CSS

Cheat sheet

- 3 ways to apply CSS
- CSS selectors
- CSS color systems
- CSS Box model
- CSS Display property
- CSS position property
- CSS float and clear
- CSS Text properties
- Google fonts
- CSS units
- CSS Flexbox
- CSS Grid
- CSS Media queries and responsive design
- CSS transforms
- CSS transitions
- CSS animations
- CSS Variables

The 3 ways to apply css

Inline (Specify CSS as style attribute in element tag) style="property : value;”	para 2 para 6

Internal
(CSS is specified inside <style> tag)
 	
selector
{
property : value ;
}
 	
selector - who you want to modify ?
property - what you want to modify ?
value - how you want to modify ?
	<style>
	body{
	background-color: #EAF6F6;
	}
	</style>

external (CSS is specified in an independent .css file and linked inside the html file)	<link rel="stylesheet" href="styles/ref.css">

Different CSS selectors

Universal selector
(Select's everything)
	* {
	color : black;
	}

Element selector
(Select's all same elements)
	Img {
	width: 100px;
	height: 200px;
	}

Class selector
(Select elements with class of complete)
	.complete{
	color: green;
	}

Id selector
(Select the element with id of 'logout')
	#logout {
	color: orange;
	height: 200px;
	}

Selector list
(selects elements)
	span, div {
	border: red 2px solid;
	}
	
	#logo, img, .log{
	}

Different CSS selectors

Descendent selector
(Select all <a>'s that are nested listed inside an   )
	li a {
	color : teal;
	}

Adjacent selector
(Select only the paragraphs that are immediately preceded by an <h1>)
	h1 + p {
	color: red;
	}

Direct child
(Select only the   's that are direct children of a element
	div > li{
	color: white;
	}

Attribute selector
(Select all input elements where the type attribute is set to "text")
	Input[type = "text"] {
	color: yellow;
	width: 200px;
	}

Different CSS selectors

Pseudo selector (Specifies a special state of the selected element)	:active :checked :first :first-child :hover :not() :nth-child() :nth-of-type()
Pseudo elements (Keyword added to a selector that lets you style a particular part of Selected element)	::after ::before ::first-letter ::first-line ::selection

Difference between class and Id

Class	Id
1. Groups related elements 2. More than 1 element can have same class value 3. A single element can have more than 1 class selectors	1. For Unique elements only 2. Only 1 element can have same id value. 3. A single element can’t have more than 1 id selectors

CSS color systems

Named colors (There are 147 built-in colors)	Salmon Springgreen Teal
RGB colors rgb(RED, GREEN, BLUE) (Ranges from 0-255)	color: rgb(255, 0, 0) //red color: rgb(95, 100, 45)
Hexadecimal (Ranges from **00-FF)** #42cbf5 42 – red cb – green f5 – blue 3 digit notation gives only 4096 colors 6 digit notation gives more than 16m	color: #FFFFFF //white color: #000000 //black color: #00FF00 // green color: #000 //black color: #FFF //white color: #FF0 //yellow

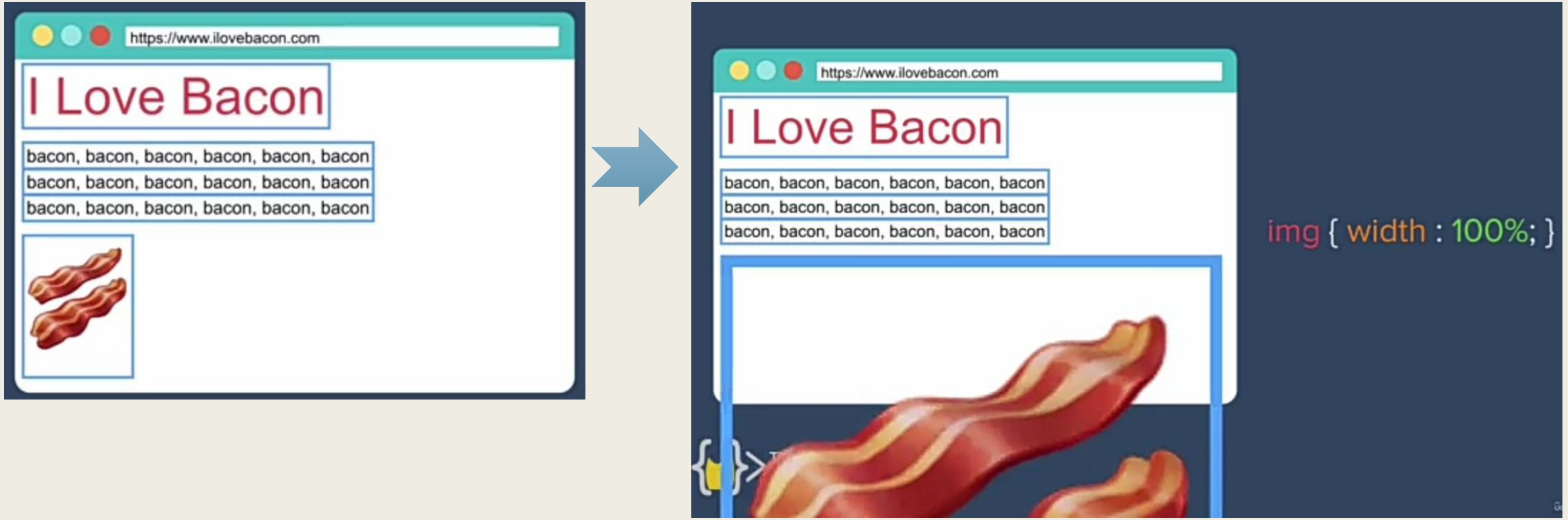
CSS color systems

HSL
hsl(hue, saturation, lightness)
hue – 0-360 (pos. on color wheel)
0 – red
120 – green
240 – blue
saturation – 0%-100%
Lightness - 0%-100%
	color: hsl(45, 80%, 5%)
	color: hsl(0, 50%, 0%) //black
	color: hsl(0, 50%, 100%) //white
	color: hsl(0, 100%, 50%) //red
	color: hsl(200, 80%, 80%) //light blue

