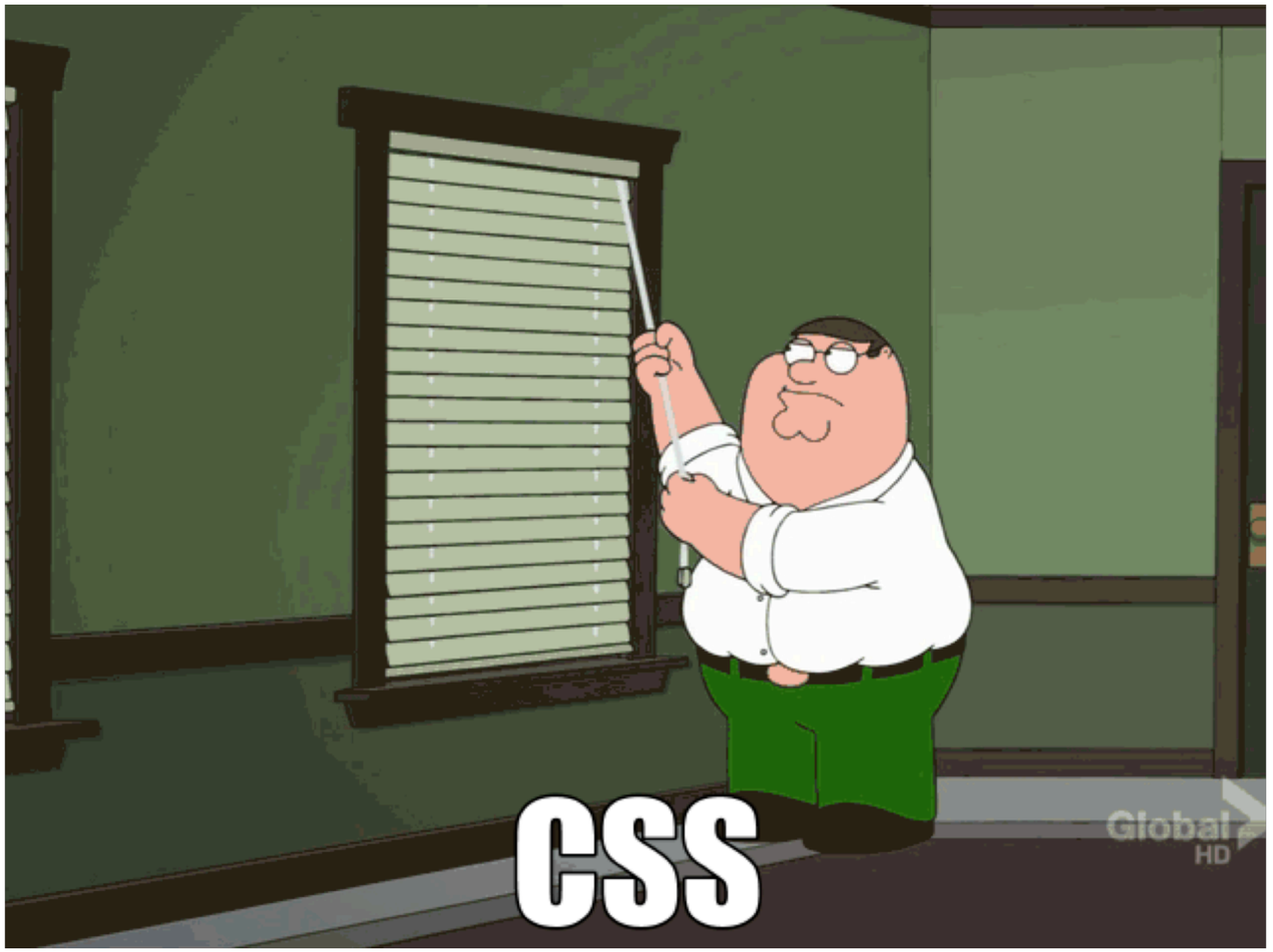
Ready for more?



