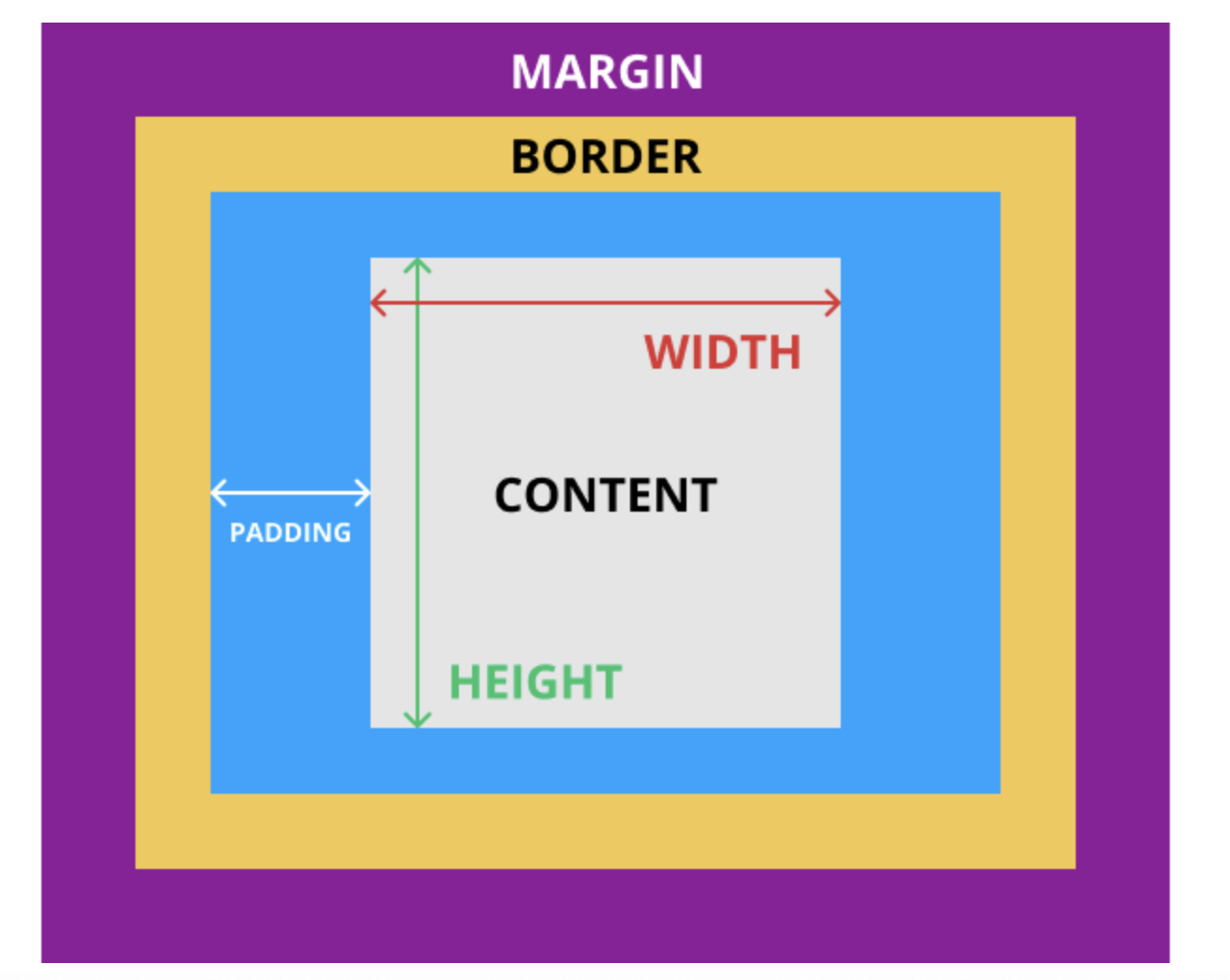
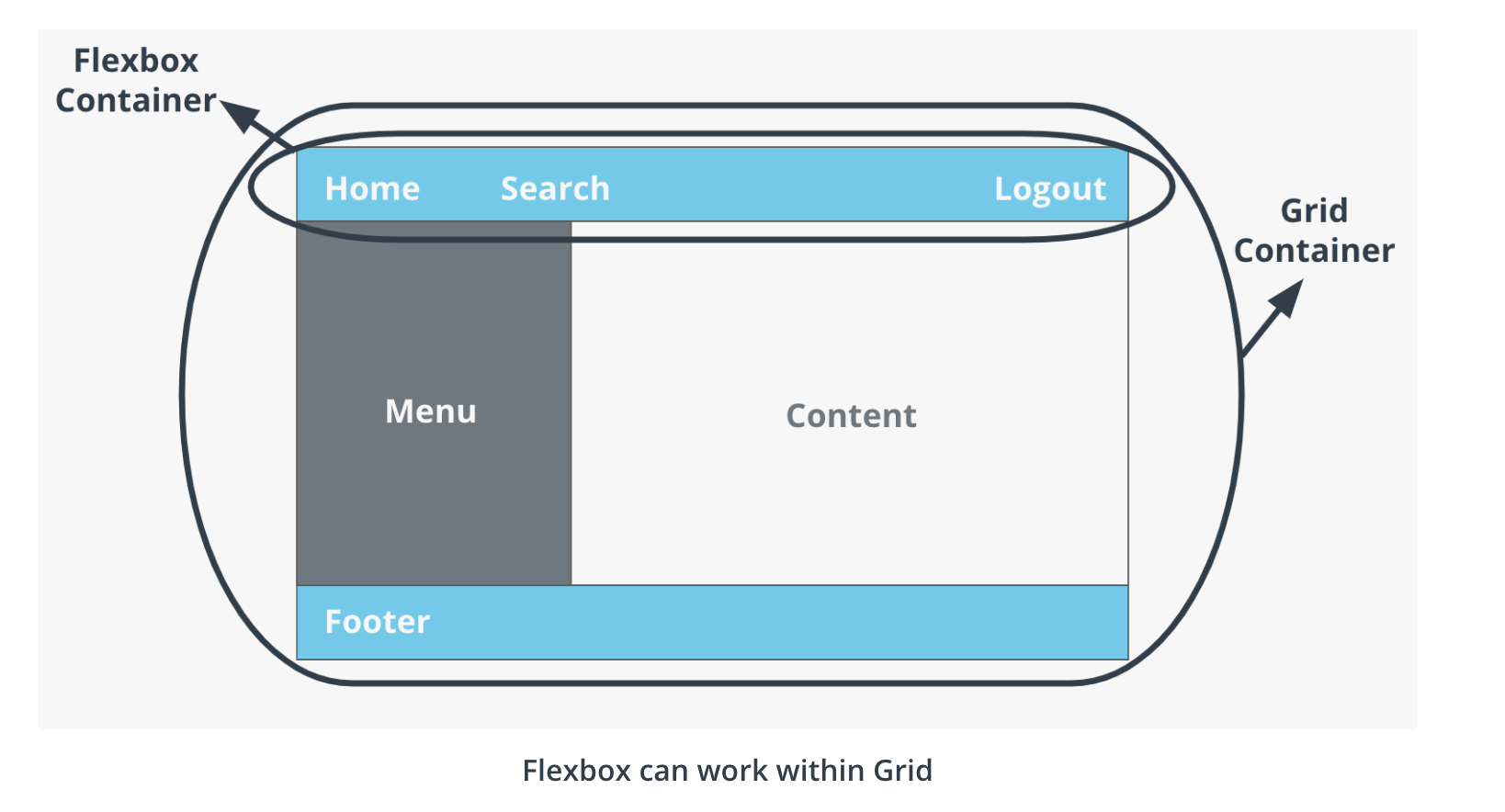
<https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox>