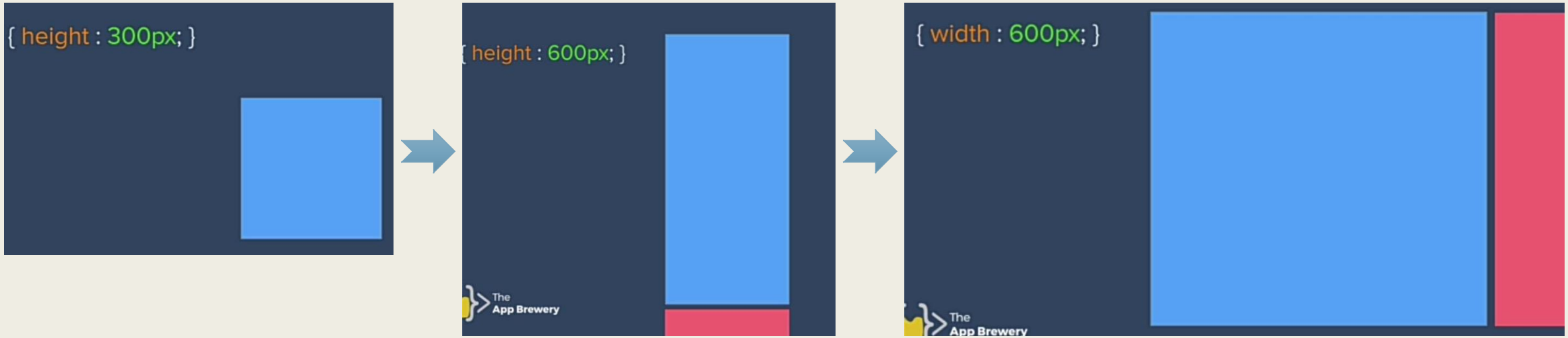
RGBA , HSLA
A – represents alpha
(how transparent)
It ranges from 0-1
0 – full transparent
1 – not transparent
 	
For hexadecimal add 2 more digits
 	
We can also set opacity property →
But it will set opacity for other elements like borders, text within etc.
	color: hsla(180, 50%, 50%, 0.5)
	// transparent cyan
	
	color: rgb(220, 180, 0, 0.6)
	//transparent yellow
	
	color: #99aef709
	// transparent purple
	
	color: #99aef7;
	opacity: 0.5;

The Box Model

Every HTML element can be treated as a box and we can style these boxes to affect the layout of the page

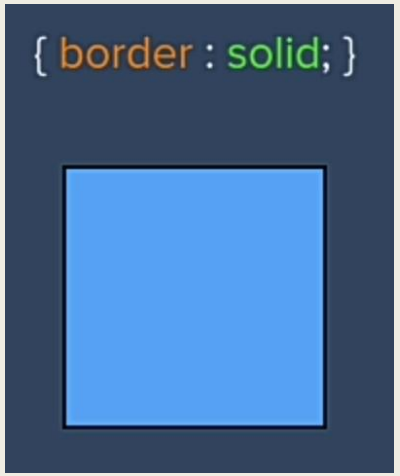


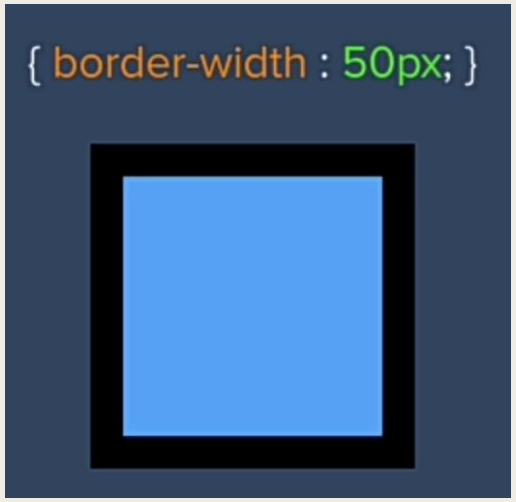
Consider, a div of height and width 300px

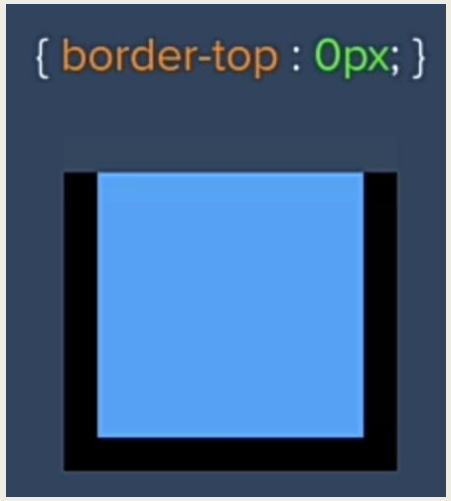


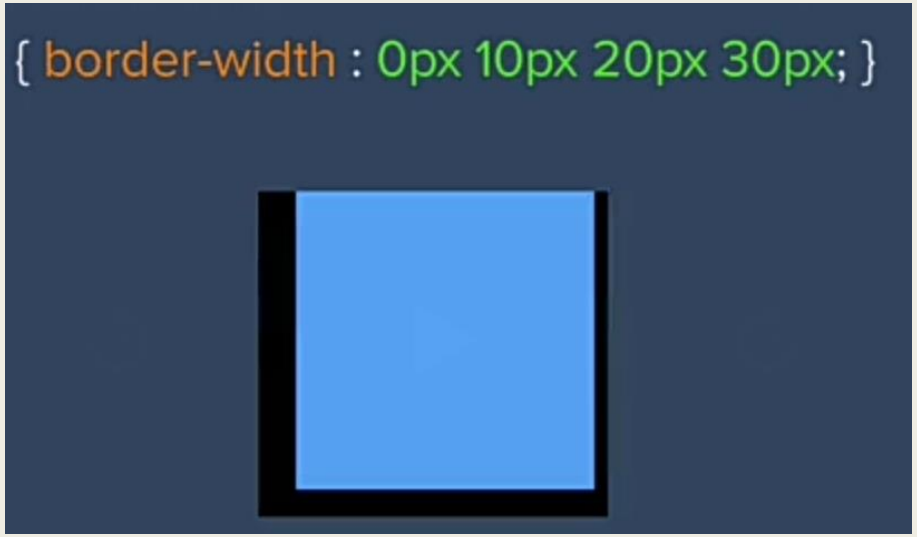
We can change its height and width to 600px