CSS Layout

The default arrangement for elements is one thing after another top to bottom, according to the html.

Box Dimensions with Width and Height

By default a box will be just big enough to hold its contents, but you can set exact dimensions with the `width` and `height` properties.

```
width: 100px;
width: 50%;
width: 30 rem;
height: 200px;
height: 10rem;
```

Centering

To horizontally center a single block-level element relative to the horizontal size of the window or its parent element, you can let the browser figure it out:

```
margin: 0 auto;
```

For more complicated scenarios and considerations, see CSS-Tricks' excellent Guide to Centering in CSS

The Display Property

This powerful property lets you change the behavior of the rectangular boxes (which is how we think about elements in a layout) on your page.

```
display: inline;
display: block;
display: inline-block; /* inline flow but can have height and width */
display: none; /* hidden from view but still in the dom */
```

`display: flex`

aka

Flexbox

aka

The Flexible Box Layout Module (no one calls it that)

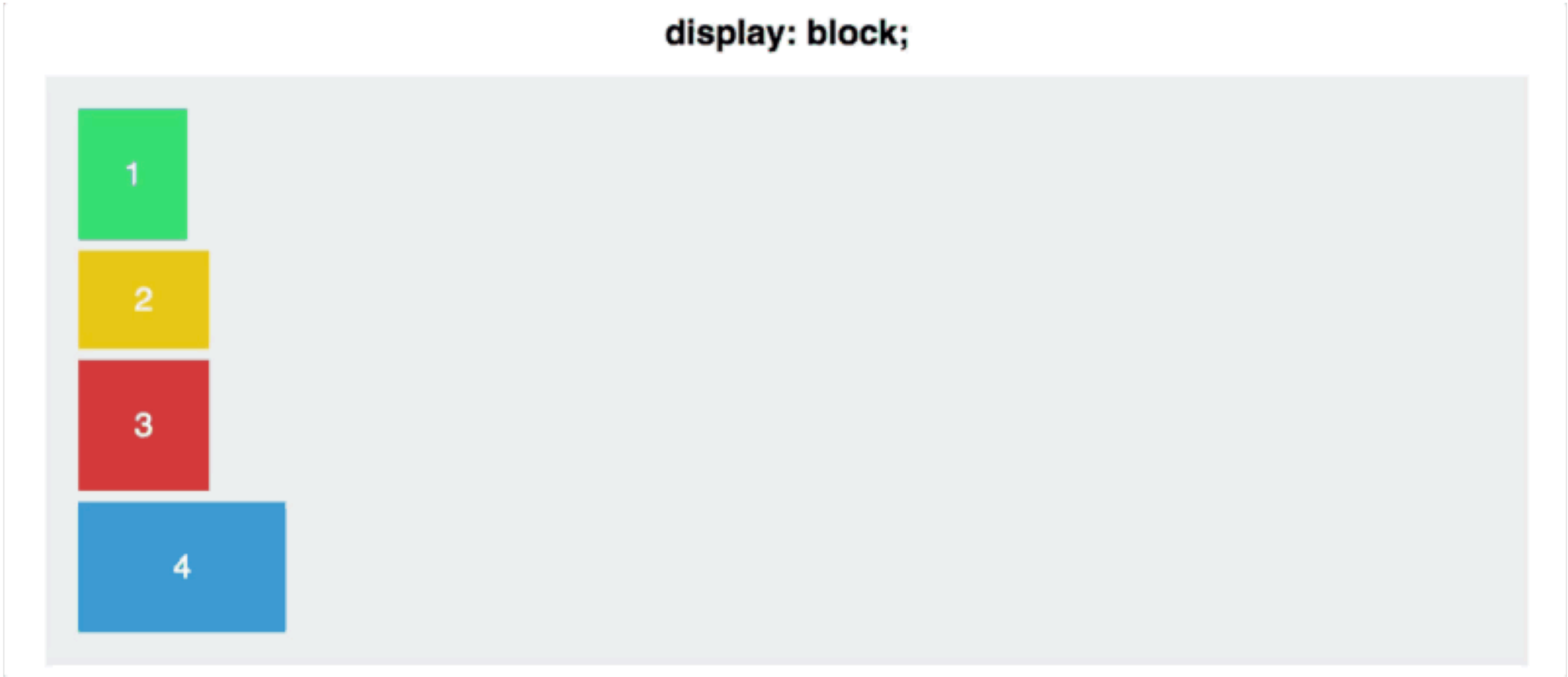
Flexbox

Flexbox is one of three major ways to lay out web pages, along with *floats* and *grid*.

With flexbox, we fill one-dimensional containers with multiple blocks that grow and shrink to fit the current page size.

To use it, you need to use multiple properties in conjunction with each other. It's important to have a conceptual understanding of how it works.

```
.container {
  display: flex;
}
```



Using Flexbox

One trick to using flexbox is to think about the properties that go on a parent element (the "flex container"), and the properties that go on the element (the "flex items").

CSS-Tricks: Complete Guide to Flexbox

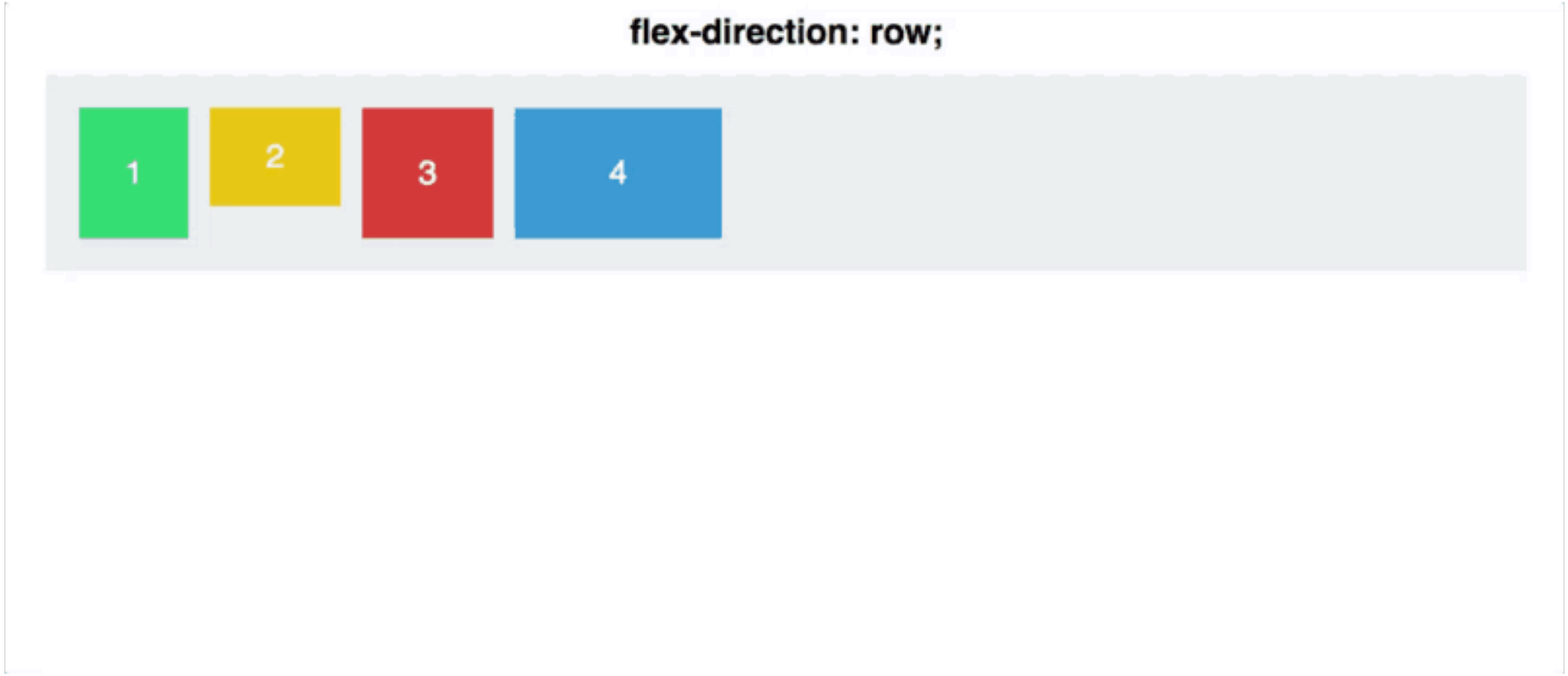
Why is it called "flex"?

