CSS3

Designing a better future

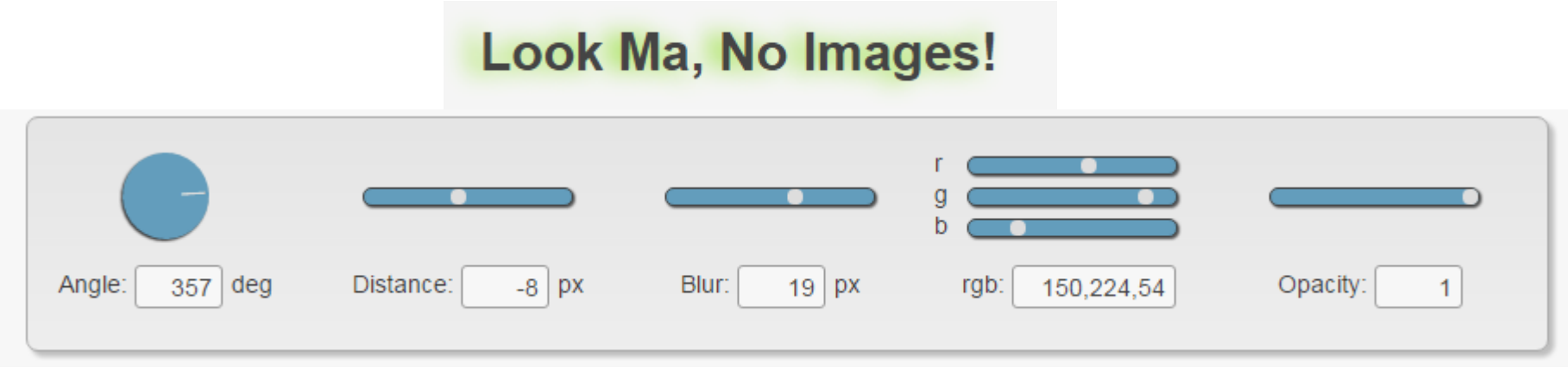
CSS3 is here!

- CSS3 is the latest evolution of the Cascading Style Sheets language and aims at extending CSS2.1.
- It brings a lot of long awaited features, like: rounded corners, shadows, gradients , transitions and animations.

- Also new layouts like multi-columns , flexible box and grid layouts.
  - Not everything is fully supported yet, so use with caution for old browsers.

Text effects

- Few additions:
  - text-shadow
  - Word wrap (wrap long text even in the middle of a word)
  - Text-overflow
    - leave a visual “hint” to the user that text has been clipped (demo)



Web fonts

@font-face is here!

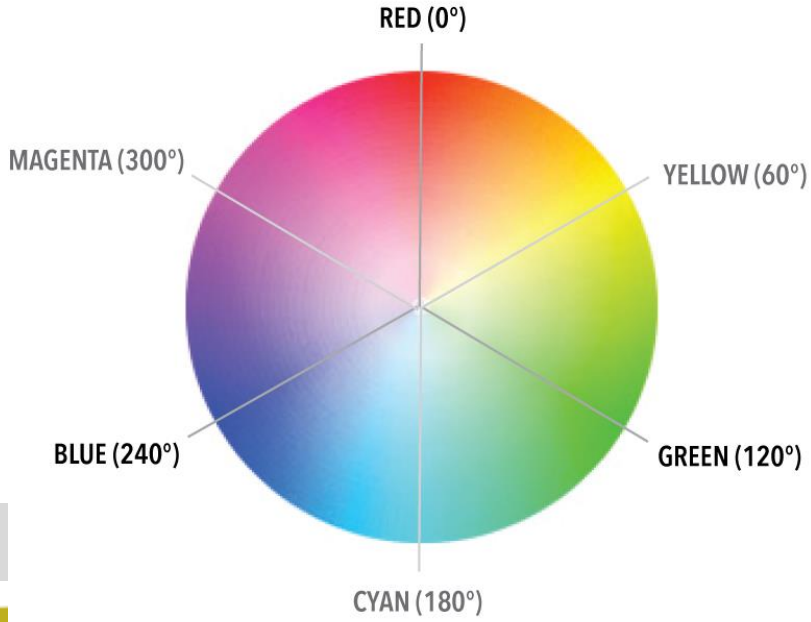
- TTF and OTF are preferred
- In the @font-face rule define a name for the font
- Then point to the font file
  (use only lowercase for the font url, otherwise IE bites)

Edwardian Script Dobkin Lucinda Calligraphy
Amaze Brock Script Freehand591 BT
Kristen ITC Adler Bernard Tango BT
Bradley Hand ITC
Park Avenue Pristina French Script MT
T4C Beaulieux Ghosthly
Harrington
Black Chancery Rockwell Americana BT
Blackadder ITC Patrick
Elphinstone
Celtic Garamond Gothikka Cloister Black BT
Chalk Childish Alpha BoysRGross Curlz MT
Burton's Nightmare Comic Sans MS Times New Roman
Haunt Dracula Psycho Poetry Black Jack Old English

HSL support for Colors

HSL takes three values:

- Hue is a degree on the color wheel;
  0 (or 360) is red, 120 is green, 240 is blue.
  Numbers in between reflect different shades.
- Saturation is a percentage value; 100% is the full color.
- Lightness is also a percentage; 0% is dark (black), 100% is light (white), and 50% is the average.

