Page

(i) Inline style
(ii) Id selectors
(iii) classes & pseudo-classes
(iv) Attributes
(v) Element & pseudo-elements

external css or <style> tag >= high specifity <style> tag

CSS !important

→ used to add more important to a property/value than normal.

→ use of !important it breaks specifity rules.

CSS math functions

width: calc(100% - 100px)

calc() ⇒ can used in calculate the width:
max() ⇒ max(value1, value2, ...); width(30px, 30px)
min() ⇒ min(value1, value2 ...); width: min(50, 90)

CSS performance and optimization

Date
Page

- efficient css helps your websites load faster. run more smoothly.

1. use simple selectors
2. Avoid Universal Selector
3. Avoid inline styles
4. Combine and minify css files. use multiple css file.
   → use one css file when possible and remove spaces and comment to reduce file size.
   → css minifier → remove comment, extra spaces, newlines, unused styles. → it is a application.
   → postcss → you can set it up with npm and use plugin to minify.
   → online compressors : clean css, minify code, free formatter css minifier
5. Load css Efficiently
   → your main css in the head so it loads before your page is rendered.
6. Use shorthand Properties
7. Cache your css
   Make sure your css file is cached by the browser by giving it a long expiration time in your server settings.

- keep selectors short and simple
- Avoid layout-thrashing operation
- use external, minified and cached stylesheets
- Use efficient hiding and animation tech.

CSS Accessibility

Help all user interact with your website, including those with visual or mobility challenges.

(1) Use Visible Focus Indicators
: focus

(2) provide high Contrast

(3) Avoid styling thats hides focus

(4) Use semantic HTML + CSS
For exam div, nav for selecting every thing

(5) Respect user preferences
(prefers-reduced-motion)

Summary:
- Keep focus outlines visible
- maintain good color contrast
- use semantic HTML element
- Respect motion preferences

CSS Border Images

Date Page

border-image : url("border.png") 30 round;

<source> <slice> <repeat>

(30 → image devided into 30 part)

stretch
repeat
round
space

CSS Backgrounds

background-size:
background-origin:
background-clip:

CSS support multiple background image.

CSS color keyword

background-color: transparent; → transparent background.
background-color: currentcolor; → hold the current value of the color property
background-color: Inherit;

CSS Gradient

Linear Gradient (down/up/left/right/diagonally)
Radial Gradient (define by their center)
Conic Gradient (rotated around a center point)

* Linear Gradient

```
background-color: red;
background-image: linear-gradient(red, yellow);
background-image: linear-gradient(to right, red, yellow);
background-image: linear-gradient(to bottom right, red, yellow);
background-image: linear-gradient(0deg, red, yellow);
```

0deg → L-R
180deg → U-B
-90deg → R-L

```
background-image: repeat-linear-gradient(red, yellow, blue);
```

* Radial Gradient

```
background-color: red;
background-image: radial-gradient(red, yellow, green);
background-image: radial-gradient(circle, red, yellow, green);
```

use of different size
closed-side, farthest-side, closest-corner, farthest-corner

```
background-image: repeat-radial-gradient(red, yellow, green);
```

* Conic Gradients

background-color : red;
background-image : conic-gradient (red, yellow, green);
background-image : conic-gradient (red 45deg, yellow 90deg, green 120deg);
background-image : repeating-conic-gradient (red, yellow, green);

CSS shadows

Text shadow

~~text-shadow: 2px 2px;~~ horizontal, vertical
text-shadow : 2px 2px red; horizontal vertical, color
text-shadow : 2px 2px 5px red; x, y, blur, color
text-shadow : 0 0 3px #FF0000

Box shadow

box-shadow : 10px 10px; x, y
box-shadow : 10px 10px lightblue; x, y, color
box-shadow : 10px 10px 5px lightblue; x, y, blur, color
box-shadow : 10px, 10px, 5px, 2px, blue; x, y, blur, spread radius, color
box-shadow : 10px, 10px, 5px, 12px, blue, inset;

→ Inset parameter changes the shadow from outer shadow to an inner shadow.

