Sofascore Frontend Academy Lecture 02,
February, 2023

Advanced CSS

Alen Murtić

Sofascore

TABLE OF CONTENTS

01

Box model

CSS Box model

- Each element is a box defined with x, y, width and height
  - x and y mark the top left edge of the box
  - width and height are applied to content part of the box
- Where are content, padding, border and margin on the cats image?

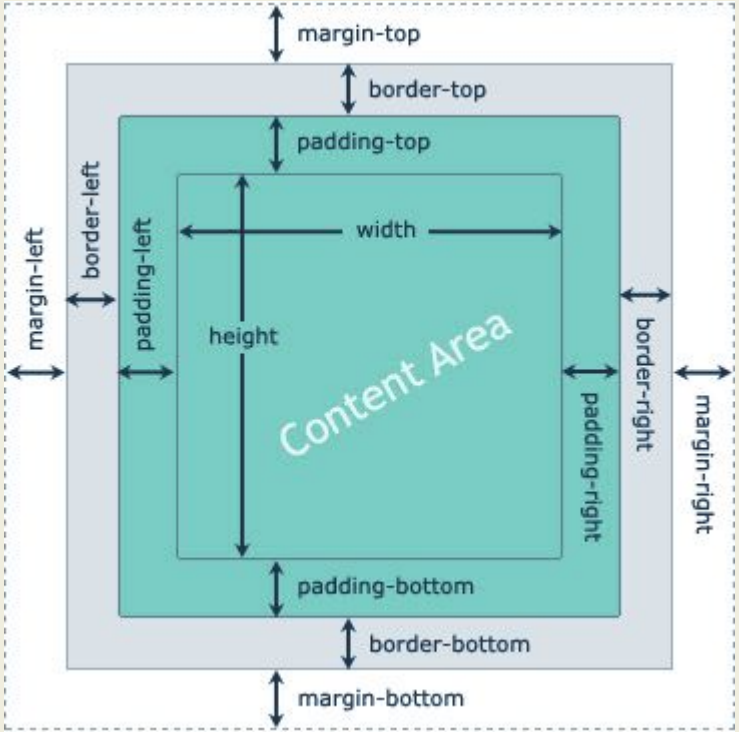
Box model

CSS Box model

- Content can be distributed in the box with padding
- Box can be spaced from other boxes with margin

  (e.g. spacing between sibling elements)

  - Margins of adjacent elements don't stack!!! They collapse. Bigger wins.
  - Rules of margins
  - Margin can be negative - a bit hackish
- Box can be made visible with border - draws line on the edge of the box



Box model

CSS Box model and box-sizing

- box-sizing property changes how browser calculates size of the element
  - Whether to border or padding included or excluded from the width
  - Default value is content-box, which includes only content
  - We at SofaScore use box-sizing: border-box -> includes content, padding and border
    - Very simple example why border-box can be superior: link

CSS Box Model Example

Sofascore

02

Display

Display

CSS Display Properties

- Display property is used to specify how should element behave in the parent element.
- Elements have default display properties defined in HTML specification
- Common display properties:
  - block
  - inline
  - inline-block
  - none - removes element from screen
  - grid, table, **flex** - special layouts

Display

CSS Display Properties

- Display: block - e.g. div, ul, main, p
  - Element will take as much space as possible in the single row
  - No two block elements in the same row
- Display: inline - e.g. span, b
  - Element will use minimal width needed
  - Ignores width and height set
  - Possible multiple elements in the same row
- Display: inline-block - e.g. img
  - Inline element which doesn't ignore width and height set

03

Flex

Flex

Flexbox - display: flex

- Displays element as block level container - it appears the same, children will be changed
- Applies flexible width (or height) and positioning to children
- Aligns children on defined axis - row, column
- Children positioning can be specified
    - Align all with start, center or end of the parent container
- If children take more space than available - define wrapping
- If children take less space than available - define layout

Flexbox - display: flex

- Best additional resources to learn flex
  - W3C - Schools - usually great, but it’s not their best work, in my humble opinion
  - A Complete Guide to Flexbox - phenomenal, great read even if you know flex
  - Flexbox Froggy game - when you feel ready, try this out
- Flex vs grid
  - Flex can do everything grid can, not exactly visa versa
  - Flex is more supported on older browsers, but it’s becoming moot point in 2023
  - Grid is better for very simple layouts (designed for Windows 8 start menu)