In addition to height and width, we can also specify the border

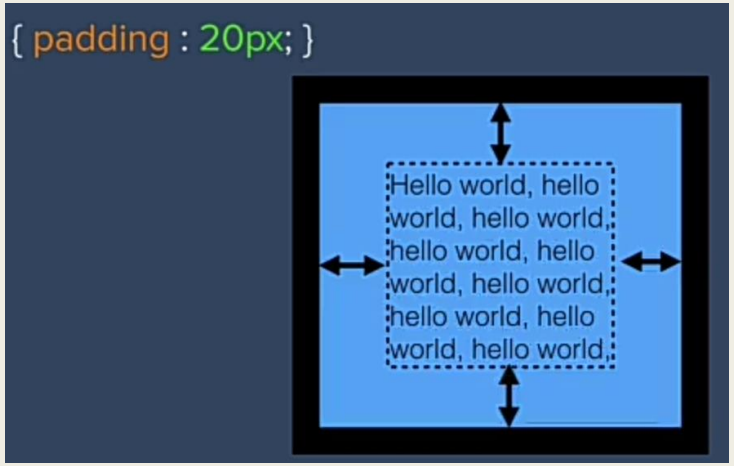


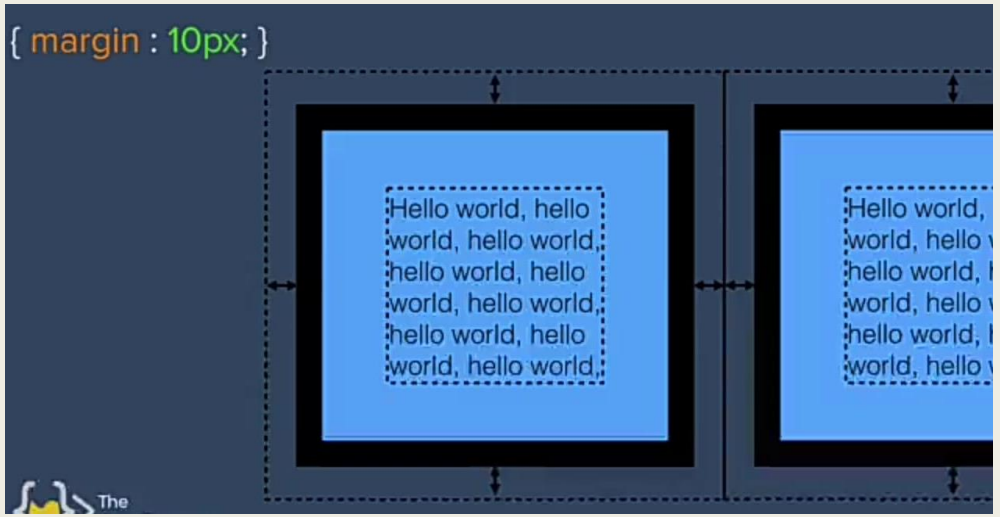






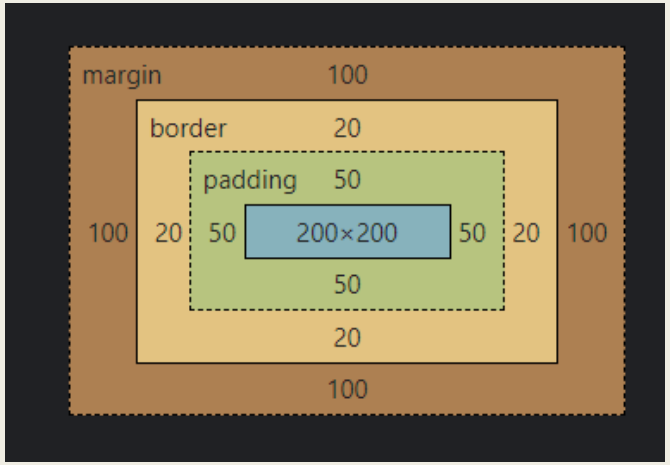
Suppose we have some content in our div, we can also give some margin and padding





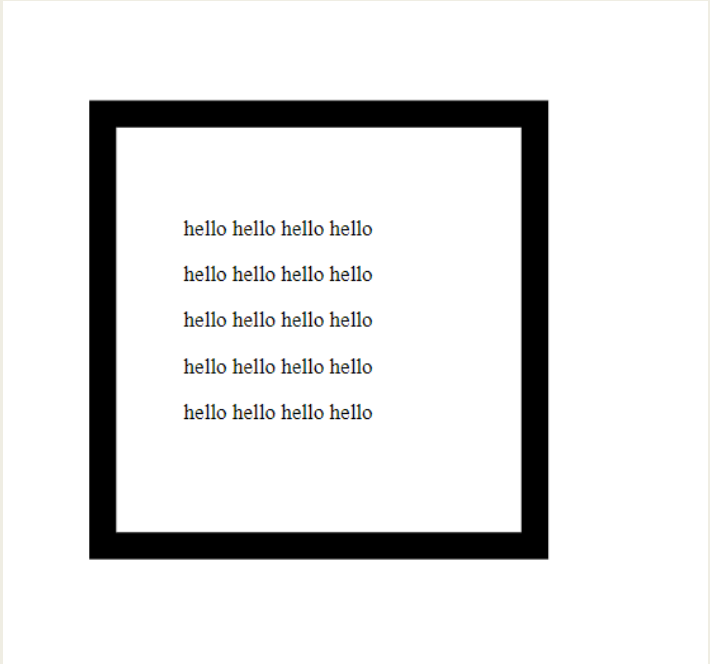
CSS →

```
div{
        height: 200px;
        width:200px;
        border:solid;
        border-width: 20px;
        padding:50px;
        margin: 100px;
    }
```

