CSS3

CSS3 transitions and transformations

Transitions and transformations are new set of css properties through which we could able to control the transformation of the dom element while rendering.

adding the animation delay through transition

Css Transform property

Any dom element can be transformed in following three diff ways

① rotate ② translate ③ scale

① Transform with "Rotate" property

Through which any dom element can be rotated within 0 to 365°

Eg: transform: rotate (90deg);

3D transform:

transform: rotatex (90deg);
transform: rotatey (90deg);

② Transform with "translate" property

Through which we could able to move dom element to new position with provided values

Eg: transform: translate (30px, 20px);

③ Transform with "scale" property

through which we could able to re-draw the element with zoom effects

Eg: transform : scale(3,4); //predraws the dom with 3 times of width and 4 times of height.

13/4/20 CSS3 Keyframes

The '@' Keyframe (or) the set of CSS rules which lets the user to control the intermediate steps in css animation sequence by establishing Keyframes along with the animation sequence which must be reached to a certain points during the animation.

* In order to make use of Keyframes we create a Keyframe rule with user defined name, we use css animation property to match an animation to its Keyframe list.

* each Keyframe rule contains a style list of Keyframe selectors each of which contains a percentage along with the animation at which the Keyframe occurs as well as a block containing transformation for that Keyframe.

Ex: creating Keyframe

Syntax:

```
@ Keyframes < Keyframe name > {
        Value %. {
            ::: // set of css properties
        }
        Value 2%. {
            ::: // set of css properties
            :::
        }
        Value 3%.
            ::: // set of css properties
    }
```

Keyframe example

```
@keyframes mystyles {
    10% {
        font-size : 10px;
        background : green;
        ...
    }
    13% {
        ....
    } ... (cont...)

    26% {
        ....
        ....
    }
    90% {
        ....
        ....
    }
```

14/4/20

CSS - 'linear-gradient':

The linear-gradient() is a CSS function which creates an image consisting of a progressive transition between two or more colors along a straight lines.

using CSS3 linear gradients we could add smooth transitions between two or more specified colors.

CSS3 supports two types of linear gradients

1. linear gradients:
   These gradients goes in direction left/right/up/down or diagonally

2. radial gradients:
   These gradients goes to center of container.

In order to create a linear gradient for sure we need to add minimum of two color stops which render smooth transition among. while adding transition we can even specify starting point or a direction or an angle.

Note:
linear gradient has to be added to CSS 'background-image' property.

Eg: background-image: linear-gradient (red, green)

with direction:

linear-gradient (to right bottom, blue, pink, yellow);

direction could be.

to left
to right
to left bottom
to right top
to right bottom
etc.

with direction as angle:

linear-gradient (60deg, red, blue);

Repeat linear gradients:

background-image: repeating-linear-gradient (red, yellow. 5%, blue 4%, pink 4%);

Radial gradients:

using which the gradients can be specified with a center.

By default the radial gradient shape is ellipse and position is center, we can still change the position and shape.

radial-gradient (red, green, blue);

radial-gradient (red 5%, green 10%, blue 6%);

radial-gradient (circle, red, green, blue);
// with shape.

15/4/20

FLEX

following are 5 layout models beens supported in css

1. Block layout
2. Inline layout
3. Table layout
4. positioned layout
5. flex box layout

flex box layout: flex container becomes flexible by setting the display property to flex

```
display : flex;
```

Through flex box layout models we could design responsive layout structures with out any dependency of either float or positions

following are the properties can be applied to flex container.

1. flex-direction
2. flex-wrap
3. flex-flow
4. justify-content
5. align-items
6. align-content

1. flex-direction property: Through which we can specify in which direction the flex items to be rendered.

* following are the posible values it takes

1. Column
2. row
3. column-reverse.
4. row-reverse.

2. flex-wrap: Through which the items with flex container gets wrapped if needed

flex-wrap: wrap/nowrap/wrap-reverse;

3. flex-flow: Its a shortcut way to define both direction and wrap property.

flex-flow: row wrap;

4. justify-content: Through which we can align the items under flex container.

justify-content: center/flex-start/flex-end/ stretch/baseline.

5. align-items: Through which we could align flex items

align-items: center/flex-start/flex-end/stretch/ baseline.

6. align-content: to align content inside flex container

flex-wrap: wrap;

align content: space-between / space-around / stretch / center;

flex-item properties

following are the properties can be applied to flex items

1. order
2. flex-grow
3. flex-shrink
4. flex-basis
5. flex
6. align-self

1. order: through which we can control rendering order of flex item

```
order: 1/3/5...;
```

2. flex-grow: through which we can specify how much an item grows relative to other flex items in it.

```
flex-grow: 1/3/8.. etc
```

3. flex-shrink: specifies how much item shrinks relative to other items

```
flex-shrink: 0; // no shrink
```

4. flex-basis: through which we can specify initial width to flex item.

```
flex-basis: 120px;
```

5. flex: Through which we can control flex-grow, shrink and basis at a time.

```
flex: 1 0 200px;
```

6. align-self: Through which we can make item to get aligned automatically within the container

```
align-self: center;
```

17/4/20

CSS3 Media Queries

a pre-defined feature being supported in CSS3 through which we could able to design multiple set of css classes for different device dimensions so that corresponding css will be applied for particular device dimensions

"@media" is a keyword through which we define set of css classes for particular dimensions.

Syntax: @media <media type> and (dimension) {

```
....
}
```

Eg: @media screen and (min-width: 400px) {

```
.abc{
...
}
```

```
#pqr {
    font-size : 10px;
    ...
}
div {
    :::
}
}
```

following are set of properties through which we can specify device dimensions within media queries.

- min-width
- min-height
- max-height
- max-width

following are types of media can be specified while defining media queries

Media types:

Screen - Any color screen
all - Indicates all screens
handheld - indicates all small screens
projection
TV
print - indicates print device.
etc

→ while defining the media queries we can specify the device orientation type using orientation property

"orientation" - possible values landscape/ potrait

Syntax

@media screen and (max-height : 300px) and (min-width : 300px) and (orientation : landscape) {

..... // set of css classes

}

→ the media queries can be injected into the website by @media queries or

Eg:

<link rel = "stylesheet" media = "max-width : 200px; min-height : 400px; orientation : potrait" href = "/data/app.css">