Responsive Web Design with Flexbox and CSS Grid v3

Part 1: Basic Responsive Web Design

Jen Kramer @jen4web

Page Layouts 2001

- HTML 3.2, some HTML 4
- No CSS supported in browsers
- JavaScript was a ”toy” used in image rollovers
- Netscape 4 vs. Internet Explorer 3 (IE 6 released)
- Macromedia Dreamweaver, MS FrontPage
- 640px vs. 800px screen sizes
- FTP documents to web host
- **Table-based layouts**

Page Layouts 2008

- XHTML 1.0 Transitional, discussion of HTML 5
- CSS 1, CSS 2.1
- jQuery, jQuery UI
- Chrome 1, Firefox 3, IE 7 (IE 6 still popular)
- WordPress, Joomla, Drupal
- iPhone was released in 2007; Android in 2008
- cPanel
- **Float-based layouts**

Responsive Design, 2010

- Defined by three characteristics
  - Flexible grid-based layout
  - Media queries (CSS3)
  - Images that resize

One codebase to style the same web page for all devices, regardless of screen size

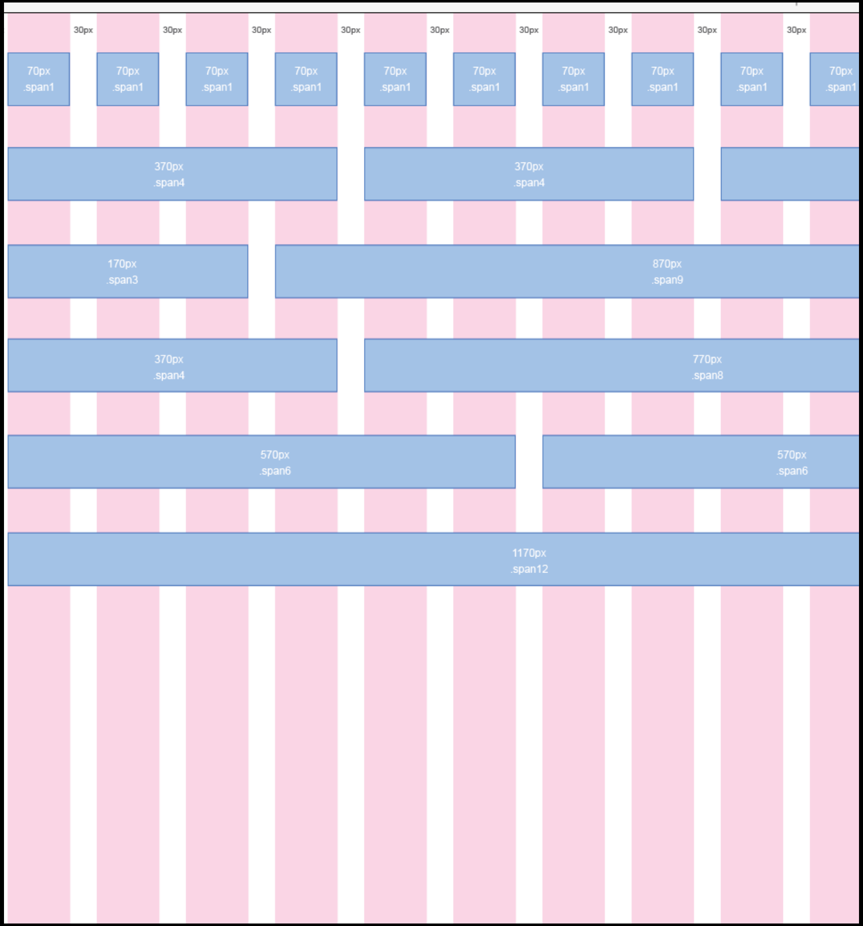
Grid-based layout

2010: Grid-based layouts achieved with floats and hand-coding

2014-2017: Grid-based layouts with Flexbox/Bootstrap

2018: Support for Grid

2023: Container queries (size only, Feb 2023)
Subgrid (Sept 2023)



Images that resize

- Images should change size, based on screen resolution
- Today's solutions include `<picture>`, *`sizes`*, and *`srcset`*, part of HTML spec

CSS Media Queries

- Browser reports screen resolution
- Based on current width, serve a stylesheet with layout for that width
- No JavaScript involved

- PS – media queries go beyond width!
  - Height
  - Aspect-ratio
  - Orientation
  - Colors

How does this hold up today?