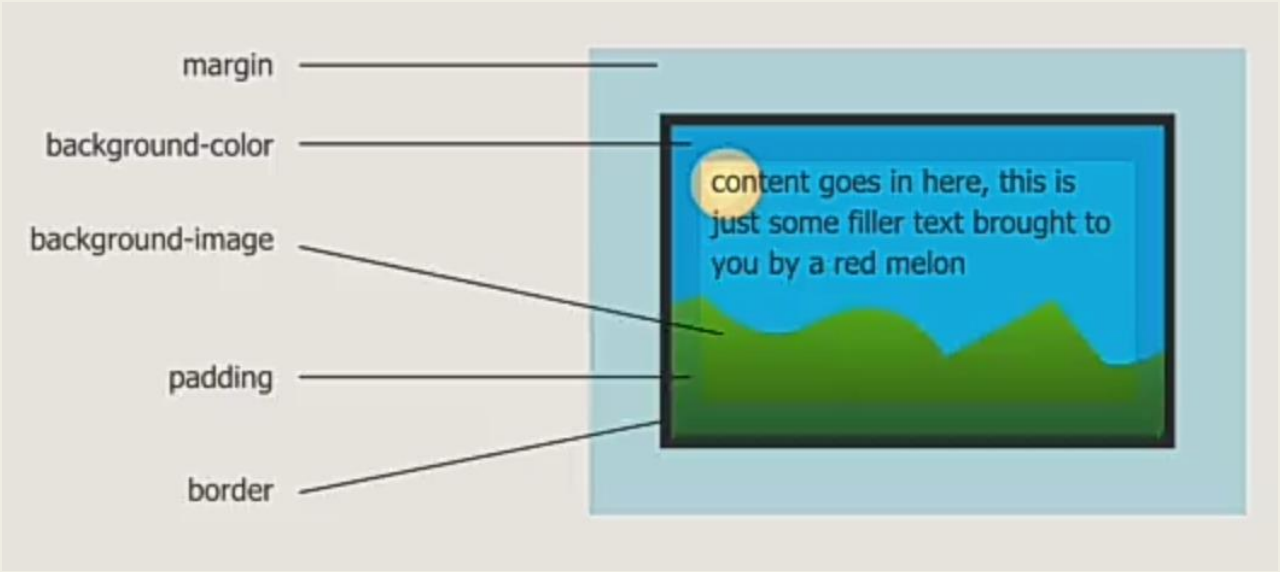


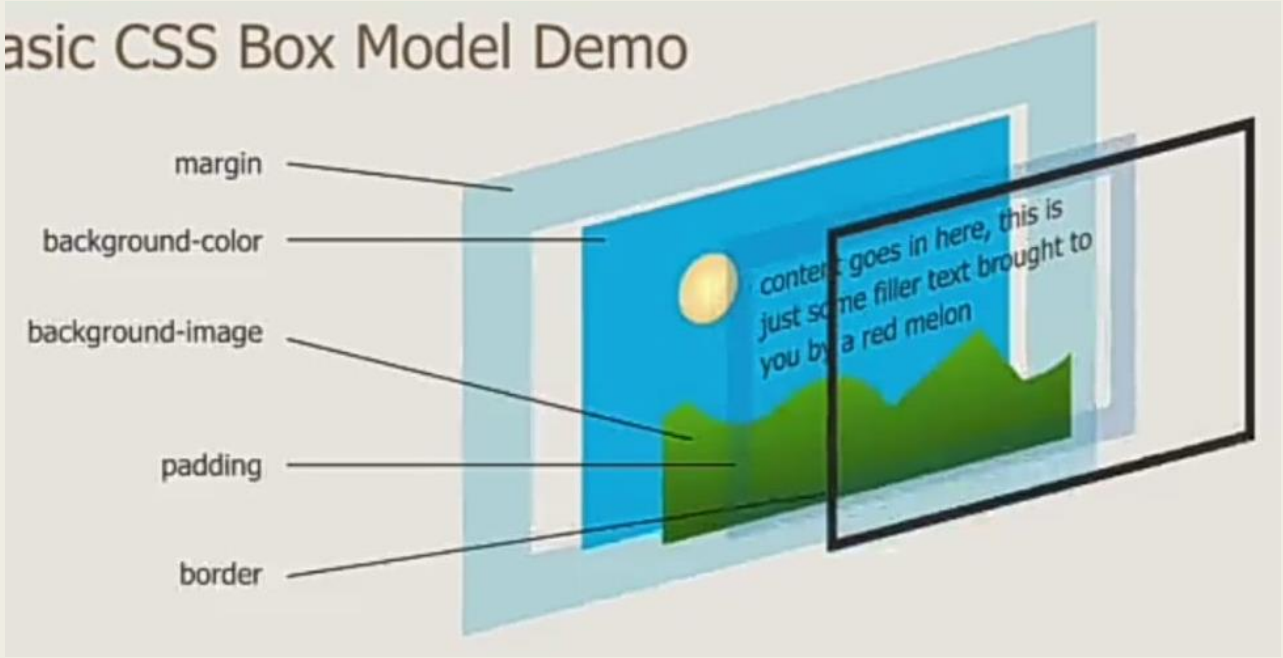
← Box model

OUTPUT →



CSS Box model Demo





The Box Model

Every element in CSS is a box !!!

Content box – height, width.	height: 600px width: 600px
Border	border: 2px solid purple border: 2px dashed purple border-width: 50px border-top: 0px border-width: 0px 10px 20px 30px border-top-color: cyan border-top-style: dotted
Padding (Spacing between content box and border)	padding: 20px padding: 10px 20px 30px 40px padding-bottom: 40px
Margin (Spacing between border and other elements)	margin: 10px margin: 10px 20px 30px 40px margin-bottom: 30px margin: auto → Horizontally centres element within it's container

The Box Model

Other box model properties

Box-sizing Width and height only apply to content → Width and height apply to content, padding & borders →	 box-sizing: border-box box-sizing: content-box // default

Background-clip
Background extends till content box →
Background extends till padding box →
Background extends till border box →
Background extends till text (exp)* →
	
	background-clip: content-box;
	background-clip: padding-box;
	background-clip: border-box; //default
	background-clip: text

Display property

Inline • They don’t change height and width • They don’t push other elements to next line • Elements include . span, img and a	display: inline;
block • They change height and width • They occupy full line and push other elements to next line • Elements include p, h1-h6, div, ol, ul, li and form.	display: block;
Inline-block • Hybrid of inline and block element • They don’t push other elements to next line and change height and width	display: inline-block;
none • it gets rid of element	display: none;

Position property

top , bottom , left , right co-ordinates • take that co ordinate and give it a push(margin)	top 20px; give the element a 20px push from top
z-index • Reordering and positioning elements in front or behind	z-index: 1;
static (default) • here, left, right, top, bottom co-ordinates will not work	position: static;
relative • here, left, right, top, bottom co-ordinates will work	position: relative left: 300px; right: 50px; top: 150px; bottom: 300px;

Position property

absolute
• Here, the element is taken out of the natural flow.
• It, positions element relative to its parent
• if no parent is given then the body acts as parent
	#abs{
	position: relative;
	}
	
	# abs absolute{
	position: absolute;
	left: 20px;
	right: 10px;
	top: 80px;
	bottom: 30px;
	}

sticky • Element will scroll with it's container until it is at the top of the container and will remain at same position when page is scrolled	position: sticky top 20px;

Position property

fixed • Fix position of element relative to the viewport	position:fixed left: 0; right: 0; top: 0; bottom: 0; margin: auto

CSS float and clear

Float • Pushes the element to left or right and the following siblings will wrap around the floating element	float: right;
clear • It prevents following siblings to wrap around the floating element	clear :right;

Text Properties

Font-size (Changes font-size)	font-size: 60px
Text-align (change horizontal alignment of block of text) Works only for block elements	text-align: center text-align: right text-align: left text-align: justify
Font-weight (Control's weight of boldness)	font-weight: 400 //normal font-weight: 700 //bold
Font-style (Specifies font should be italic/ oblique/normal)	font-style: normal font-style: italic font-style: oblique
Font-family (specifies font)	font-family: Verdana //font stack font-family: Verdana, Arial, sans-serif

Text Properties

Line-height (controls height of each individual line of text)	line-height: 20px
Word-spacing (controls space between words)	word-spacing: 50px;
Letter-spacing (controls space between letters)	letter-spacing: 20px;
Text-decoration (controls underlines/ decorative lines on text) → Mostly used to get rid of blue underline on link's	text-decoration: underline wavy pink text-decoration: none text-decoration: overline text-decoration: line-through Text-decoration: dashed
Text-transform (Controls case of text)	text-transform: capitalize text-transform: uppercase text-transform: lowercase

White-space	// content remains on single line
	white-space: nowrap;
	
	// content is wraped
	white-space: normal;

Text Properties

Text-shadow (Adds shadows to text) → specify offset-x, offset-y, blur(optional) and a color	text-shadow: 10px 10px magenta text-shadow: -5px 5px 5px magenta; text-shadow: 2px 2px 5px grey // Multiple text shadows text-shadow: -5px 5px 5px magenta, -10px 10px cyan;
Text-indent (Control indentation of text) → For paragraphs it only indents first line and not whole paragraph.	text-indent: 20px;
Text-overflow (Specifies how to clip of the end of a line) → Will only work if we specify overflow and white-space properties	overflow: hidden white-space: nowrap; text-overflow: clip; text-overflow: ellipsis; Text-overflow: inherit;

Google fonts

1. Go to https://fonts.google.com/
2. Select your font style's
3. Go to view your selected fonts > use on the web
4. Copy the code for <link> and paste in on <head> tag of your html file.
5. Copy the code for CSS rules to specify families and paste in css section for the desired text.