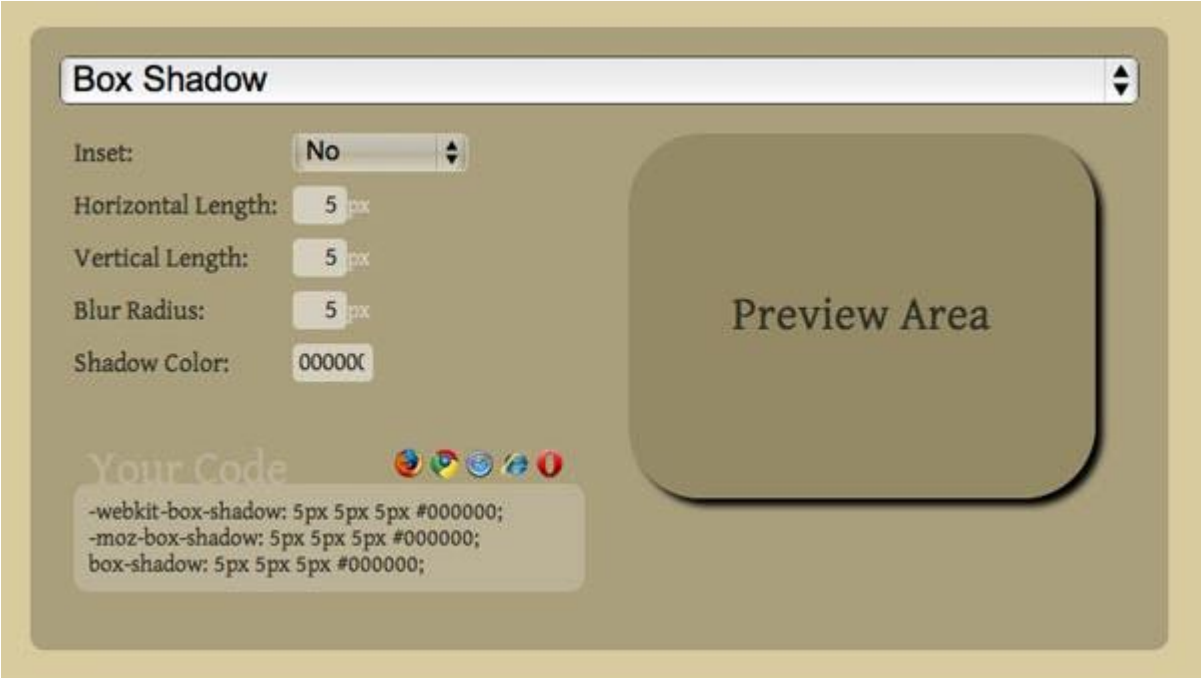


This gives a very wide spectrum of available colors and tones.
There is also HSLA that support opacity

```
background-color: hsl(0,100%, 50%) // RED
```

Borders

- border-radius
- box-shadow
- border-image



Gradients

Smooth transitions between two or more specified colors:

- linear-gradient()
  background: linear-gradient(*direction*, *color-stop1*, *color-stop2, ...*);

- radial-gradient()
  radial-gradient(*center, shape size, start-color, ..., last-color*);

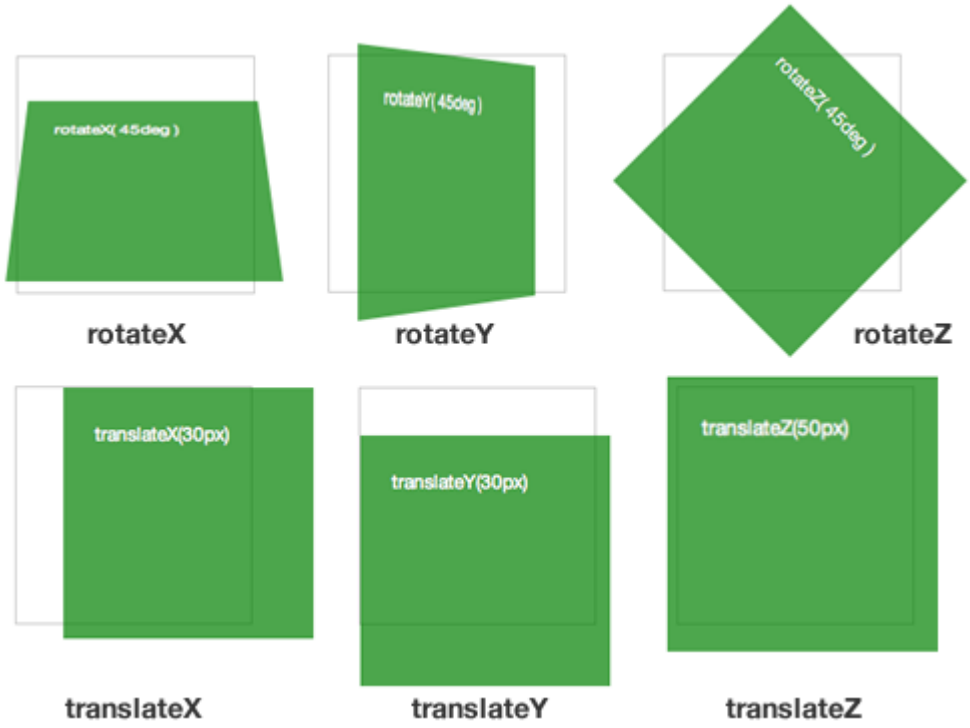
- And also:
  repeating-linear-gradient()
  repeating-radial-gradient()

Background

- Few additions:
  - background-origin: border-box / padding-box / content-box
  - background-clip: border-box / padding-box / content-box
  - background-size: px / percent / cover / contain
- Also, multiple background images are supported

2d transforms

- translate()
- rotate()
- scale()
- skew()
- matrix()



3d transforms

- Translating
  - translate3d(x,y,z), translateX(x), translateY(y), translateZ(z)
- rotating
  - rotate3d(x,y,z,angle), rotateX(angle), rotateY(angle), rotateZ(angle)
- scaling
  - scale3d(x,y,z), scaleX(x), scaleY(y), scaleZ(z)
- perspective
  - perspective(n) - Defines a perspective view for a 3D transformed element

Transitions

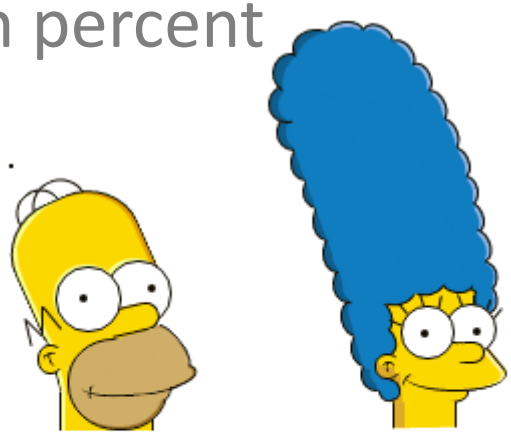
- transitions are effects that let an element gradually change from one style to another
- To do this, you need to
  - Specify the property you want to add an effect to (or all)
  - Specify the duration of the effect.
- Control using the following properties:
  - Transition – shorthand
  - transition-property
  - transition-duration
  - transition-timing-function
  - transition-delay

Columns layout

- 3 new properties, to help us layout in a newsletter style…
- column-count
- column-gap
- column-rule

CSS Animations

- CSS animations are based on the @keyframes rule
- The animation is performed by gradually changing from one set of CSS styles to another.
- During the course of the animation, styles can alter many times.
- Specify when the change will happen in percent



Homer Marge

View CSS View CSS

Backface-visibility

- Hide the backside of a rotated div element
- whether or not an element should be visible when not facing the screen
- Values: hidden / visible