→ we can use the multiple shadow on text & Box Element.

CSS Flexbox
(Flexible box Layout)

- Arranging items row & columns
- It gives flexible responsive layout structure, without using float or positioning.

- one dimensional Layout
  row OR column

✱ Flex Container properties

Flex-direction:
flex-wrap:
flex-flow:
justify-content:
align-items;
align-content:

✱ flex-direction property
row, column, row-reverse, column-reverse

✱ flex-wrap property
nowrap, wrap, wrap-reverse

✱ flex-flow property
short hand property for direction & wrap
flex-flow: row wrap;

* justify-content property (horizontal alignment)
→ center, flex-start, flex-end, space-around, space-between
space-evenly.

* CSS align-items property (vertically align)
→ center, flex-start, flex-end
stretch, baseline, normal

* CSS align-content property
→ align-content is similar to align-items, it aligns the flex-lines.
property:
① center, stretch, flex-start, flex-end, space-around,
space-between, space-evenly

Flex items propeties

order
flex-grow
flex-shrink
flex-basis
flex.
align-self

* order property
→ speaies the order of flex inside the flex container.
style: "order: 3"
style: "order: 1"

* flex-grow property
→ flex-grow property speaies how much a flex item will grow relative to the rest of the flex items
 style(flex-grow 8); >
<div style(flex-grow: 2); >

* flex-shrink
→ How much a flex item will shrink relative to the rest of the flex items.
<div style: "flex-shrink: 0"> 3

* flex-basis
→ initial length of a flex-item.
<div style: "flex-basis: 900px"> 3

* flex-basis Property

* flex
→ flex property is a shorthand property for the flex-grow, flex-shrink and flex-basis properties.

* align-self
→ specifies alignment for the selected item inside the flexible container.
→ The align-self property override the default alignment set by the containers align-items property.

CSS Grid

Date
Page

→ Grid based layout system with rows and columns.
→ Responsive layout structure without using float or positioning.

Display : grid, inline-grid ;

Grid columns, Rows and Gaps

column-gap : 50px
row-gap : 50px
gap : 50px 50px ; → short hand :

Grid Lines

- grid-column-start : 2 ;
- grid-column-end : 3 ;
- grid-row-start : 2 ;
- grid-row-end : 3 ;
- grid-column : 2 / span 2 ;
- grid-row : 2 / span 2 ;

Grid container

property:
(i) grid-template-rows: 20px 20px auto auto;
(ii) grid-template-columns: 1fr 1fr 1fr;
(iii) grid:
(iv) grid-template:

Cell sizing with fr unit

→ The fr unit stands for "fraction" automatically devides the available space into fractions.
→ 1 fr will take 1 fraction of the available space.

Justify-content property

* space-evenly
* space-around
* space-between
* center, start, end

Align-content property

* space-evenly
* space-around
* space-between
* center, start, end

Place-content property

→ place-content property is a shorthand property for the align-content and the justify-content properties.

→ { start, end, center }

CSS Grid item

```
grid-colum-start : 1 ;
grid-column-end : 3 ;
grid-column : 1/span 2;

grid-row-start : 2 ;
grid-row-end : 3 ;
grid-row : 2/span ;
```

The grid area property

→ The grid area property is a short hand property for the grid-row-start, grid-column-start, grid-row-end and grid-column-end.

This syntax
row-start / colum-start / row-end / column-end

```
grid-area : 1 / 2 / 3 / 2 ;
```

~~grid-template-areas~~

naming Grid items with grid-area

grid-area → can be use asign the name to items.

grid-template-areas → refer the item name.

→ (area area area)

→ (area area area ... ;)

* The justify-self property
→ Justify-self is used to align the content of a grid item along with row axis.

right, left, center

* The align-self property
→ align the content of a grid item along the column axis.

CSS support rules.

Date
Page

→ @supports check browser supports a specific css property or value and define fallback.

Basic syntax:

```
@supports (property : value) {
    /* css rules */
}
```