If you don't want to use the full fonts but only need to use for particular character set
then add $text=character_set at the end of the url

E.g. I want to use a particular font family only for "hello"
$text=HELLO

The url gives a CSS file and consist a bunch of font face declarations

The url in src: if you request it you have actually downloaded that font which is a woff2 file
(web open Font Format)

Devtools > sources → if you use that font, you will see a bunch of requests to use that font

CSS UNITS

Relative

% (percentages)
- Always relative to some other value.
- Value – sometimes is parent or element
	// half the width of parent
	width: 50%
	
	// half the font-size of element
	line-height: 50%

em - 1em = font-size of parent or element	
rem - relative to root html element	If root font-size is 20px 1 rem is always 20px
vw (view width) - 1 vw is 1% of width of viewport	100% vw → make an element take full width on screen
vh (view height) - 1 vh is 1% of height of viewport	100% vh → make an element take full height on screen

CSS UNITS

Absolute *– fixed in size in relation to one another*

px (pixel)	- Commonly used - Not recommend for responsive websites
Other absolute units :- **cm, in, mm, Q, pc, pt →**	Not commonly used

16px = 100% = 1em

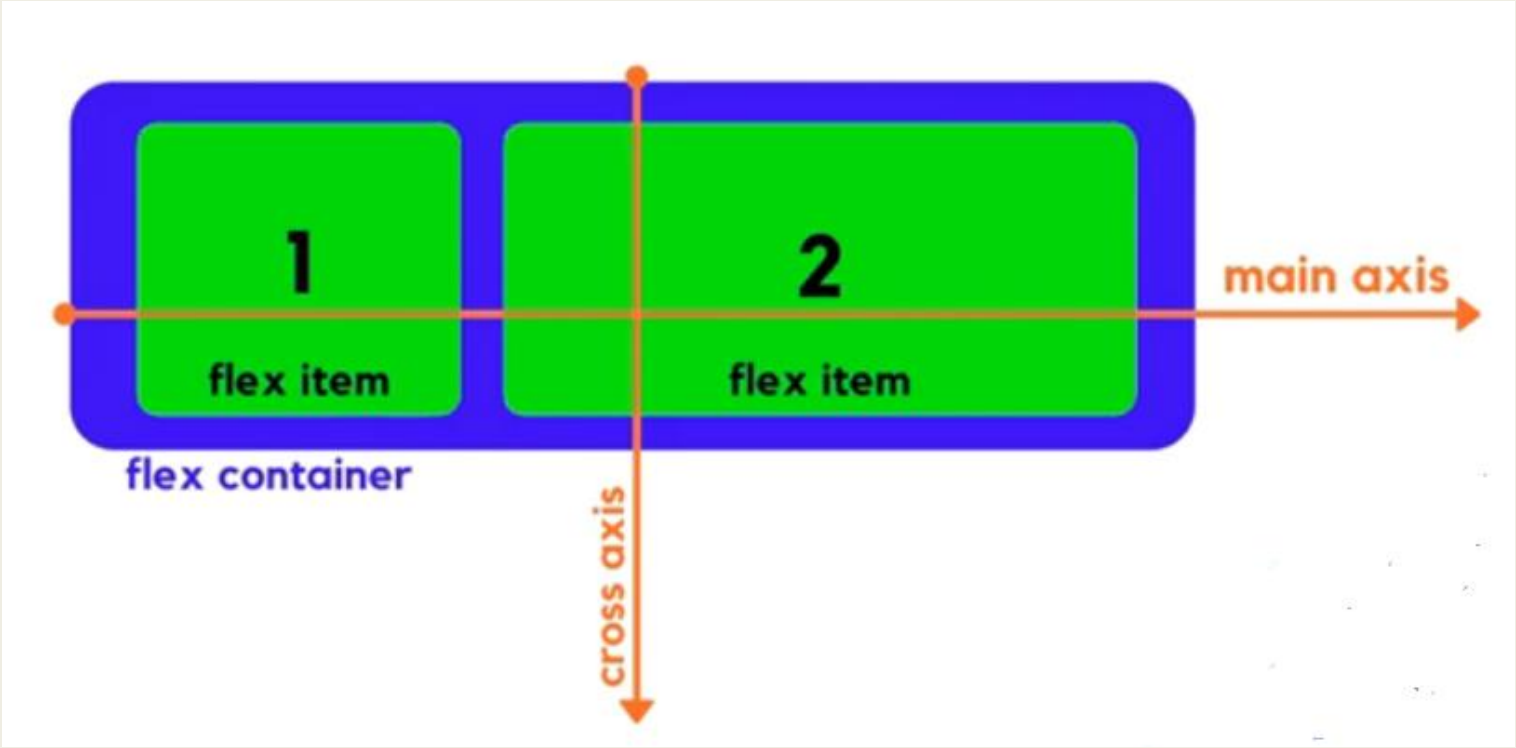
What Should I Use ?	
px	• Avoid for font-sizes • Use for small details like borders & shadows
rem/em	• Go to choice for font-sizes and padding/margin
%	• Useful for defining layouts and widths
vh/vw	• Use for larger layout concerns

Other CSS properties

opacity	
Box-Shadow	text-indent: 20px;
Text-overflow (Specifies how to clip of the end of a line) → Will only work if we specify overflow and white-space properties	overflow: hidden white-space: nowrap; text-overflow: clip; text-overflow: ellipsis; Text-overflow: inherit;

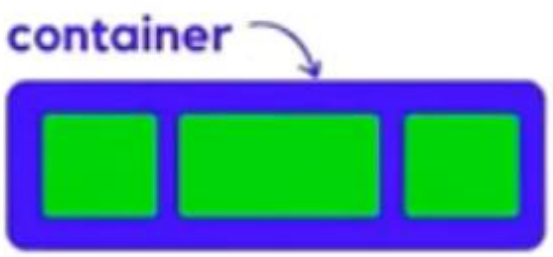
CSS Flexbox

1-dimensional system that can layout, align and distribute space among items in a container.

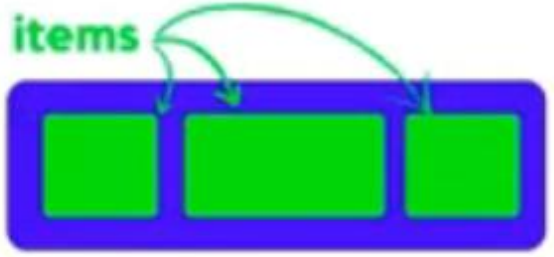


main axis – row axis
cross axis - column axis

CSS Flexbox Properties



- display
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content



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

CSS Flexbox

To layout, align and distribute space among items in a container.