The idea of a flexible layout is that it gracefully expands, resizes, or wraps according to the available space. If there is no extra space available, the effects of flexbox won't be obvious or seem to apply.

- You may need to resize your browser window to see if your flexbox properties are having the intended effect.
- You may need to add a height to a flex container if your `flex-direction` is set to `column` so that you create extra space to distribute.

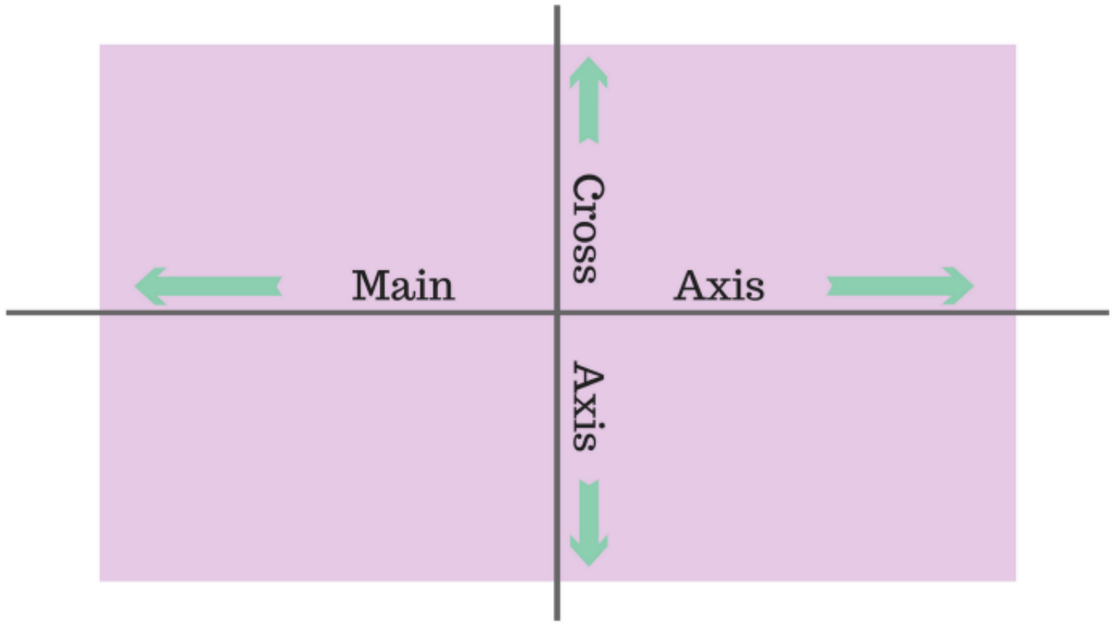
flex-direction

The default flex-direction is `row`, which is horizontal.



The main axis goes in the same direction as the flex-direction

The cross axis is perpendicular to the flex-direction



Values for `flex-direction`

Changing the value of `flex-direction` rotates the main axis.