<https://css-tricks.com/>

<https://developer.mozilla.org/en-US/docs/Web/CSS/background-image>

<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox#Alignment_justification_and_distribution_of_free_space_between_items>





* Grid is two dimensional, while Flex is one
* Grid is layout first, while Flexbox is content first
* Flex is for components of an app, Grid is for the app layout itself

<https://medium.com/youstart-labs/beginners-guide-to-choose-between-css-grid-and-flexbox-783005dd2412>

<https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf>

<https://css-tricks.com/css-grid-replace-flexbox/>

A core difference between CSS Grid and Flexbox is that — CSS Grid’s approach is **layout-first** while Flexbox’ approach is **content-first**.  
*If you are well aware of your content before making layout, then blindly opt for Flexbox and if not, opt for CSS Grid.*

<https://cssgrid-generator.netlify.com/>

*/\* row start/column start/ row end/ column end \*/* grid-area: 1/2/3/3;

<https://css-tricks.com/snippets/css/complete-guide-grid/>

Look at repeat 1fr and minmax for the row size for example

Grid-auto-rows = minmax…

<https://gedd.ski/post/grid-item-placement/>

<https://rachelandrew.co.uk/archives/2015/02/04/css-grid-layout-creating-complex-grids/>

<https://gridbyexample.com/examples/example21/>

VERY GOOD ARTICLE:

<https://drafts.csswg.org/css-grid/#overview>

Example from article above: The following declares a grid with as many rows of at least 5em as will fit in the height of the grid container (100vh). The grid has no explicit columns; instead columns are added as content is added, the resulting column widths are equalized (1fr). Since content overflowing to the right won’t print, an alternate layout for printing adds rows instead.

main {

grid: repeat(auto-fill, 5em) / auto-flow 1fr;

height: 100vh;

}

@media print {

main {

grid: auto-flow 1fr / repeat(auto-fill, 5em);

}

}

**Another way to declare the grid-items – this tells where the start and end are ;-)**

/\* equivalent layout to the prior example, but using named lines \*/

#grid {

display: grid;

grid-template-columns: 150px [item1-start] 1fr [item1-end];

grid-template-rows: [item1-start] 50px 1fr 50px [item1-end];

}

#item1 {

grid-column: item1-start / item1-end;

grid-row: item1-start / item1-end;

}

With media queries and breakpoints, you just need to put the CSS you want to change the rest can stay as is

<https://developer.mozilla.org/en-US/docs/Web/CSS/Viewport_concepts#What_is_a_viewport>

<https://www.w3schools.com/cssref/css_selectors.asp>

<https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag>

@**media(feature:value)** - media query with breakpoint

Start setting up the content for the smallest devices and go up from there

Ideally put the media query at the bottom to make sure we don’t get overwritten by the default

<https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries>

DOM MATERIAL

<https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>

<https://html.spec.whatwg.org/multipage/parsing.html#parsing>

<https://www.w3.org/TR/?tag=dom#w3c_all>

<https://www.ecma-international.org/ecma-262/#sec-global-object>

[https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById](https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById" \t "_blank)

<https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByClassName>

<https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>

Element is the base class descendent of the Node therefore inheriting its properties…

<https://developer.mozilla.org/en-US/docs/Web/API/Element>

WebAPI: <https://developer.mozilla.org/en-US/docs/Web/API>

<https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent>

TextContent returns all the text

InnerText returns how it would be seen visually so includes CSS…

InnerHTML needs to be used to change the HTML else the tags won’t be rendered

<https://www.smashingmagazine.com/2007/07/css-specificity-things-you-should-know/>

The setAttribute() method is not just for styling page elements. You can use this method to set any attribute for an element. If you want to give an element an ID, you can do that!:

Liste of Events:

<https://developer.mozilla.org/en-US/docs/Web/Events>