Container properties

First set Display of container to flex →	display: flex

Flex-direction
(Species how items are placed in flex container, defining the main axis and it's direction)
	flex-direction: row; //default
	flex-direction: row-reverse;
	flex-direction: column;
	flex-direction: column-reverse;

flex-wrap
(Specifies whether items are forced into a single line or can be wrapped into multiple lines)
	flex-wrap: nowrap ; //default
	flex-wrap: wrap;
	flex-wrap: wrap-reverse;

Flex-flow (Shorthand for flex-direction and flex-wrap)	flex-flow: row wrap;

CSS Flexbox

To layout, align and distribute space among items in a container.

Container properties

justify-content
(Defines how space is distributed between items in flex container along row axis)
	justify-content: flex-start; //default
	justify-content: flex-end;
	justify-content: center;
	justify-content: space-between;
	justify-content: space-evenly;
	justify-content: space-around;

align-items (Specifies how space is distributed between items in flex container. Along column axis)	align-items: flex-end; align-items: flex-start; align-items: center; align-items: space-between; align-items: space-around; align-items: space-evenly; align-items: stretch; align-items: baseline;

CSS Flexbox

To layout, align and distribute space among items in a container.

Container properties

align-content (Specifies how space is distributed between rows in flex container along the column axis)	align-content: flex-start; align-content: center; align-content: space-between; align-content: space-around; align-content: space-evenly; align-content: stretch; align-content: baseline;

CSS Flexbox

To layout, align and distribute space to individual items separately in a container.

Flex item properties

align-self (allows you to override align-items on individual flex items)	align-self: flex-start; align-self: flex-end; align-self: stretch;
flex (shorthand for flex: ‘flex-grow’ ‘flex-shrink’ ‘flex-basis' ;)	flex: 2 2 10%
flex-basis (specifies the ideal size of a flex item before it's placed into a flex container.)	flex-basis: 25%;

flex-grow
(Dictates how the unused space should
be spread amongst flex items)
	flex-grow: 0; //default
	flex-grow: 1;
	/* All boxes share space evenly */

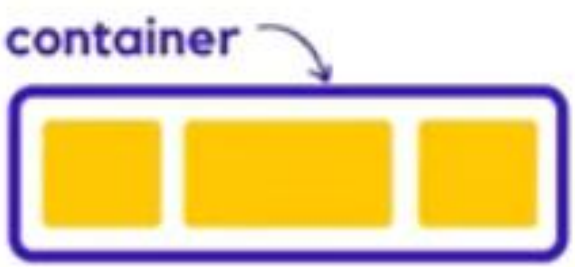
flex-shrink (Dictates how items should shrink when there isn't enough space in container)	flex-shrink: 1;
Order (Specifies the order used to layout items in their flex container)	order : 2; order: -5;

CSS Grid

2-dimensional layout system that can layout elements in rows and columns at the same time.

grid lines --> columns and rows are made using grid lines
grid track --> full columns or rows in a group container
grid area --> total space surrounded by grid lines
grid cell --> single space within our grid

CSS Grid Properties



- grid-template-columns
- grid-template-rows
- grid-template-areas
- grid-template
- grid-column-gap
- grid-row-gap
- grid-gap
- justify-items
- align-items
- place-items
- justify-content
- align-content
- place-content
- grid-auto-columns
- grid-auto-rows
- grid-auto-flow
- grid

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row
- grid-area
- justify-self
- align-self
- place-self

CSS Grid

Container properties

First set Display of container to grid →	display: grid
grid-template-columns (Species the size and no. of columns)	
• create 3 columns of 200px size →	grid-template-columns: 200px 200px 200px ;
• auto fills undefined space →	grid-template-columns: 50% 25% 25% auto ;
• create 3 equal width columns →	grid-template-columns: 1fr 1fr 1fr ; or grid-template-columns: repeat(3, 1fr) ;
• auto-fill will fill our container with col. with width that we have specified →	grid-template-columns: repeat(auto-fill, 300px) ;
• auto-fit will fill container with col. only if we have specified them in our html →	grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)) ;
• minmax species min. value and max value of col. →	grid-template-columns: 1fr minmax(300px, 1fr) ; grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
• We can also name the grid lines	grid-template-columns: [col-start] 1fr [col-2] 1fr [col-end]

CSS Grid

Container properties

grid-template-rows (Species the size and no. of rows) create 2 rows, 1st 100px and 2nd 50px →	grid-template-rows: 100px 50px;
grid-auto-columns **grid-auto-rows** (control row and columns of implicitly defined grid tracks)	grid-auto-rows: 50px;
grid-auto-flow (Specifies how things are placed inside grid container)	// items are placed in container according to the column grid-auto-flow: column grid-auto-flow: dense;
grid-row-gap **grid-column-gap** (specifies space between rows/columns)	grid-row-gap: 10px; grid-column-gap: 10px;
grid-gap (shorthand for grid-row-gap and grid-column-gap)	grid-gap: 20px 30px; grid-gap: 10px

CSS Grid

Container properties

align-items
(positions the items vertically within the grid cell)
	align-items: stretch; //default
	align-items: start;
	align-items: end;
	align-items: center;

justify-items
(positions the items horizontally within the grid cell)
	justify-items: stretch; //default
	justify-items: start;
	justify-items: end;
	justify-items: center;

place-items (short-hand for align-items and justify items)	place-items: center stretch; place-items: center;

CSS Grid

Container properties

align-content
(specifies how we align our content vertically within our container)
	align-content: start; //default
	align-content: end;
	align-content: center;
	align-content: space-between;
	align-content: space-around;
	align-content: space-evenly;

justify-content
(specifies how we align our content horizontally within our container)
	justify-content: start; //default
	justify-content: end;
	justify-content: center;
	justify-content: space-between;
	justify-content: space-around;
	justify-content: space-evenly;

place-content (shorthand for align-content and justify-content)	place-content: center space-evenly; place-content: center;

CSS Grid

Container properties

grid-template-area (visual representation of our container, works hand in hand with grid-item property)	grid-template-areas: "header header header" "main main sidebar" "footer footer footer"; .header { grid-area: header;} .main { grid-area: main; } .sidebar { grid-area: sidebar; } .footer { grid-area: footer; }
grid-template: (shorthand for setting grid template rows, columns and areas)	grid-template: "header header header" 60px "main main sidebar" auto "footer footer footer"; 100px /[col-start] 1fr [col-2] 1fr [col-3] 1fr [col-end];
grid (shorthand for setting either grid-template-rows and grid-template-columns or grid-auto-rows and grid-auto-columns)	grid: 50px 100px / repeat(3, 1fr); grid: auto-flow 100px / repeat(3, 1fr);

CSS Grid

Grid item properties

grid-column-start
grid-column-end
(position our items in container according to the grid column lines)
	// items start's at 1 and span's across line no. 3
	grid-column-start: 1;
	grid-column-end: 3; or
	grid-column-start: col-start;
	grid-column-end: col-3;
	
	// span 3 lines across
	grid-column-end: span 3;

grid-column (shorthand for grid-column-start and grid-column-end)	grid-column: col-start / span col-2;
grid-row-start **grid-row-end** (position our items in container according to the grid row lines)	// items start's at row 2 and span's across row 5 grid-row-start: 2 grid-row-end: 5
grid-row (shorthand for grid-row-start and grid-row-end)	grid-row: 2/5;