- Grid-based layout associated with frameworks, CSS Grid
- Images that resize are often forgotten
- Devs try to avoid media queries
- CONTAINER QUERIES / SUBGRID WILL SOLVE ALL (ahem)

Consider these as guiding principles

CSS Grid-based layout

Sometimes Flexbox

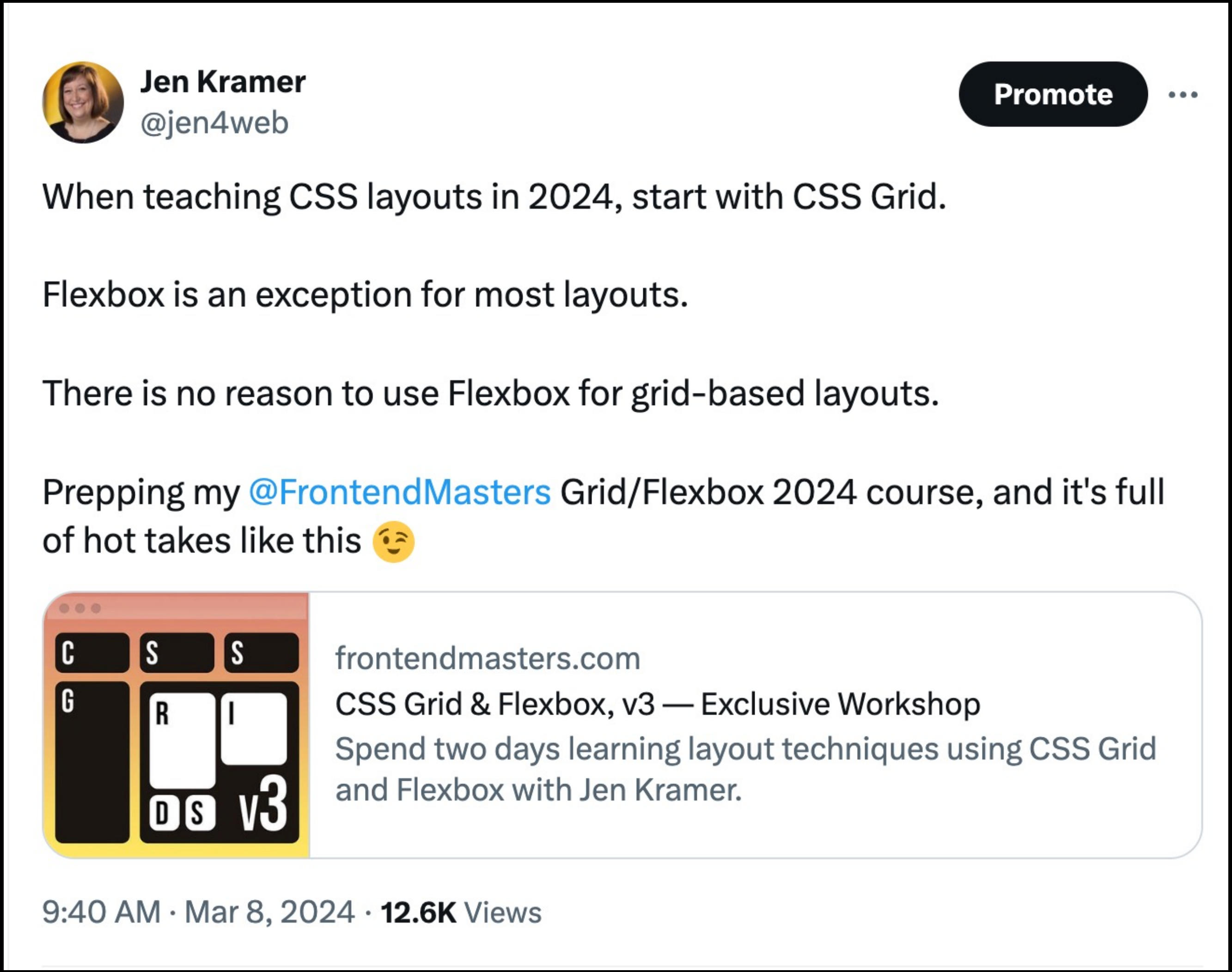
Images that resize

`<picture>, sizes, srcset`

Media queries

Container queries

Flexbox vs Grid



Jen Kramer
@jen4web

Promote

When teaching CSS layouts in 2024, start with CSS Grid.

Flexbox is an exception for most layouts.