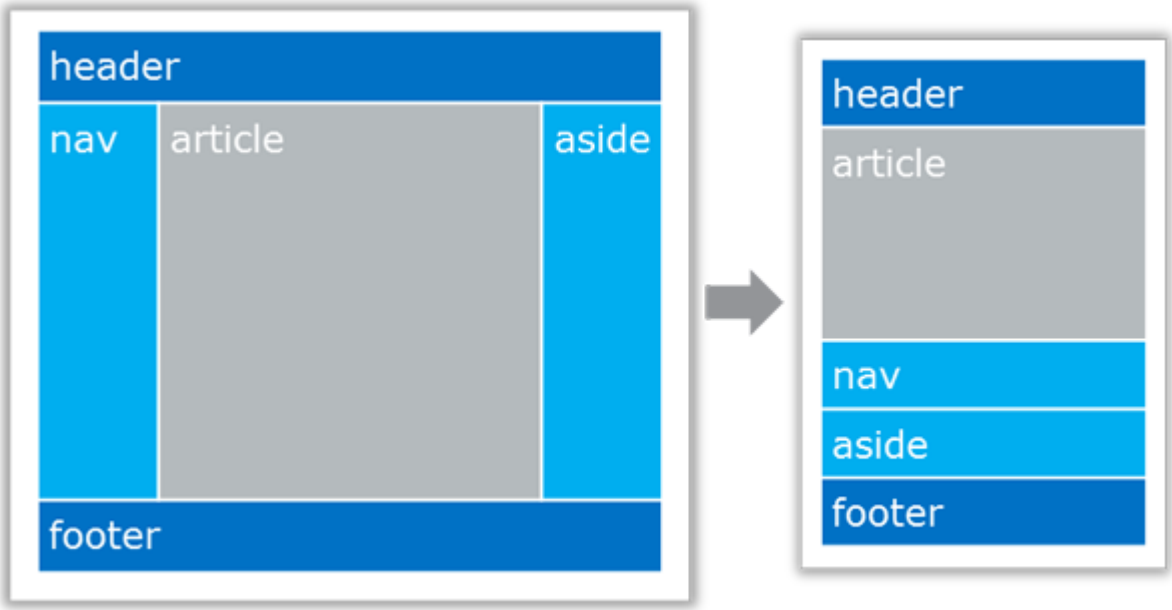
Flip

Substring Matching Attribute Selectors

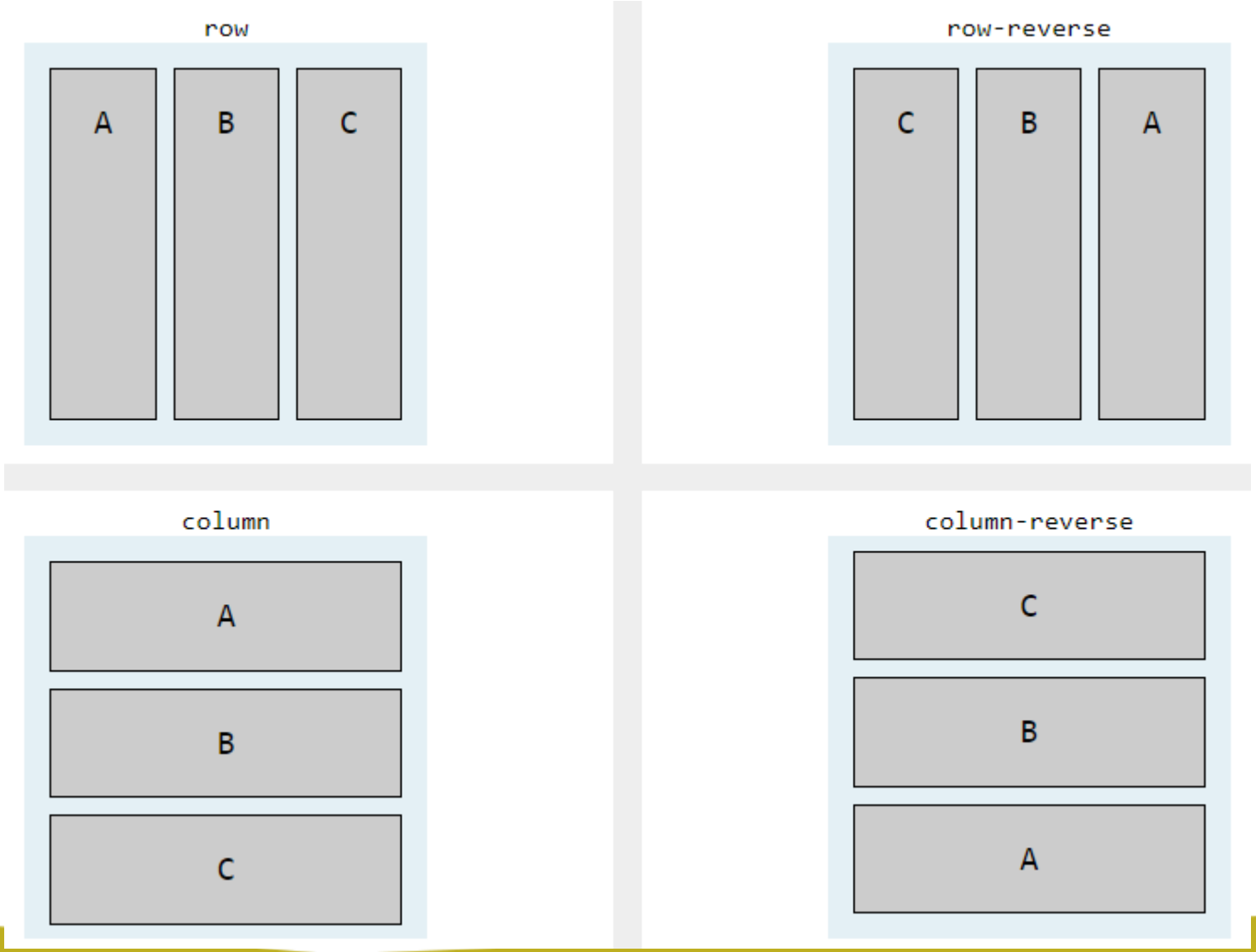
- These new selectors are as follows:
  - [att^=val] – the “begins with” selector
    e.g. - href begins with “mailto:”
  - [att$=val] – the “ends with” selector
  - [att*=val] – the “contains” selector

Flexible boxes

- Note: IE11 and up
- Helps in supporting different screen sizes
  - A flex container expands items to fill available free space, or shrinks them to prevent overflow.