CSS Grid

Grid item properties

align-self
(allows to position a single item vertically)
	align-self: stretch; //default
	align-self: start
	align-self: center
	align-self: end

justify-self
(allows to position a single item horizontally)
	justify-self: stretch; //default
	justify-self: start
	justify-self: center
	justify-self: end

place-self (shorthand for align-self and justify-self)	place-self: center start; place-self: center;
grid-area (short hand for item row-start, col-start as well as row-end and col-end)	// item start at (row 2, col 1) and spans across (row 5, col 2) grid-area: 2 / 1 / 5 / 2 ;

CSS Media queries and responsive design

Allows to control how our website looks across multiple device's

Your html page should have the tag :-

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

Before using media queries

3 main types of media queries :- • **speech** – for screen readers • **print** – for printing the webpage • **screen** – for the webpage	• @media speech{…} • @media print{…} • @media screen{…}

@media print
 	
Execute the following css when loading the print version of the webpage →
	@media print {
	body {
	background-color: #ffffff;
	color: #000000;
	}
	}

@media screen and min-width
 	
Execute the following css when width is 320px or higher than that →
	@media screen and (min-width: 320px) {
	h1 {
	text-align: center;
	text-decoration: underline;
	}
	}

CSS Media queries and responsive design

Allows to control how our website looks across multiple device's

@media screen and max-width
 	
Execute the following css when width is 319px or lesser than that →
	@media screen and (max-width: 319px) {
	p {
	color: red;
	}
	}

@media and orientation
 	
Execute the following css when device orientation is landscape →
	@media (orientation: landscape) {
	p {
	color: red;
	}
	}

CSS Transforms

It transforms elements by rotation, scaling, skew etc.

rotate( )
(rotate element by specified angle)
	// rotate by 45 degrees
	transform: rotate(45deg);
	
	// rotate by 2π radians
	transform: rotate(2πrad);

scale( )
(To increase/decrease size of element)
 	
No. > 1 (increases size)
No. < 1 (decrease size)
	// increase the size of element by 3 times its size
	transform: scale(3);
	
	// decrease the size of element by half
	transform: scale(0.5);
	
	// increase width by 3 and height by 4
	transform: scale(3, 4);

scaleX()
scaleY()
(changes the scale along width or height)
	// increase width of element by 2 times its width
	transform: scaleX(2);
	
	// increase height of element by 2 times its height
	transform: scaleY(2);

CSS Transforms

It transforms elements by rotation, scaling, skew etc.

skewX()
skewY()
(skew's the element either along X axis or Y axis)
	// skew element 20 deg along X axis
	transform: skewX(20deg);
	
	// skew element 20 deg along Y axis
	transform: skewY(20deg);

translate( )
(To move element around by right, left, down or up)
	// move element by 50 px to right
	transform: translate(50px)
	
	// move element by 50 px to left
	transform: translate(-50px)
	
	// move element by 50 px to right and 100px down
	transform: translate(50px, 100px)

translateX( ) **translateY( )** (move element along X or Y)	transform: translateX(50px) transform: translateY(100px)

CSS Transforms

It transforms elements by rotation, scaling, skew etc.

matrix() (shorthand for scale, skew and translate property) transform: matrix(scaleX, skewY, skewX, scaleY, translateX, translateY);	transform: matrix(2, 1, 1, 2, 70, -200);

3d transforms	div:hover {
	cursor: pointer;
	transition: all 2s linear;
	transform: rotateY(360deg);
	}

CSS Transitions

allows to smoothly change property values over a given duration.

transition-property
(Species what property to transition like width, height, bgcolor, etc.)
 	
You will specify transition properties in hover pseudo selector of element
	// change width of div to 250px when hovered upon
	transition-property: width
	div:hover{
	  width:250px;
	}
	
	// transition all properties specified in hover
	transition-property: all

transition-delay (specify a delay before the change is triggered)	// start the transition after 1s of delay transition-delay: 1s;
transition-duration (specifies how much time the transition should take)	// The overall transition should take 0.5 sec transition-duration: .5s

CSS Transitions

allows to smoothly change property values over a given duration.

transition-timing-function (controls how the element transitions)	// **ease-in:** transition start's off slower and then speeds up transition-timing-function: ease-in; // **ease-out:** transition start's normally and then speeds up transition-timing-function: ease-out; // **linear:** transition has constant speed transition-timing-function: linear;
cubic-bezier() : it allows to provide custom timing https://cubic-bezier.com/	transition-timing-function: cubic-bezier(.17,.67,.83,.67)
transition (Shorthand for all transition properties) transition: [property] [duration] [timing-function] [delay];	transition: all .5s liner;

CSS Animations

For animating our elements and creating cool effects.

animation-name (Specifies a name for our animation, you can name it anything)	animation-name: square;
animation-duration (specifies how long you want the animation to run)	animation-duration: 4s;
animation-timing-function (Specifies how the animation will run) It includes values like linear, ease-in, ease-out, cubic-bezier() etc.	animation-timing-function: linear
animation-delay (delay's the animation for required seconds)	animation-delay: 2s;
animation-iteration-count (specifies how many times we want the animation to play) you can specify number or infinite	animation-iteration-count: infinite

CSS Animations

For animating our elements and creating cool effects.

animation-direction (specifies in which direction to play the animation)	// **backward** - animation is played backward animation-direction: backward // **alternate** - animation is played forward first then backward animation-direction: alternate; //**alternate-reverse** - animation is played forward first then backward animation-direction: alternate-reverse;

animation-fill-mode
(specifies a style when animation is not playing)
	//animation will keep style from very last keyframe
	animation-fill-mode: forwards
	
	//animation will keep style from very first keyframe
	animation-fill-mode: backwards
	
	//animation will keep style from both keyframes
	animation-fill-mode: both

animation-play-state (species whether animation is playing or paused)	animation-play-state: paused; animation-play-state: running;

CSS Animations

For animating our elements and creating cool effects.

animation (Shorthand for animation) animation: [name] [duration] [timing-function] [delay] [iteration-count] [direction] [fill-mode] [play-state];	animation: square 4s linear 2s infinite alternate backwards running;

keyframes property

- Animations work hand in hand with keyframes property
- keyframes specify what property should be animated and how

@keyframes animation-name (You link your animation-name with keyframes)	animation-name: square; @keyframes square { }

CSS Animations

For animating our elements and creating cool effects.