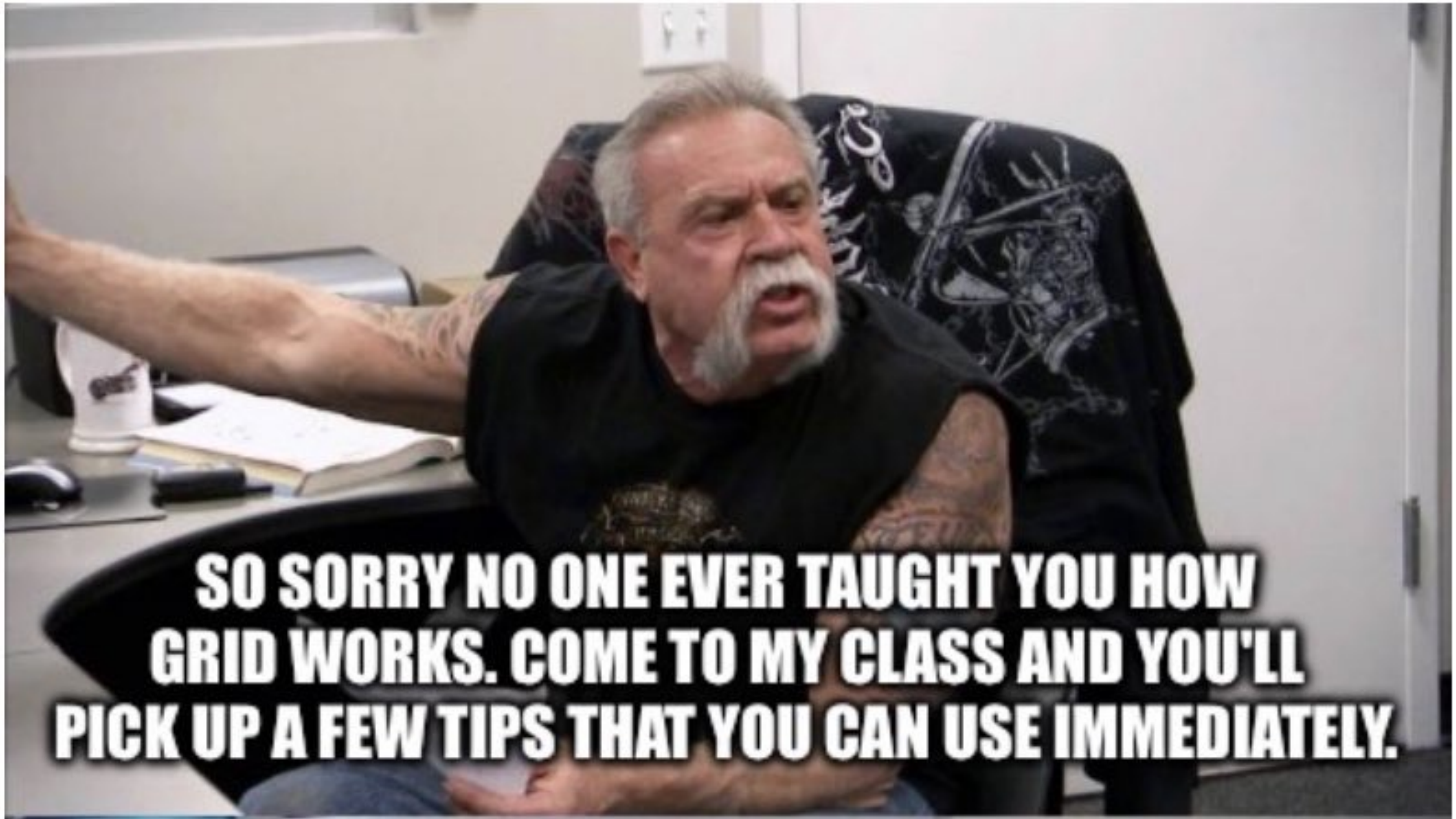
There is no reason to use Flexbox for grid-based layouts.

Prepping my @FrontendMasters Grid/Flexbox 2024 course, and it's full of hot takes like this 😉