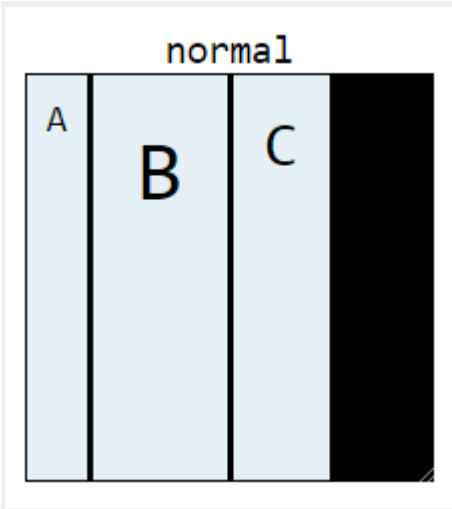
Flex-direction (Sets the Main Axis)

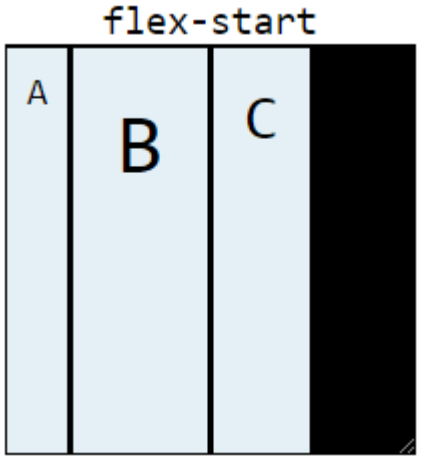


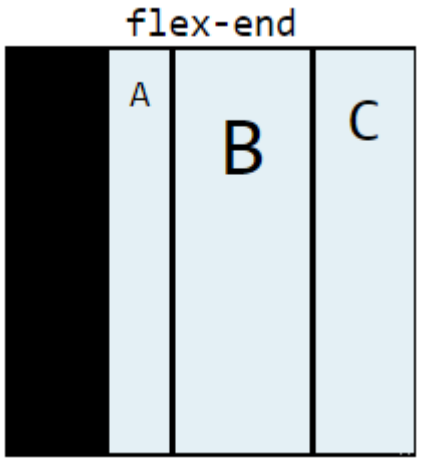
Parent

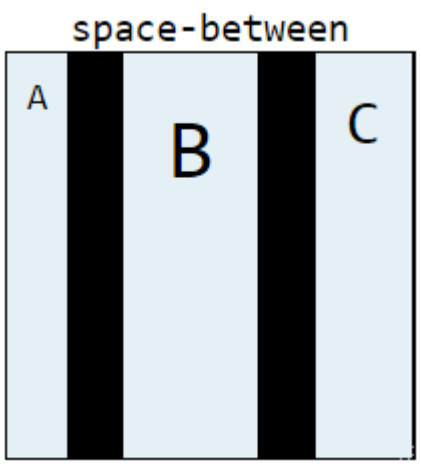
Justify-content (By the Main Axis)

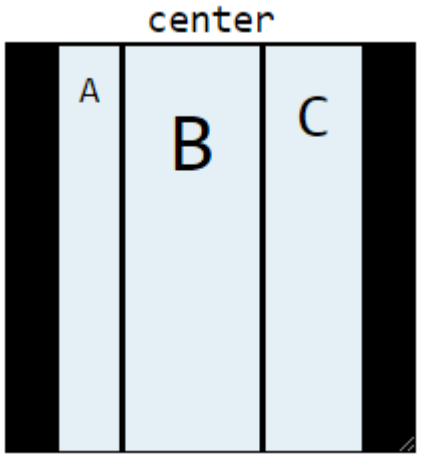
Play Here

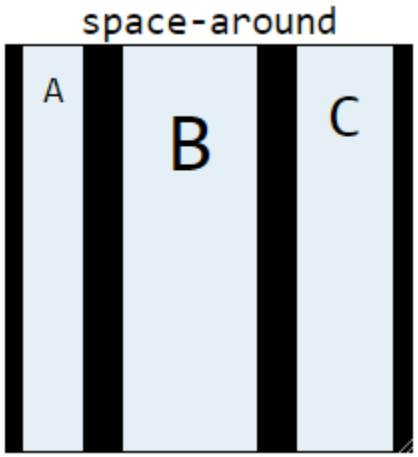










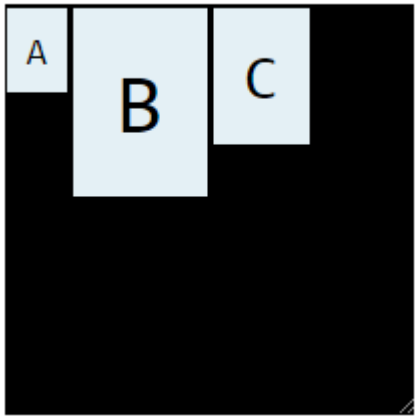


Parent

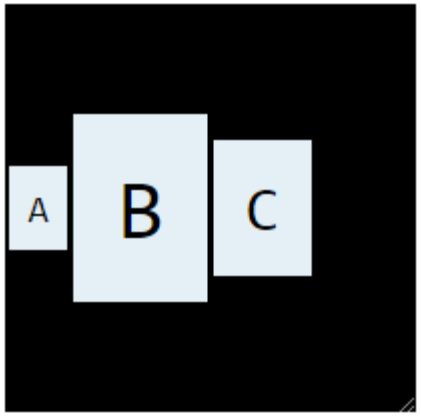
align-items (By the Cross Axis)

