Css

Kenan Ismail

November 8, 2022

Contents

1 Shit I dont know:

1.1 Select single child (+)

```
//if we have a p followed by a ul, and we want to select the ul this
                                        p + ul{
    color: blue;
}
```

1.2 Select all siblings (~)

```
//if we want to select all siblings of a ul that are p
ul ~ p{
    color: red;
        }
```

1.3 selecting a specific element but only using a class

```
//if we have multple p nodes and we want to select only some of them by class

p[class="lmao"]{
    color: green;
        }
```

1.4 Classes can be shared

```
<head>
  <style type="text/css">
    .a{
        color: yellow;
        }
    .b{
        color: purple;
        }
    .shared{
        padding: 5px;
        }
    </style>
  </head>

<body>
  <p class="a shared">Hi</p>
  <p class="b shared">Hello</p>
</body>
```

1.5 Pseudo classes (selecting an element in mutlple states)

Let's say we have a button, we want it blue but when we hover over it, we want to change the cursor to be the clicking cursor and make the opacity 80%

```
button{
    color: blue;
        }
button:hover{
    cursor: pointer;
    opacity: 0.8;
        }
```

1.6 Precedence

1.6.1 ID

Worth 100 points

1.6.2 class

worth 10 points

1.6.3 tag

worth 1 measly point

1.7 !important

can be applied to a CSS property, bypasses all precedence.

1.8 Block vs Inline

- Block respects width and height properties, but inline (an asshole) doesnt.

1.8.1 inline block

Basically an inline element but it respects width and height

1.8.2 Can be set using the display property

1.9 box-sizing: Border box vs Content box

Content box REALLY respects width and height properties, it tries to fit the padding inside it.

1.10 position

can be:

1.10.1 static

1.10.2 absolute

- if it is a child of a relative element, then its position will depend on that element, otherwise it will be like fixed (dependent on viewport)

1.10.3 relative

- Provides a 'z-index' basically layers, the higher the index the more it is on top.
- margins are also now 'relative' to the object itself, for example if an object exists on x:10y:10 and we make a margin in a relative position of 10, then the y will be 20 not 10.
- the child of a relative can have an absolute position

1.10.4 fixed

- Used for navbars, commonly with width=100vw;
- You also must specify the offset from its parent element (top: 0;)
- It no longer takes space in the body

1.10.5 sticky

1.11 CSS measurement units

- pixels pixels
- percentages how much % of the viewport container (if the viewport is resized the element will also be resized)

- rem (fonts) % increase/decrease of font size dependent on how many pixels it's set at (default is 16px)

  so 1.2em will be 120% of 16px, 20% increase or 1.2*16px

- em (fonts) same as rem but it is now relative to the parent element

1.12 font-weight

1.13 border-radius

Use for rounded courners (in px)

1.14 box-shadow

use for shadow

box-shadow: none|h-offset v-offset blur spread color |inset|initial|inherit;

1.15 vertical-align

1.16 overflow

can be set to **hidden** to hide elements that overflow out of the parent container.

can also be set to **auto** to scroll through overflowing content

2 4 step method to write CSS

- How do i want to arrange the elements on the page?
- How would it look in HTML?
- How can i implement it in CSS instead?
- How should each element be styled?

3 Responsive design

3.1 Breakpoints

basically like an if condition for what style of page to display (depending on screen size, etc)

3.1.1 basic syntax

```
@media /* logical conditions*/ (min-width: 768px){

 }
```

3.1.2 At-rules

1. @media is considered to be an at-rule, it selects which type of device, for example it can be @media screen, @media speech, @media all, etc
2. @supports checks if a browser supports a feature

3.1.3 logical operators

we have logical operators in CSS, and or
For example this code

```
@media only screen and (min-width: 400px) and (max-width: 600px){
    *,*{
        color:red;
    }
        }
```

Translates to iff the device is a screen, and its width is between 400 to 600 px, style all elements red.

3.2 Checklist

- Fluid layouts - does the content resize on different screens?
- Images - do they maintain a good aspect ratio and size on different screens?
- Text - The text should be comfortable to read on any screen size (rule of thumb, 16px p size, 700px max container size)
- Overflow - nothing should overflow
- Full-width mobile content - dont show columns on mobile, make it all 100% width of vp
- General usability

- Are buttons big enough?
- Are most important things showing first?
- Use your site on an actual phone, you can quickly find out if something needs work.

4 Flexbox

Flexbox is a way of organizing elements on one axis.

4.1 Parent-child

If you set an element's display property to 'flexbox', then all its children will be flex items. And element can be a flex item only if its parent is a flex container.

4.2 Properties

4.2.1 flex-direction

Sets the axis, can be row wise or column wise.

4.2.2 justify-content

Imagine that you have a white page in ms word, this is your container, and the children is the text. The same way you justify stuff there, applies here, sort of.

1. : start the default, makes elements start from the origin of the axis
2. : end Makes elements begin from the end of the axis
3. : center Centers the elements (to the center of the container)
4. : space-between makes the elements spread throughout the whole container with spaces in between
5. : space-around same as above but puts spaces at the ends too
6. : space-evenly inverse of space-between

4.2.3 align-items

For aligining items on the cross axis. (vertically)
setting width and height manually overrides this

1. normal stretch
2. stretch stretch item to container
3. center
4. flex-start
5. flex-end
6. baseline

4.2.4 flex-wrap

1. wrap the way to handle overflow in flexbox, makes stuff appear in a grid.

   Important If we set this, then we no longer use align-items, we use **align-content** instead!

4.2.5 align-self

when you want to change the alignment of 1 flexitem only

4.2.6 flex-grow

how much to use up of available space using free space units

4.2.7 flex-shrink

how much to shrink a flexitem using overflow units. Keep in mind that each item has a minimum size and wont shrink past that.

4.2.8 flex-basis

by default it's auto, but basically this takes precedence over any other flex sizing property to change the size

4.2.9 order

used to change the order of the flexitems

5 CSS grid

First of all, set the display property to grid.

5.1 grid-template-rows and grid-template-colums

creates a grid with a defined number of rows and columns, can be used in conjunction with the repeat() css function.

```
someselector{
    display: grid;
    grid-template-rows: 30px 30px 30px;
    grid-template-columns: 20px 30px;
    /*creates a 3x2 grid*/
}

anotherselector{
    display: grid;
    grid-template-rows: repeat(3,30px);
    grid-template-columns: repeat(2,20px);
    /*same as the above*/
}
```

5.2 fractional units (Xfr)

a unit of scale in css grid, the above example but with 1fr instead of 30px

5.3 Placing elements

5.3.1 grid-column-start and grid-column-end

Basically these properties tell us when to begin the element and when to end it inside our css grid. this is to be used by **the children** of the css grid element.

5.4 The grid area

what we defined above is actually the grid area. When we set the width and height of our child elements, it needs to fit inside the grid area.

5.4.1 justify-items and align-items

to be used in the parent (css grid) element, affects the placement of the child elements within their grid area, can be set the same values as its counterpart in flexbox. (default is stretch)

5.4.2 justify-self and align-self

the same as the above but to be used by the child elements (if you want to target only one element)

5.5 justify-content and align-content

Acts on the css grid element (parent) itself to position the element. same properties as flexbox.