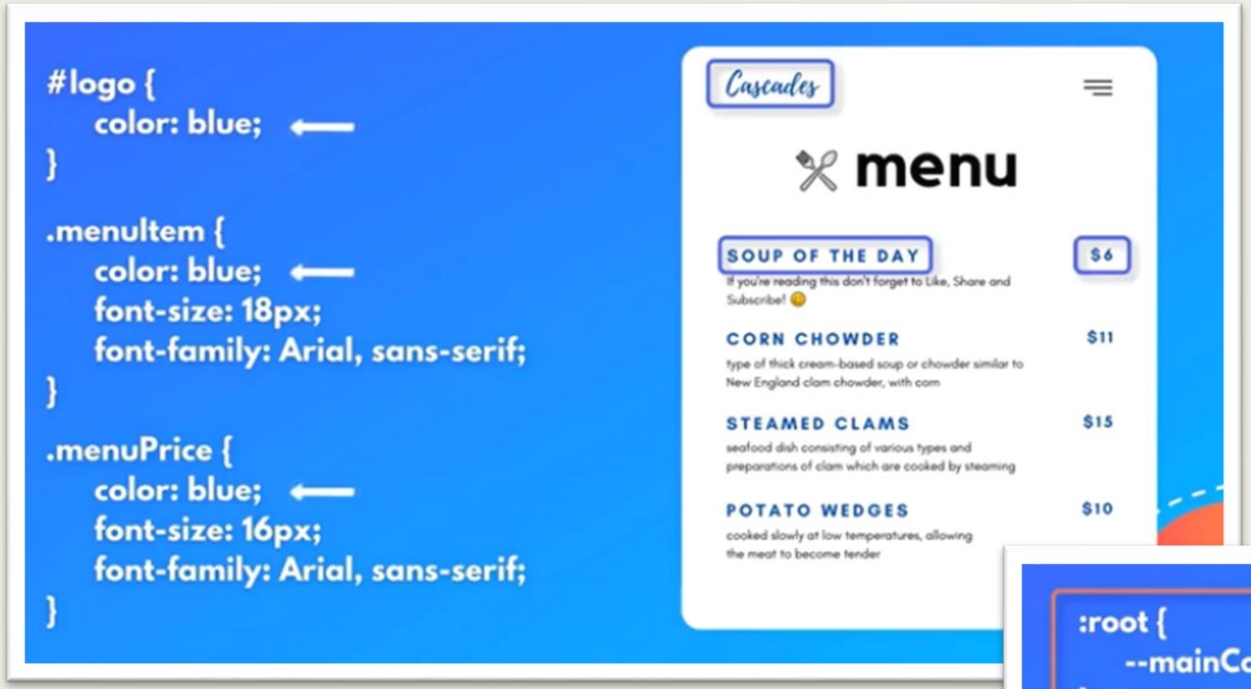
- All animations have an beginning and ending animation state
- There are 2 states for specifying animations states
  1. using from and to
  2. using percentages

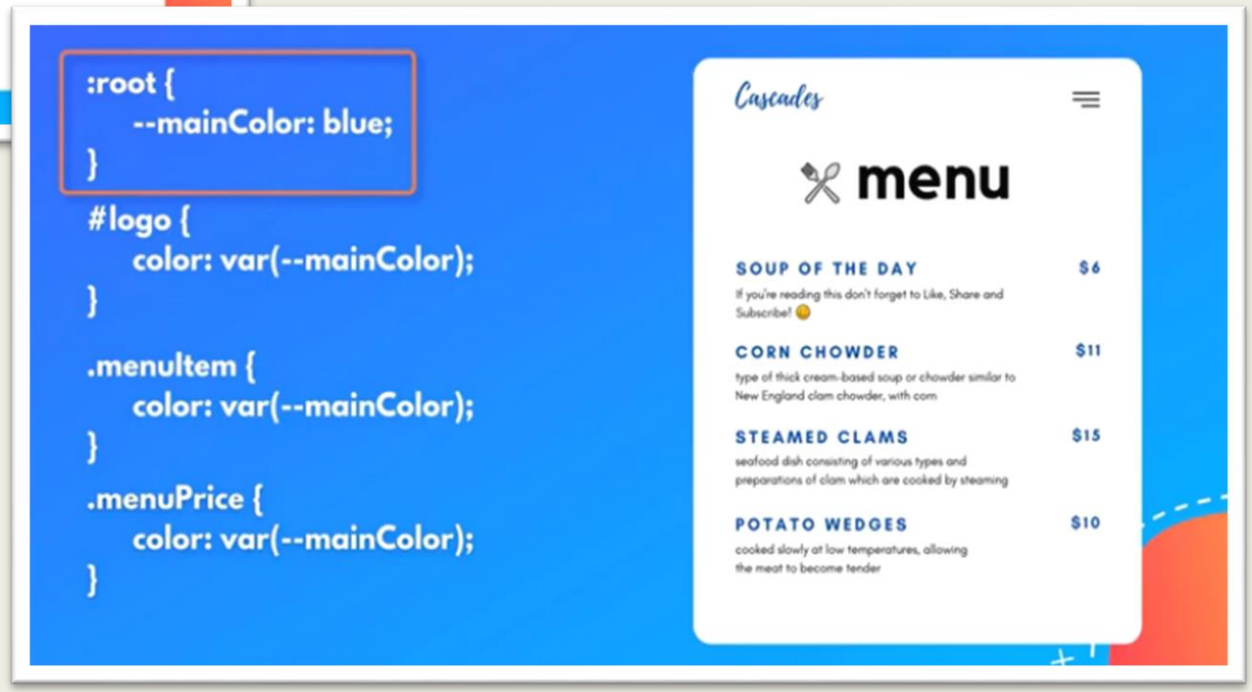
1st method – Using from and to
 	
from – beginning state
to – ending state
	// change color of square from red to blue
	@keyframes square {
	from {background-color: red;}
	to {background-color: blue;}
	}

2nd method - using percentages
 	
0% - starting state
100% - ending state
50% - between ending and starting state (middle state)
25% - between ending and middle sate
	// change color of square from red to lime to powderblue to orangered to blue
	@keyframes square {
	0% {background-color: red;}
	25% {background-color: lime;}
	50% {background-color: powderblue;}
	75% {background-color: orangered;}
	100% to{background-color: blue;}
	}

CSS Variables

are used to declare a style once then use it throughout our entire website





CSS Variables

are used to declare a style once then use it throughout our entire website

Declaring variables
Variable names always begin with - -
- - variable_name: value;
	//here root refers to <html> tag
	:root {
	--mainColor: orange;
	--mainFont: arial, sans-serif;
	--boxShadow: 12px 5px 10px grey;
	}

Using variables
property: var(--variable name)
	#logo{
	color: var(--mainColor);
	}

1. CSS variables are case sensitive	:root {
	--mainFont: arial, sans-serif;
	}
	
	// this wont work
	font-family: var(--mainfont);

2. You can use css variables with other variables too	--newColor: red; --newBorder: 2px solid var(--newColor) border: var(--newBorder);

CSS Variables

are used to declare a style once then use it throughout our entire website

3. You can also write variables in inline css styling	<h2 style = "border: var(--newBorder)">Text</h2>

4. While using css variables you can also give fallback values in case you made a typo or if the browser is not able to render the specified css	--newColor: red;
	
	color: var(--newcolor, blue);
	color: var(--newColor, blue);

5. you can also define css variables for a particular element or scope	#scoped{
	  --scopedFontSize: 30px;
	}
	
	#scoped button{
	  font-size: var(--scopedFontSize);