Play Here

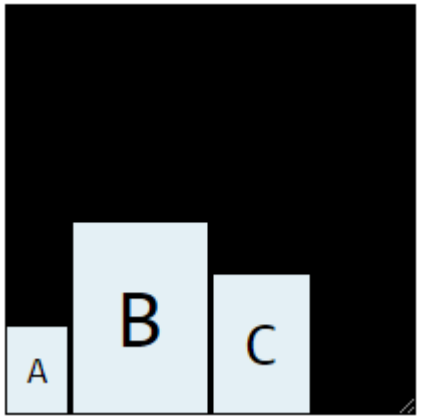
flex-start



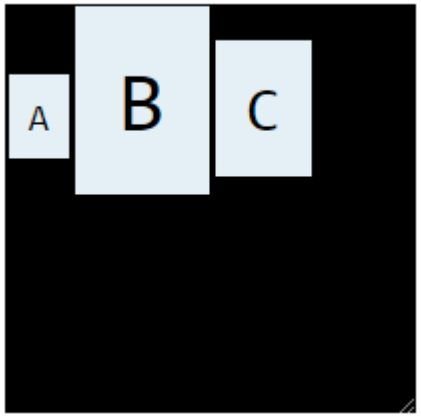
center



flex-end



baseline



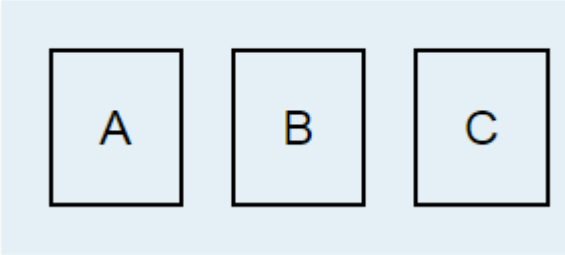
Parent

Flex-wrap

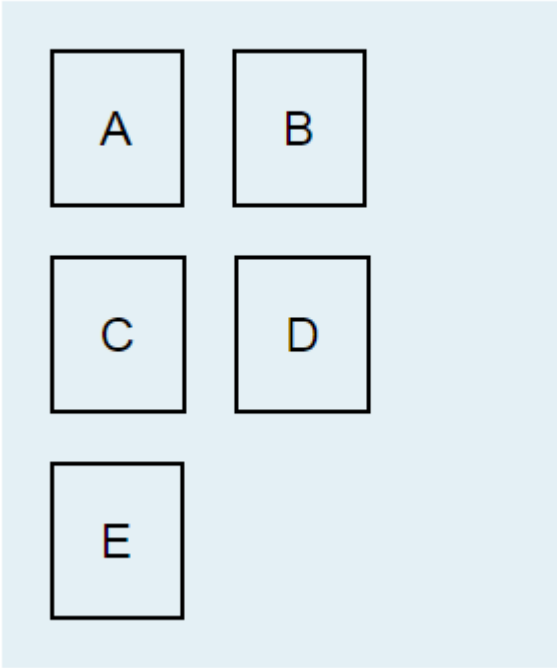
Play Here

specifies whether flex items are forced into a single line or can be wrapped onto multiple lines

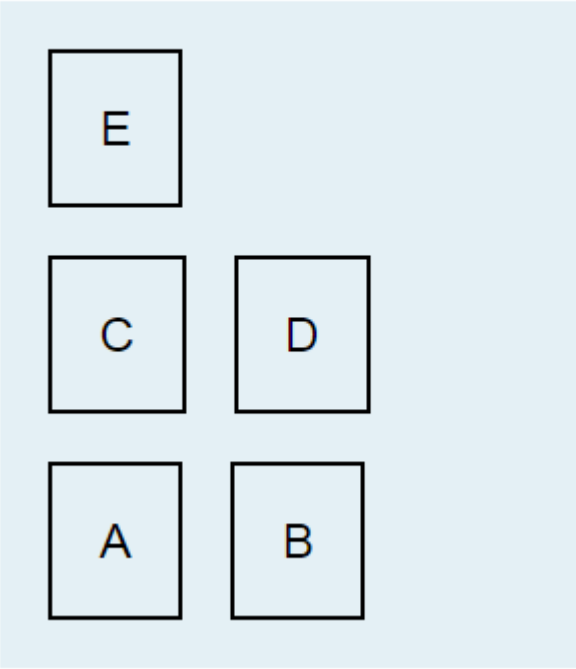
`flex-wrap: nowrap`



`flex-wrap: wrap`

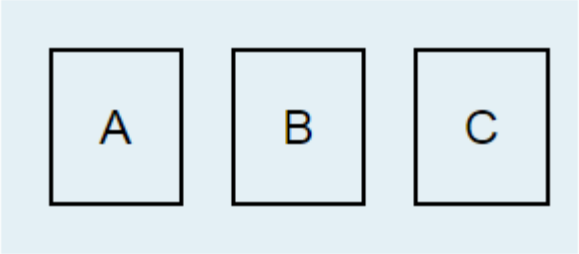


`flex-wrap: wrap-reverse`



flex-flow (shorthand)

The **flex-flow** property is a shorthand property for: flex-direction and flex-wrap.



flex-flow: row nowrap;
flex-flow: column wrap;
flex-flow: column-reverse wrap-reverse;

flex-basis

The flex-basis property specifies the initial main size of a flex item.

This property determines the size of the content-box unless specified otherwise using box-sizing.

```
flex-basis: 5em;
flex-basis: 60px;
flex-basis: auto;
```

flex-grow

The flex-grow property specifies the flex grow factor of a flex item.

It specifies what amount of space inside the flex container the item should take up.

The flex grow factor of a flex item is relative to the size of the other children in the flex-container.

flex-grow:

0.5	1	2

flex-grow: 2;
flex-grow: 0.5;

flex-shrink

The flex-shrink property specifies the flex shrink factor of a flex item.

flex-shrink:

0.5	1	3
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed at purus vitae ipsum hendrerit vulputate quis vitae risus.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed at purus vitae ipsum hendrerit vulputate quis vitae risus.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed at purus vitae ipsum hendrerit vulputate quis vitae risus.

flex-shrink: 3;
flex-shrink: 0.5;

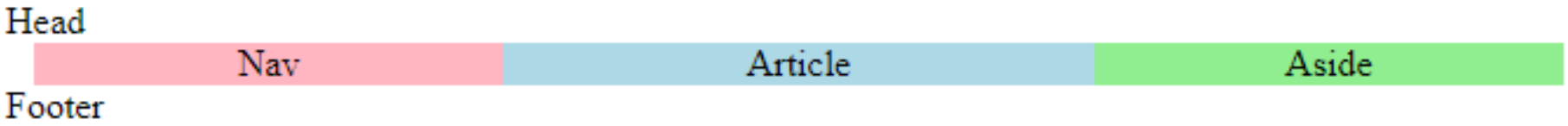
flex (shorthand)

The **flex** property is a shorthand property that sets flex-grow, flex-shrink, and flex-basis.

```
/* One value, unitless number: flex-grow */
flex: 2;
 /* One value, width/height: flex-basis */
flex: 10em;
flex: 30px;
/* Two values: flex-grow | flex-basis */
flex: 1 30px;
 /* Two values: flex-grow | flex-shrink */
flex: 2 2;
 /* Three values: flex-grow | flex-shrink | flex-basis */
flex: 2 2 10%;
```

Order try here

the order property specifies the order used to lay out flex items in their flex container.
(Ascending order)



order: 3;
order: -1;

align-self try here

The align-self property aligns the flex item overriding the align-items value.

The property doesn't apply to block-level boxes, or to table cells.