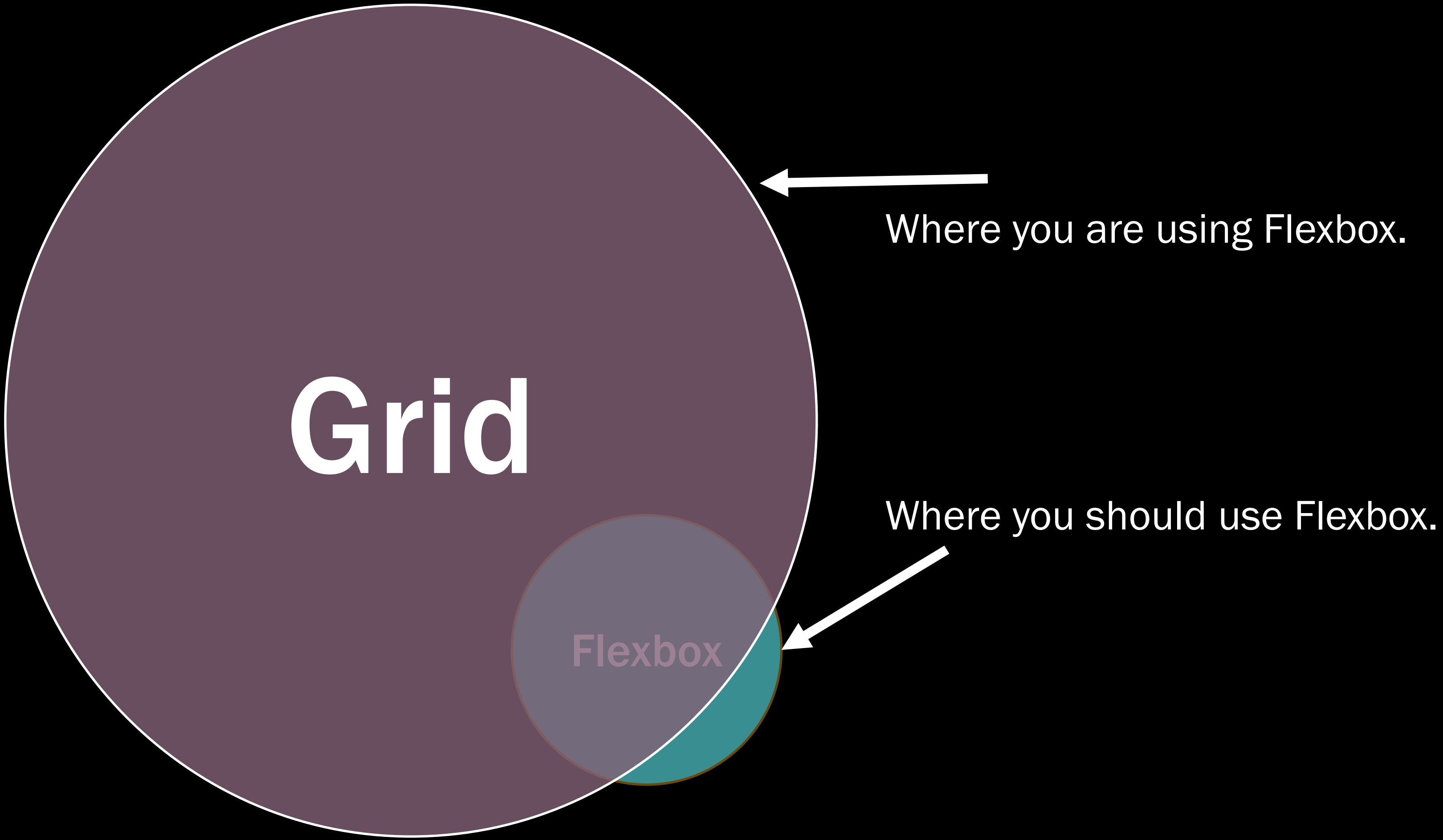
frontendmasters.com
CSS Grid & Flexbox, v3 — Exclusive Workshop
Spend two days learning layout techniques using CSS Grid and Flexbox with Jen Kramer.

9:40 AM · Mar 8, 2024 · **12.6K** Views

“Disagree. In my experience grid is still the exception, there's more work involved for varying viewport sizes using grid vs flex, and complete shifts in layout construction.”


SO SORRY NO ONE EVER TAUGHT YOU HOW GRID WORKS. COME TO MY CLASS AND YOU'LL PICK UP A FEW TIPS THAT YOU CAN USE IMMEDIATELY.

Grid is where you should start.
Leave Flexbox to do what Grid can't.



Things change. Sorry.

- Your bootcamp taught you Flexbox in 2015 because of browser support.
- Bootstrap used Flexbox because of browser support.
- Grid was designed for web page layouts. HTML tables, floats, and Flexbox were not designed for web page layouts.
- Grid is the layout solution we'll be using in CSS going forward. Sorry for 20 years of hacks before this. Who knew we wanted to lay out web pages?

“Flexbox is better for most things.”

“I can barely keep up with how fast things are changing in JavaScript. I don’t want to learn CSS.”

Grid

Grid is best for…

- A series of boxes that are the same width
- Layouts that span across rows and/or columns
- **Siblings that need to overlap or cover each other**
- **This should be your default layout method**

The key to Grid (and Flexbox)

- There is an HTML relationship between parents and children.
  - We leverage this in CSS selectors.
  - What layout properties available to you depends on if you're working with a parent or a child.

- Review your HTML structure carefully before creating your grid (or flexbox layout).

Grid quirks

- There are several types of syntax with Grid that accomplish the same thing!
  - Choose the syntax that you like best.
  - What's most understandable for you?

- Grid does its best to guess at what you want.
  - Even if you haven't specified the correct number of columns and/or rows, Grid will try to place the next box in the right location.

Grid properties

Parent (Grid Container)

display: grid | inline-grid;

grid-template-columns
grid-template-rows
grid-template-areas
column-gap
row-gap
gap: <row-gap> <column-gap>

Align box along row/inline axis:
justify-items: start | end | center | stretch

Align box along column/block axis:
align-items: start | end | center | stretch | baseline

