Centering in CSS

- The challenge
  - The Resilience Ringer
    - Legend
- The 5 Contestants
  - 1. Content Center
  - 2. Gentle Flex
  - 3. Autobot
  - 4. Fluffy Center
  - 5. Pop & Plop
- The winner
  - MVP
  - Conclusion
- Community remixes

Centering in CSS is a notorious challenge, fraught with jokes and mockery. 2020 CSS is all grown up and now we can laugh at those jokes honestly, not through clenched teeth.

If you prefer video, here's a YouTube version of this post:

{% YouTube 'ncYzTvEMCyE' %}

The challenge

There are different types of centering. From differing use cases, number of things to center, etc. In order to demonstrate a rationale behind "a winning" centering technique, I created The Resilience Ringer. It's a series of stress tests for each centering strategy to balance within and you to observe their performance. At the end, I reveal the highest scoring technique, as well as a "most valuable." Hopefully you walk away with new centering techniques and solutions.

The Resilience Ringer

The Resilience Ringer is a representation of my beliefs that a centering strategy should be resilient to international layouts, variable sized viewports, text edits and dynamic content. These tenets helped shape the following resilience tests for the centering techniques to endure:

1. **Squished:** centering should be able to handle changes to width
2. **Squashed:** centering should be able to handle changes to height
3. **Duplicate:** centering should be dynamic to number of items
4. **Edit:** centering should be dynamic to length and language of content
5. **Flow:** centering should be document direction and writing mode agnostic

The winning solution should demonstrate its resilience by keeping contents in center while being squished, squashed, duplicated, edited, and swapped to various language modes and directions. Trustworthy and resilient center, a safe center.

Legend

I've provided some visual color hinting to help you keep some meta information in context:

> {% Img src="image/tcFciHGuF3MxnTr1y5ue01OGLBn2/4K35cL1tVpEsGqb4FgKp.png", alt="", width="800", height="438" %}

- A pink border indicates ownership of centering styles
- The grey box is the background on the container which seeks to have centered items
- Each child has a white background color so you can see any effects the centering technique has on child box sizes (if any)

The 5 Contestants

5 centering techniques enter the Resilience Ringer, only one will receive the Resilience Crown □□.

1. Content Center

> Editing and duplicating content with VisBug

1. **Squish**: great!
2. **Squash**: great!
3. **Duplicate**: great!
4. **Edit**: great!
5. **Flow**: great!

It's going to be hard to beat the conciseness of `display: grid` and the `place-content` shorthand. Since it centers and justifies children collectively, it's a solid centering technique for groups of elements meant to be read.

```
.content-center {
  display: grid;
  place-content: center;
  gap: 1ch;
}
```

{% Compare 'better', 'Pros' %} - Content is centered even under constrained space and overflow - Centering edits and maintenance are all in one spot - Gap guarantees equal spacing amongst _n_ children - Grid creates rows by default
{% endCompare %}

{% Compare 'worse', 'Cons' %}

- The widest child (`max-content`) sets the width for all the rest. This will be discussed more in Gentle Flex.

{% endCompare %}

Great for macro layouts containing paragraphs and headlines, prototypes, or generally things that need legible centering.

{% Aside %} `place-content` is not exclusive to `display: grid`. `display: flex` can also benefit from `place-content` and `place-item` shorthands. {% endAside %}

2. Gentle Flex

1. **Squish:** great!
2. **Squash:** great!
3. **Duplicate:** great!
4. **Edit:** great!
5. **Flow:** great!

Gentle Flex is a truer centering-*only* strategy. It's soft and gentle, because unlike `place-content: center`, no children's box sizes are changed during the centering. As gently as possible, all items are stacked, centered, and spaced.

```
.gentle-flex {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 1ch;
```

```
}
```

{% Compare 'better', 'Pros' %} - Only handles alignment, direction, and distribution - Edits and maintenance are all in one spot - Gap guarantees equal spacing amongst _n_ children
{% endCompare %}

{% Compare 'worse', 'Cons' %}

- Most lines of code

{% endCompare %}

Great for both macro and micro layouts.