Items will be laid out either in rows or columns, in order from left to right or right to left, depending on the direction of the main axis.

```
flex-direction: row; /* default; horizontal in order left to right */
flex-direction: row-reverse; /* horizontal in order right to left */
flex-direction: column; /* vertical, in order top to bottom */
flex-direction: column-reverse; /* vertical, in order bottom to top */
```

`flex-wrap`

Flexbox tries to fit all your flex items on one line by default, which may resize your flex items.

But if you want them to wrap to the next line to accommodate their size, you need to set the flex-wrap property.

```
flex-wrap: nowrap; /* default; items will squish on one line */
flex-wrap: wrap; /* nicely wrap them from line to line as needed */
```

visual demo on CodePen

Aligning flex items

`justify-content` and `align-items` depend on the direction of the main axis.

`justify-content` arranges items along the main-axis, in the same direction as flex-direction
`align-items` arranges items along the cross-axis, perpendicular to flex-direction

`align-content`

use with `flex-wrap`

Tells the flex container what to do about the extra space along the cross-axis, similar to how `justify-content` works on the main-axis.

It doesn't do anything if your content takes up all the width along the vertical axis. That is, it is helpful when you have content that wraps

The `flex` shorthand property

- `flex-grow` - how flex-items should take up any extra space. A value of 1 means distribute the available space equally among all items. A value of 0 (the default) means the items do not grow.
- `flex-shrink` (optional) - how small items should get when their container gets smaller
- `flex-basis` (optional) - this is value used as a reference point to calculate the sizes for grow or shrink.

You can just use the shorthand property `flex` and let the browser figure out what the other values should be.

```
.flex-item {
  flex: 1;
}
```

CSS-Tricks Flex shorthand property

Floats

If you're using flexbox, you really should not have to use floats at all. It comes in handy sometimes, though, especially when you want to get text to wrap around images.

Setting the `float` property takes an element out of the normal flow and moves it to the left or right of its containing box. It will "float" into that position until it touches the edge of the box or bumps into another floated element.

```
float: left;
float: right;
float: none;
```

MDN float property

Responsive Layout

Front-end developers need to think about the size of the screen that will display a page.

Responsive design means creating a layout that can adapt to the size of the window that is displaying it.

The viewport meta tag

You need this tag in the `head` section of your page to allow the page to resize.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

MDN viewport meta tag

Techniques to create responsive layouts

1. A flexible, grid-based page structure
2. Flexible images
3. Media queries

The Grid

Designers rely on the intrinsic order of columns and rows to give meaning and shape to page structure.

A grid structure guides the arrangement and spacing of the boxes on the page.

Material Design example of a responsive grid

Making a flexible grid

- Use percentage measurements for elements inside your container
- Include margins
- Grids don't have to be a certain number of columns or even columns of equal size
- For your grid container, consider px or rem for width and % for max-width

Flexible images

```
img {
  max-width: 100%;
  height: auto;
}
```

The max-width property ensures that the image width will not exceed the size of its container.

The height property will maintain the image's aspect ratio.

Media Queries

```
@media screen and (max-width: 1024px) {...}
@media screen and (min-width: 768px) and (max-width: 1023px) {...}
@media print {...} /* if the page is printed by the user */
@media (max-width: 576px) {...} /* applies to all types */
```

Media types to keep in mind are print and screen. Including a media type in a media query is optional.

Breakpoints

Device sizes are going to change all the time, so don't worry about getting the perfect ones. Use whatever makes sense to you.

People do like standards, so here are some, taken from Bootstrap 4.

phones: >= 576px
tablets: >= 768px
desktops: >= 992px
big desktops: >= 1200px

Bootstrap 4 responsive breakpoints

Basic Web Design principles for developers

- typography and whitespace
- principle of least surprise
- aim for clean and simple
- judicious use of color
  - start only with black and white; add color last and with purpose
  - rely on neutrals then choose a base color and an accent color

7 Rules for Creating Gorgeous UIs