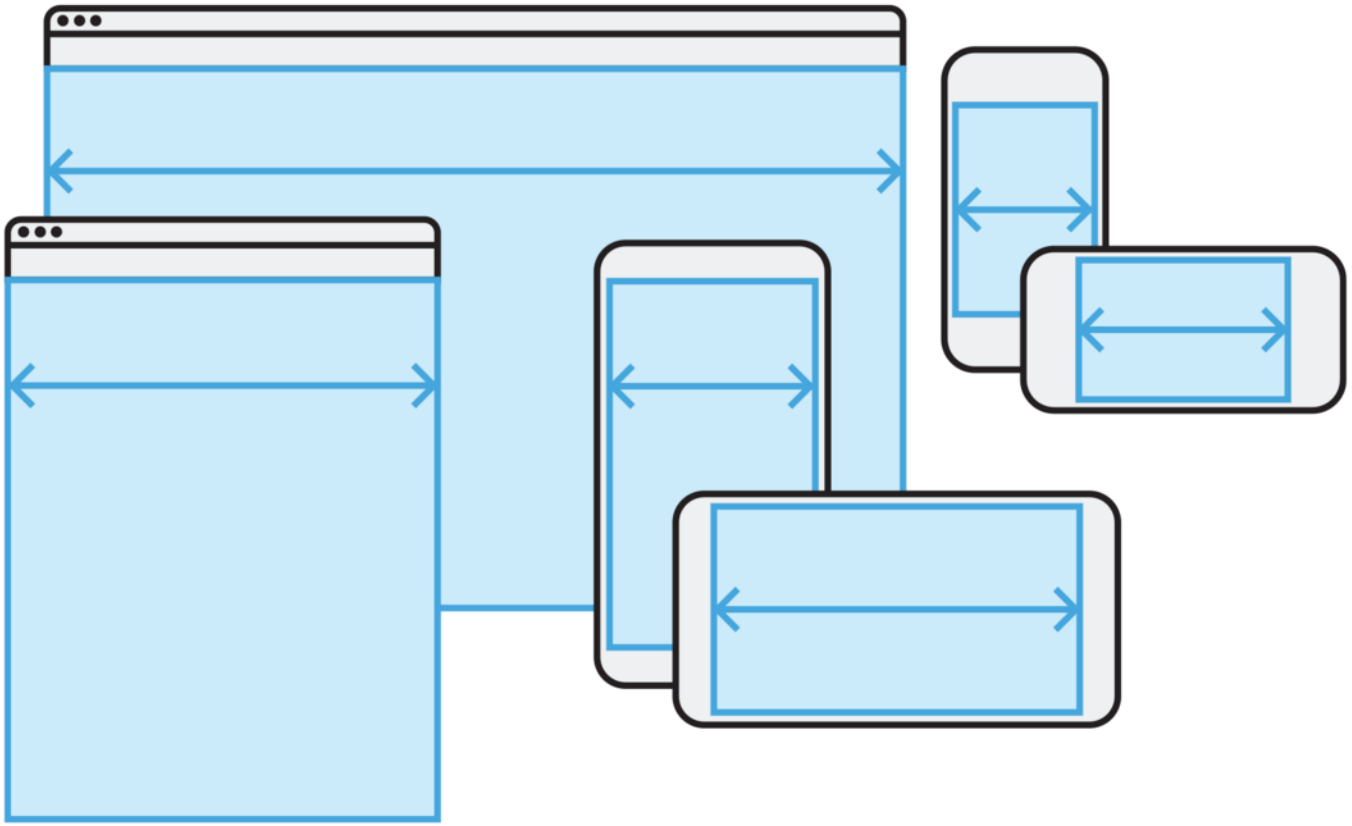
```
align-self: center; /* Put the item around the center */
align-self: flex-start; /* Put the flex item at the start */
align-self: flex-end; /* Put the flex item at the end */
```

Viewport Units

- Viewport units control attributes for elements on the page based on the size of the screen
  - whereas percentages inherit their size from the parent element.
  - i.e: *height: 100%;* applied to an element is relative to the size of its parent.
  - In contrast, *height: 100vh* will be 100% of the viewport height regardless of where the element resides in the DOM.

