Flexbox advice and best practices

Flexbox advice and best practices

Here are some quick tips to help you get the most out of flexbox.

Remember the minimum

If you have a parent element that contains some child elements, then putting `display:flex;` on the parent is all that is needed to get started. The parent itself behaves like a normal block level element, it is the flex container, and all of its children are flex items.

It's not a bad idea to specify the `flex-flow` for the flex container (`flex-flow: row wrap;`), but it isn't required.

It's generally a good idea to also initialize the `flex` property on the flex items (e.g. `flex:1`), but again, this isn't required.

Use variable dimensions on flex items instead of explicit ones

This advice may not apply to images and may not be appropriate for every flex item. However, for most flex items, try to avoid using explicit `width` and `height` properties. Instead, use the `flex-basis` to set a desired dimension (e.g. `flex: 1 1 200px;`). Or consider using `min-width` (or `max-width`) and `min-height` (or `max-height`).

Doing so will make your flex items a bit more malleable. In CSS professional parlance, this is called being "responsive".

Do not overconstrain your flex items. Let the browser work for you.

This is a follow-on to the previous piece of advice. With flexbox you give the browser some general guidelines and allow it to figure it out. (*"Hey, put these in a row, I guess they can wrap. This item should be 80 pixels wide generally, and this item should always be at least 200 pixels wide, and this item should be first."*). If you overly constrain the flexbox container by disallowing growing, shrinking, and setting explicit dimensions, then the results may be not optimal. Don't micromanage, let the flexbox do its job.

Remember also, that if you overdetermine the dimensions for an element, then you may also have to start managing its `overflow` setting and its box model. Who needs more worry? Underconstraining is the path to happiness.

Thinking of using inline-block? Consider flexbox instead.

If you are considering changing the `display` of several elements to be `inline-block`, that may indicate that you should be using flexbox instead. Perhaps their parent should be set to be a flex container and they should be flex items.

Centering? Maybe flexbox

If you need to center some content horizontally, then the previous section on centering may help. It discusses the various options for inline or block level elements. However, a flex container and a flex child is also a possible approach. The flex container property `justify-content:center;` might help. See the optional section on that. Both approaches (flexbox and "traditional") have their advantages in various situations. Use your judgement.

If you need to center something vertically, unless it is an inline element or the content of a table cell, the best answer is almost certainly flexbox. See the `align-content` and `justify-content` properties in the advanced optional flexbox material.

AVOID margin: auto on flex items

`margin:auto` prevents align-self from working (a very handy flexbox property covered in the Optional section). There are effective ways of using auto margins though, just do so with care: https://medium.com/@samserif/flexbox-s-best-kept-secret-bd3d892826b6#.wzrvpxpqv

Try to keep your flexbox usage simple

You can see here: http://caniuse.com/#search=flexbox that flexbox is widely supported across all the modern browsers. However, Internet Explorer had bugs in its support (but they are fixed in Edge). And the flexbox standard has had some changes since it

was first proposed. For this reason, try to keep your usage of flexbox close to what is covered in this course material, and, as with all CSS, be sure to always test in as many browsers as possible.

External resources

- Flexbox documentation on MDN (available in many languages!)
- What Happens When You Create A Flexbox Flex Container? (by Rachel Andrew - 2 August 2018)