Children (Grid Items)

grid-column-start

grid-column-end

grid-column (start and end numbers)

grid-row-start

grid-row-end

grid row (start and end numbers)

grid-area

Align content along row/inline axis:

justify-self: start | end | center | stretch

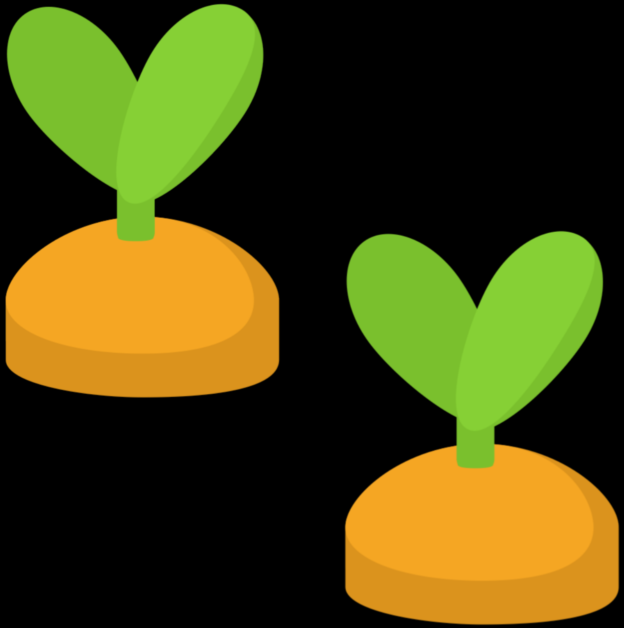
Align content along column/block axis:

align-self: start | end | center | stretch

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

Games to practice Grid **properties and values**:

- Grid Garden
  https://cssgridgarden.com/
- Grid Attack
  https://codingfantasy.com/games/css-grid-attack
- Grid Critters
  https://gridcritters.com/



Reflection

- Now that you’ve seen 3 types of syntax for coding CSS Grid, which do you like best and why?
- Can you think of use cases for any of these syntaxes specifically?

CSS Grid Review

- 3 types of syntax for Grid: hard numbers, span notation, named areas
- Intrinsic vs extrinsic grid – Grid will try to place elements in the first available space unless otherwise specified
- HTML parents and children determine Grid properties available

Media queries & RWD

Media Queries

Mobile-first

- Build your mobile styles **outside** of media queries
- Mobile styles are the default
- min-width media queries
  - May add styles not previously present
  - May override style settings that came before
  - Order MQ’s from smallest number to largest number

Desktop-first

- Build your desktop styles **outside** of media queries
- Mobile styles are the default
- max-width media queries
  - May add styles not previously present
  - May override style settings that came before
  - Order MQ’s from largest number to smallest number

Website Planning

Mobile-first

Plan content starting with mobile devices.

Add content as screen gets bigger if desired. This content should be non-critical to mobile.

Championed by LukeW (Luke Wroblewski), 2011

Desktop-first

Plan content starting with desktop/large screen devices first.

Scale content back for mobile.

(How most people started working on RWD content in 2011.)

Which approach is better?

- When planning content, use mobile first approaches.
- When building websites, use whichever approach makes sense.
- It is OK to mix approaches!
    - Example: use max-width MQ’s for navbars, min-width MQ’s for most of the rest of the page
    - Example: We started with a desktop layout and now want to add mobile – use desktop-first approach

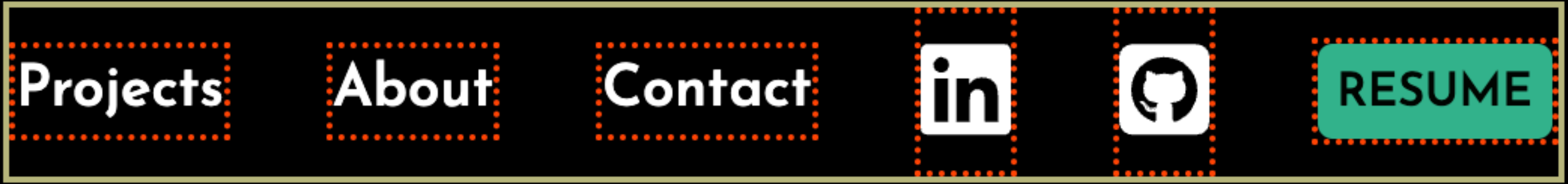
Flexbox

Flexbox is best for…

- A series of boxes that are NOT the same size
- A series of boxes that are NOT in an even-sized grid
- When the same space between elements is important, not the same width of each element.
- **Flexbox wasn’t designed to be locked down for layouts!**

It’s 2024. Time to use Grid for most layouts.

A series of boxes that are NOT the same size
When the same space between elements is important.



A series of boxes that are NOT in an even-sized grid

CSS
Boot Camp
<HTML>
{CSS}
CSS

Flexbox properties

Parent (Flex Container)

display: flex | inline-flex;

flex-direction: row | row-reverse | column | column-reverse;

flex-wrap: wrap | nowrap | wrap-reverse;

flex-flow (shorthand for flex-direction and flex-wrap)

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;

align-items: flex-start | flex-end | center | baseline | stretch;

align-content (cross axis - adjust to largest item): flex-start | flex-end | center | stretch | space-between | space-around;

Children (Flex Items)

order: <integer>;

flex-grow: <number>;

flex-shrink: <number>;

flex-basis: <length> | auto;

flex: shorthand for grow, shrink, and basis (default: `0 1 auto`)

align-self: overrides alignment set on parent
auto | flex-start | flex-end | center | baseline | stretch;