{% Aside "key-term" %} **Macro layouts** are like states or territories of a country: very high-level, large coverage zones. The zones created by macro layouts tend to contain more layouts. The less surface the layout covers, the less of a macro layout it becomes. As a layout covers less surface area or contains less layouts, it becomes more of a micro layout. {% endAside %}

3. Autobot

1. **Squish:** great
2. **Squash:** great
3. **Duplicate:** fine
4. **Edit:** great
5. **Flow:** great

The container is set to flex with no alignment styles, while the direct children are styled with auto margins. There's something nostalgic and wonderful about `margin: auto` working on all sides of the element.

```
.autobot {
  display: flex;
}
.autobot > * {
  margin: auto;
}
```

{% Compare 'better', 'Pros' %} - Fun trick - Quick and dirty
{% endCompare %}

{% Compare 'worse', 'Cons' %}

- Awkward results when overflowing
- Reliance on distribution instead of gap means layouts can occur with children touching sides
- Being "pushed" into position doesn't seem optimal and can result in a change to the child's box size

{% endCompare %}

Great for centering icons or pseudo-elements.

4. Fluffy Center

1. **Squish:** bad
2. **Squash:** bad
3. **Duplicate:** bad
4. **Edit:** great!
5. **Flow:** great! (so long as you use logical properties)

Contestant "fluffy center" is by far our tastiest sounding contender, and is the only centering technique that's entirely element/child owned. See our solo inner pink border!?

```
.fluffy-center {
  padding: 10ch;
}
```

{% Compare 'better', 'Pros' %} - Protects content - Atomic - Every test is secretly containing this centering strategy - Word space is gap
{% endCompare %}

{% Compare 'worse', 'Cons' %}

- Illusion of not being useful
- There's a clash between the container and the items, naturally since each are being very firm about their sizing

{% endCompare %}

Great for word or phrase-centric centering, tags, pills, buttons, chips, and more.

5. Pop & Plop

1. **Squish:** okay

2. **Squash:** okay
3. **Duplicate:** bad
4. **Edit:** fine
5. **Flow:** fine

This "pops" because the absolute positioning pops the element out of normal flow. The "plop" part of the names comes from when I find it most useful: plopping it on top of other stuff. It's a classic and handy overlay centering technique that's flexible and dynamic to content size. Sometimes you just need to plop UI on top of other UI.

{% Compare 'better', 'Pros' %} - Useful - Reliable - When you need it, it’s invaluable
{% endCompare %}

{% Compare 'worse', 'Cons' %}

- Code with negative percentage values
- Requires `position: relative` to force a containing block
- Early and awkward line breaks
- There can be only 1 per containing block without additional effort

{% endCompare %}

Great for modals, toasts and messages, stacks and depth effects, popovers.

The winner

If I was on an island and could only have 1 centering technique, it would be…

[drum roll]

Gentle Flex □□

```
.gentle-flex {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 1ch;
}
```

You can always find it in my stylesheets because it's useful for both macro and micro layouts. It's an all-around reliable solution with results that match my expectations. Also, because I'm an intrinsic sizing junkie, I tend to graduate into this solution. True, it's a lot to type out, but the

benefits it provides outweighs the extra code.

MVP

Fluffy Center

```
.fluffy-center {
  padding: 2ch;
}
```

Fluffy Center is so micro that it's easy to overlook as a centering technique, but it's a staple of my centering strategies. It's so atomic that sometimes I forget I'm using it.

Conclusion

What types of things break your centering strategies? What other challenges could be added to the resilience ringer? I considered translation and an auto-height switch on the container… what else!?

Now that you know how I did it, how would you?! Let's diversify our approaches and learn all the ways to build on the web. Follow the codelab with this post to create your own centering example, just like the ones in this post. Tweet me your version, and I'll add it to the Community remixes section below.

Community remixes

- CSS Tricks with a blog post

□□□ #2
□ □□□ □□□ 2026-10-31 21:43:23 UTC
□ □□□ □□□ 2026-10-31 21:43:24 UTC