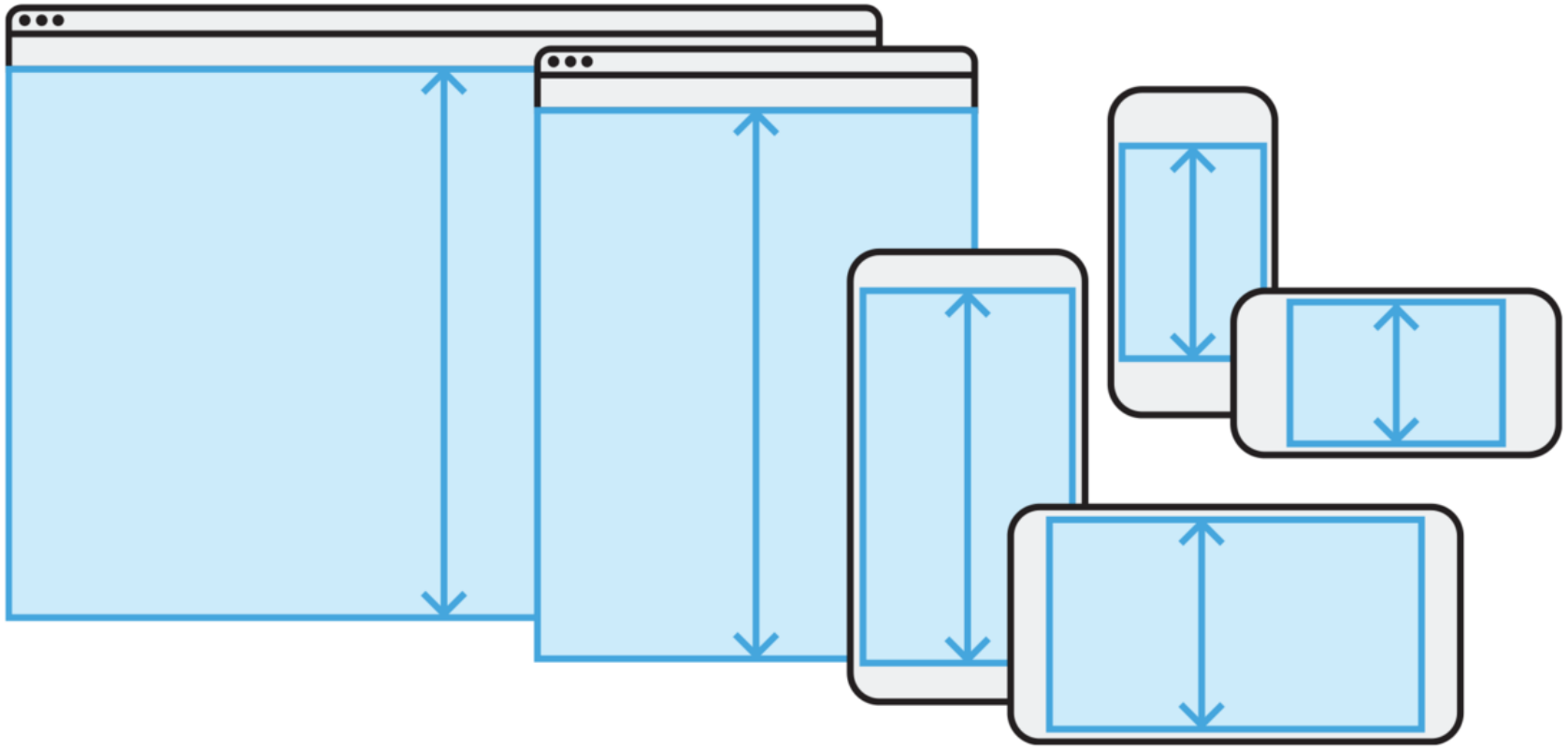
Viewport Units

100vw = 100% of viewport width



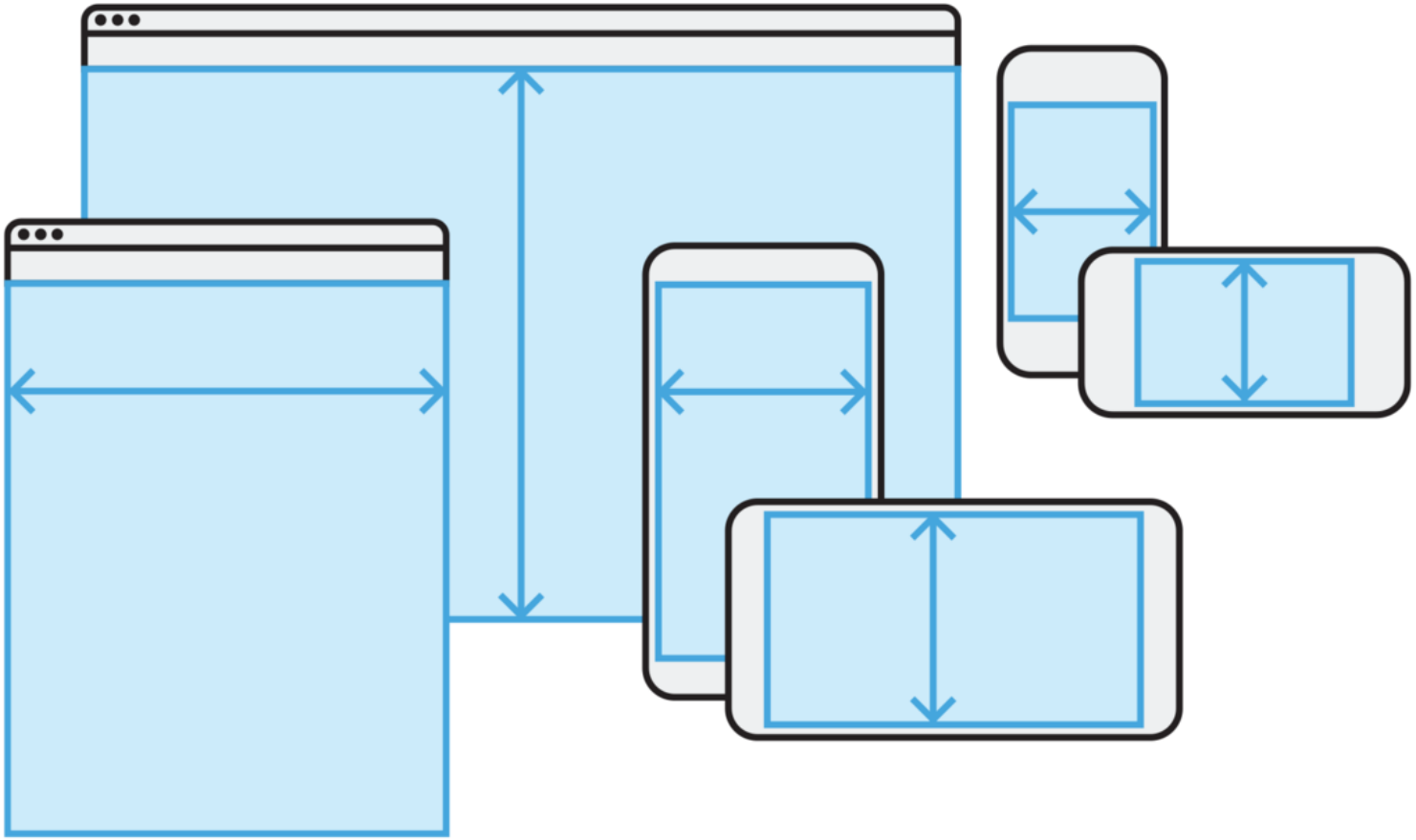
Viewport Units

```
100vh = 100% of viewport height
```