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Games to practice Flexbox **properties and values**:

- Flexbox Froggy: http://flexboxfroggy.com/
- Flexbox Defense: http://www.flexboxdefense.com/
- Flexbox Adventure https://codingfantasy.com/games/flexboxadventure/

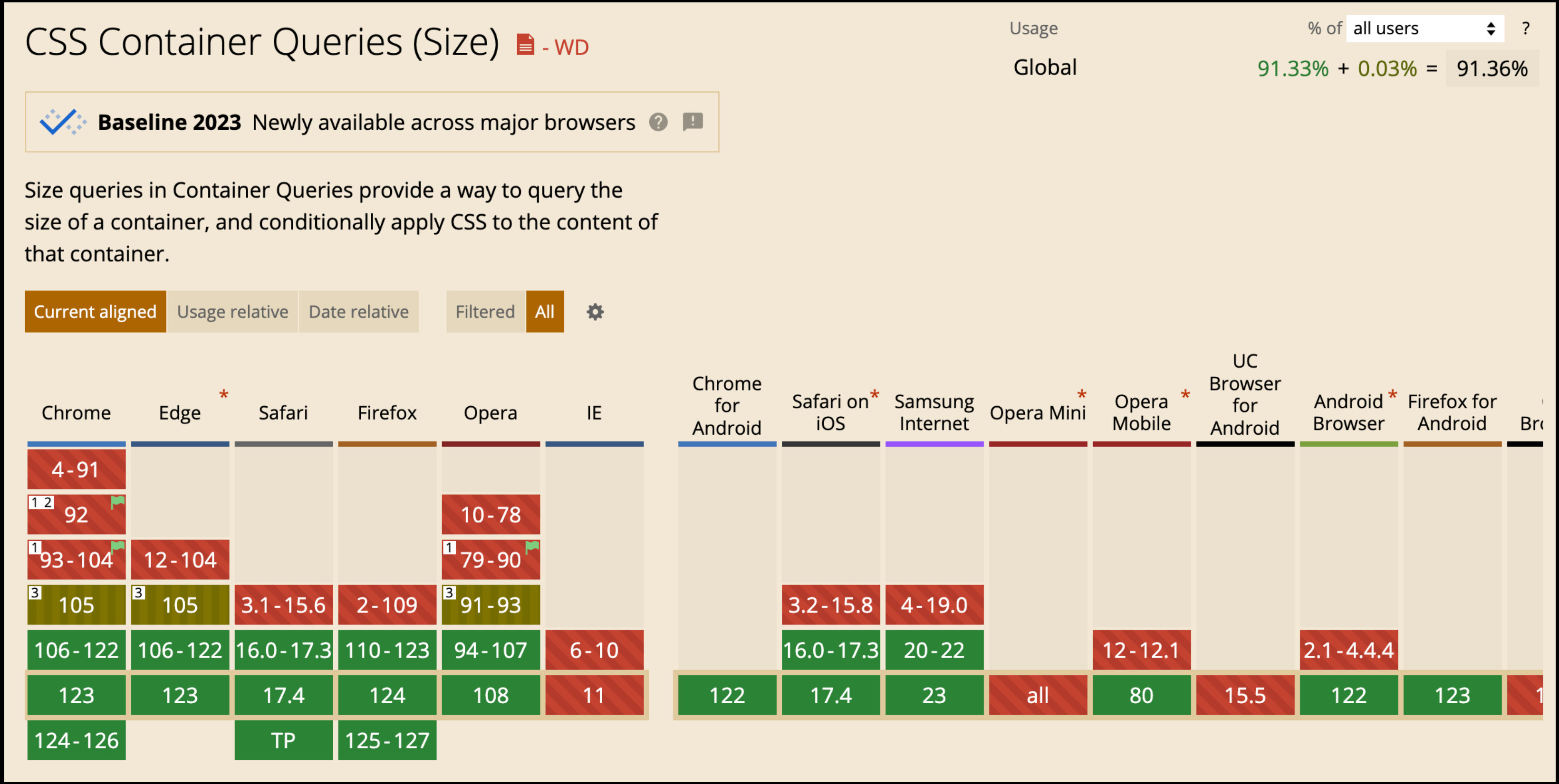
Container Queries

Container queries enable you to apply styles to elements nested within a specific container based on the features of that container.

The query returns true or false depending on whether the query condition is true for the container.