Flex

Flexbox - parent (flex container) properties

- Most important parent properties:
  - flex-direction - the direction (axis) of flex-container
    - row (default), row-reverse, column, column-reverse
  - justify-content - strategy of organizing children along the flex axis
  - align-items - strategy of organizing children along the other axis
  - flex-wrap - whether to wrap children in the next row/column when there isn't enough space
  - gap - default space between children on the container
    - Example of problematic Safari property, but less so in 2023 - Can I use link

Flex

Flexbox - child properties

- Most important children properties:
  - flex-grow - should the child grow and in what ratio
  - flex-shrink - should the child shrink and in what ratio
  - flex-basis - basis dimension for grow and shrink calculation
    - Can be used in combination with [max|min]-[width|height]
  - flex - shorthand for flex-grow, flex-shrink and flex-basis
    - e.g. flex: 3 2 50px;
  - align-self & justify-self - ignore how the container aligns/justifies children and set own rule

Flex

Practise makes perfect

Flexbox examples

04

Position and transition

Position and transition

CSS Position

- Type of positioning used for an element
- Used to define how should element be positioned based on its top, bottom, left and right, properties
- Default value: **static**
    - Ignore position properties
- z-index - tells browser how to display overlapping elements (higher number first)
    - default: first element on top

CSS Position - common values (other than static)

- relative -> position element relative to element's normal position (top, left, ...)
  - stays in normal flow, reserves its space
- fixed -> positions element relative to a screen (not content on the screen)
  - stays on the same place when page is scrolled
- absolute -> positions element relative to nearest positioned ancestor
  - positioned ancestor -> any element with position property different then static (default) !!!
  - common use with relative parent
  - removed from normal flow, no space reserved
- sticky -> positioning depends on scroll position, alternates between absolute and fixed
- MDN position documentation

Position and transition

CSS Transition

- Allow gradual/smooth transition between states, e.g. changing color, appearance, etc.
    - There are many many libraries which do more than basic CSS transition
    - Basic CSS is faster 😁
- THE most common problem in basic CSS:
    - Exiting element has to stay in HTML (DOM) until the animation finishes and then disappear
- We use Framer motion when tasked to do something more than basic CSS
    - Going way ahead of ourselves, but AnimatePresence is easy solution for the problem

Position and transition example

05

Responsive design

Responsive design

CSS Media

- Responsive design - page must look good on a large number of different devices
  - To consider different device capabilities - resolution, orientation, speed
  - Possibly different style rules by group
- Two strategies:
  - Progressive enhancement - “mobile first” - start with smaller screens and then handle wider devices
  - Graceful degradation - start with the best and largest devices, then handle all the problems

Responsive design

CSS Media

- Media queries - CSS checks for device capabilities
  - @media screen and (min-width: 400px) { ...will apply styles only on screens wider than 400px }
  - Much more than just screen size, e.g. @media (pointer: coarse|fine) {...} - link to docs
- Using viewport width and height in CSS: 100vh and 100vw - full viewport width and height
  - Great for desktop, a bit messy on mobile 😔
    - Problem: each browser handles disappearing toolbars and menus differently

Responsive design

CSS Media Breakpoints

- Media queries are usually not set at random screen sizes, but on few standardized values called media breakpoints
  - Standardized per site/development team/framework, not industry-wide
  - Bootstrap breakpoints
- Sofascore supports devices from 320px upwards
  - Lower resolutions work, but don’t look particularly great
- We’ll talk more about responsive design later

Responsive design

CSS Math Functions

- CSS supports 4 Math functions as values: calc, min, max, clamp
  - Practical Uses of CSS Math Functions: calc, clamp, min, max
  - e.g. setting main's min-height: calc(100vh - 30px)
    - We have a 30px header and the rest of the viewport should have content
  - Allows to dynamically calculate some value directly in CSS
    - Plain CSS is often faster than JS

Thank you for your attention!

Sofascore