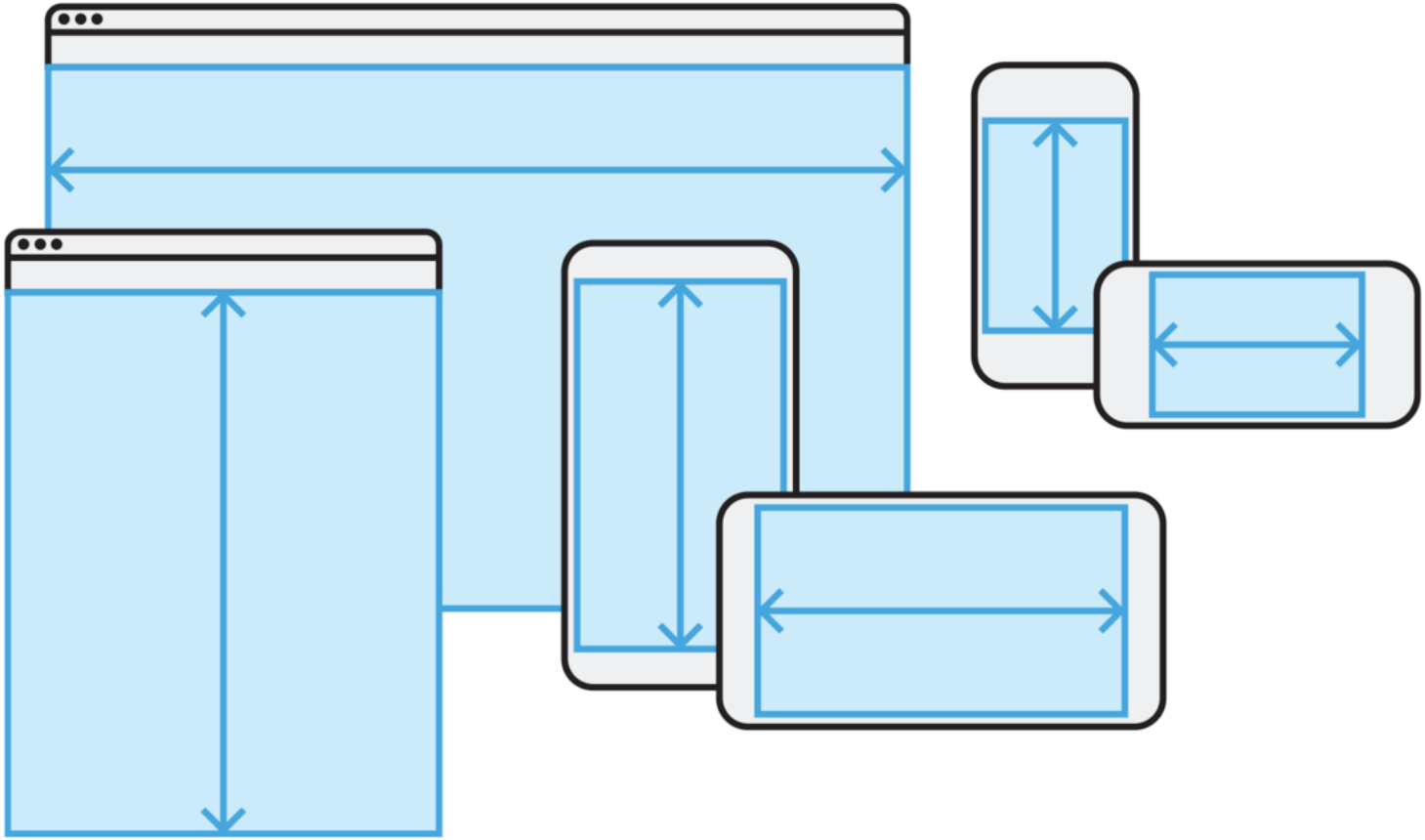
Viewport Units

```
100vmin = 100% of smallest viewport dimension
```

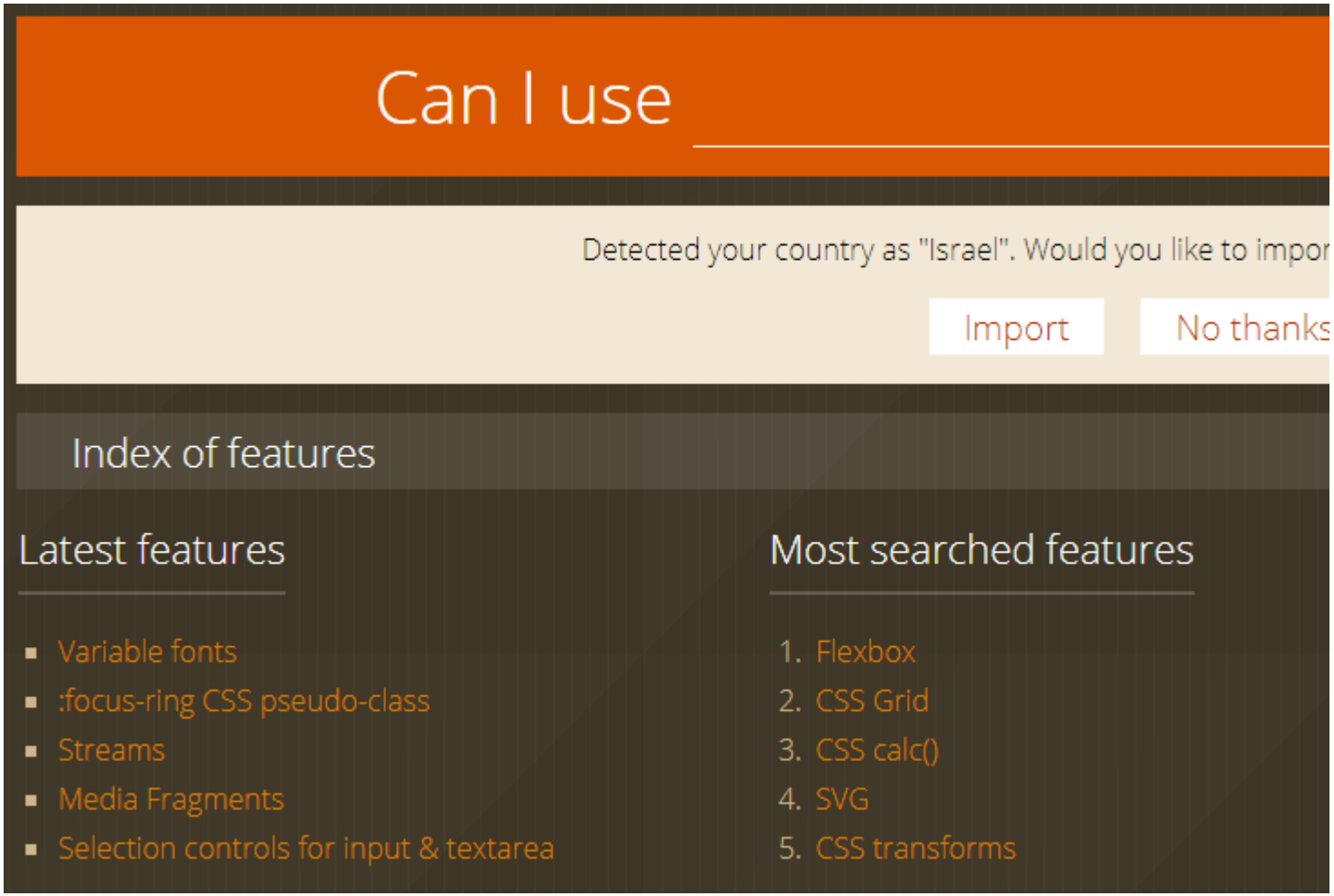


Viewport Units

100vmax = 100% of largest viewport dimension



Can I use?


Can I use
Detected your country as "Israel". Would you like to impor
Import
No thanks
Index of features
Latest features
Variable fonts
:focus-ring CSS pseudo-class
Streams
Media Fragments
Selection controls for input & textarea
Most searched features
1. Flexbox
2. CSS Grid
3. CSS calc()
4. SVG
5. CSS transforms

Victorious!

CSS is Super Power

Now lets build something great