MDN

Container size queries



CSS Container Queries (Size) - WD

Usage: Global — % of all users: 91.33% + 0.03% = 91.36%

Baseline 2023 Newly available across major browsers

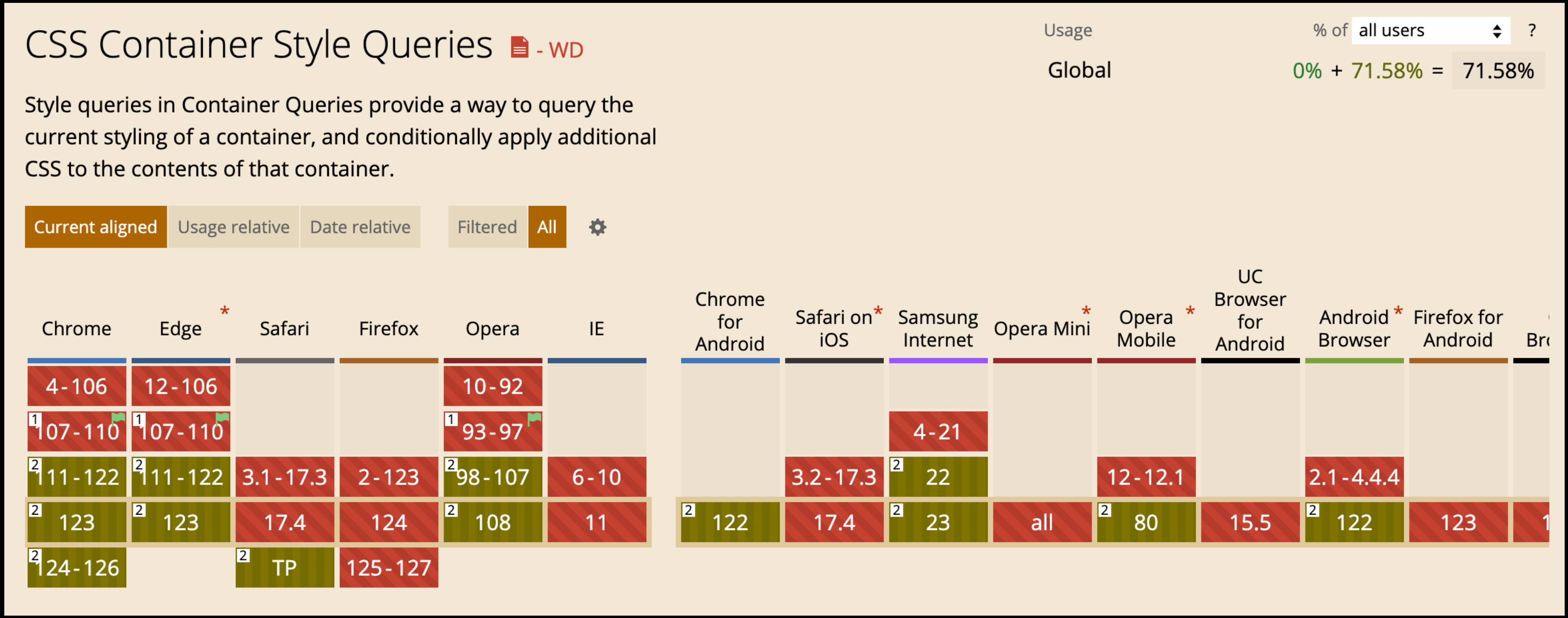
Size queries in Container Queries provide a way to query the size of a container, and conditionally apply CSS to the content of that container.

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	Br
🟥 4-91														
🟥 1 2 92				🟥 10-78										
🟥 1 93-104	🟥 12-104			🟥 1 79-90										
▨ 3 105	▨ 3 105	🟥 3.1-15.6	🟥 2-109	▨ 3 91-93			🟥 3.2-15.8	🟥 4-19.0						
🟩 106-122	🟩 106-122	🟩 16.0-17.3	🟩 110-123	🟩 94-107	🟥 6-10		🟩 16.0-17.3	🟩 20-22		🟥 12-12.1		🟥 2.1-4.4.4		
🟩 123	🟩 123	🟩 17.4	🟩 124	🟩 108	🟥 11	🟩 122	🟩 17.4	🟩 23	🟥 all	🟩 80	🟥 15.5	🟩 122	🟩 123	🟥 1
🟩 124-126		🟩 TP	🟩 125-127											

https://caniuse.com/css-container-queries

Container style queries

Support shown pertains only to CSS custom properties, not other properties



CSS Container Style Queries - WD

Usage — % of all users — ?

Global 0% + 71.58% = 71.58%

Style queries in Container Queries provide a way to query the current styling of a container, and conditionally apply additional CSS to the contents of that container.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	Br
4-106	12-106			10-92										
[1] 107-110	[1] 107-110			[1] 93-97				4-21						
[2] 111-122	[2] 111-122	3.1-17.3	2-123	[2] 98-107	6-10		3.2-17.3	[2] 22		12-12.1		2.1-4.4.4		
[2] 123	[2] 123	17.4	124	[2] 108	11	[2] 122	17.4	[2] 23	all	[2] 80	15.5	[2] 122	123	1
[2] 124-126		[2] TP	125-127											

https://caniuse.com/css-container-queries-style

Container Size Queries

Width media queries consider the viewport width

Container size queries consider the container width

“Containers” are the elements being queried

Rules within affect only the container descendants,
not the container itself

https://www.smashingmagazine.com/2021/05/complete-guide-css-container-queries/

How do you affect the container itself?

