**CSS Notes (should be reorganized)**

|  |  |  |  |
| --- | --- | --- | --- |
| CSS units and dimensions | % height/width | * Is set as **% of the element parent’s height/width**. Therefore, for desired outcome to follow, the parent’s dimension in view should be set explicitly. | |
| Set height as % to viewport height | * Set all the element ancestors’ heights to 100%. * Use **vh** units for the same purpose. | |
|  |  | |
| display | block |  | |
| inline-block |  | |
| flex |  | |
| table-cell | Using CSS we 'simulate' table display behavior, since tables support vertical alignment. | |
| block level element | Occupies the entire space of its parent container, thereby creating a "block". They start on new lines. | | |
| inline level element | Occupies only the space bounded by the tags that define the inline element. Can start anywhere in a line | | |
| block-level box |  | |  |
| **flexbox** concepts | General info | | Flexbox consists of flex containers and flex items.  A flex container is declared by setting the display property of an element to either flex (rendered as block) or inline-flex (rendered inline).  <https://css-tricks.com/snippets/css/a-guide-to-flexbox/> |
| Restrictions | | **float**, **clear** and **vertical-align** have no effect on a flex item. |
| **Align text** inside a flex item (display: flex) | | Vertical:  align-items: center  Horizontal:  justify-content: center  text-align: center |
|  | **flex** | | **flex: 0 0 auto;**  *means the same as:*  flex-grow: 0;  flex-shrink: 0;  flex-basis: auto; |
| Force image resize and keep aspect ratio | <http://stackoverflow.com/questions/12991351/css-force-image-resize-and-keep-aspect-ratio>  .container {  display: block;  width: 100%;  height: auto;  position: relative;  overflow: hidden;  padding: 34.37% 0 0 0; /\* 34.37% = 100 / (w / h) = 100 / (640 / 220) \*/  }  .container img {  display: block;  max-width: 100%;  max-height: 100%;  position: absolute;  top: 0;  bottom: 0;  left: 0;  right: 0;  } | | |
| Vertical centering | <https://philipwalton.github.io/solved-by-flexbox/demos/vertical-centering/> | | |
| Align, justify |  | | |
| Text shadow and glow effect | **text-shadow: 1px 1px 2px black, 2px 2px 10px white;**  Example explained:  - 2 effects applied  - Each effect: shadow x, shadow y, glow, color | | |
| min-height: max-content; | min-height: max-content; | | |
| Line box | The rectangular area that contains the boxes that form a line is called a **line box**.  When several inline-level boxes cannot fit horizontally within a single line box, they are distributed among two or more vertically-stacked line boxes. Thus, a paragraph is a vertical stack of line boxes.  When an inline box exceeds the width of a line box, it is split into several boxes and these boxes are distributed across several line boxes. If an inline box cannot be split (e.g., if the inline box contains a single character, or language specific word breaking rules disallow a break within the inline box, or if the inline box is affected by a white-space value of nowrap or pre), then the inline box overflows the line box. | | |
| Browser quirks mode (box sizing) | (Uses a different box model compared to the modern one.)  The "quirks" box model worked like this: width = actual visible/rendered width of an element's box height = actual visible/rendered height of an element's box The border and padding values were moved inside the element's box, cutting into the width/height of the box rather than expanding it.  <https://css-tricks.com/box-sizing/>  This box sizing model went out of fashion with the spread of responsive design approach. It relied on pixel values, and was not very convenient. It can nevertheless be applied by setting the “**box-sizing**” property to “**border-box**”. | | |
| Content-box vs. border-box (**box sizing**) | **Content-box** is a default model in modern browsers.  Border-box squeezes padding and border inside the box width setting value. The idea is the same as used in the old-fashioned quirks mode. | | |
| Pseudo-elements, general info |  | | |
| vendor prefixes | if you need to support older versions of Safari (< 5.1), Chrome (< 10), and Firefox (< 29), you should include the prefixes **-webkit** and **-moz** (before some CSS property names, e. g. box-sizing etc). | | |
| z-index | The z-index property specifies the stack order of an element.  In order for z-index to work, you'll need to give the element a position:absolute or a position:relative property.  Flex items can participate in a z-index stacking order even with position: static. | | |
| perspective-origin | The **perspective-origin** property defines where a 3D element is based in the x- and the y-axis. This property allows you to change the bottom position of 3D elements.  When defining the perspective-origin property for an element, it is the CHILD elements that are positioned, NOT the element itself.  Note: This property must be used together with the *perspective* property, and only affects 3D transformed elements! | | |
| perspective | The perspective property defines how many pixels a 3D element is placed from the view. This property allows you to change the perspective on how 3D elements are viewed.  When defining the perspective property for an element, it is the CHILD elements that get the perspective view, NOT the element itself.  Note: The perspective property only affects 3D transformed elements!  Tip: Use this property together with the perspective-origin property, which allows you to change the bottom position of 3D elements. | | |
| transform-origin | The transform-origin property allows you to change the position of transformed elements.  2D transformations can change the x- and y-axis of an element. 3D transformations can also change the z-axis of an element.  Note: This property must be used together with the *transform* property. | | |
| transform | The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements. | | |
|  |  | | |
|  |  | | |
|  |  | | |