Media queries
Container query on the container’s parent

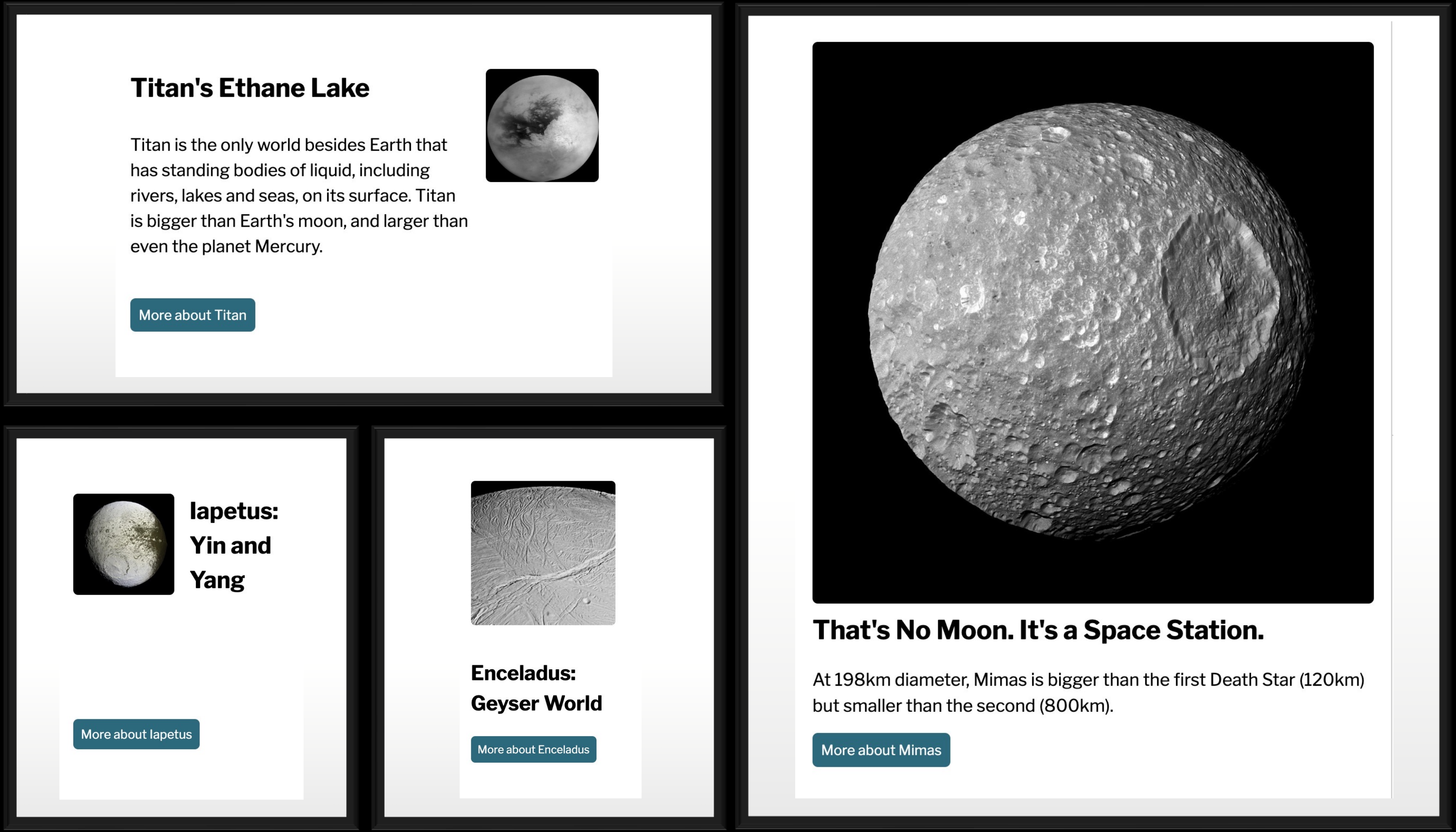
Container size queries are
an addition to responsive design

Not a replacement for media queries

Container query units

- `cqw`: 1% of a query container's width
- `cqh`: 1% of a query container's height
- `cqi`: 1% of a query container's inline size
- `cqb`: 1% of a query container's block size
- `cqmin`: The smaller value of either cqi or cqb
- `cqmax`: The larger value of either cqi or cqb

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_containment/Container_queries#container_query_length_units



Titan's Ethane Lake

Titan is the only world besides Earth that has standing bodies of liquid, including rivers, lakes and seas, on its surface. Titan is bigger than Earth's moon, and larger than even the planet Mercury.

Iapetus: Yin and Yang

Enceladus: Geyser World

That's No Moon. It's a Space Station.

At 198km diameter, Mimas is bigger than the first Death Star (120km) but smaller than the second (800km).