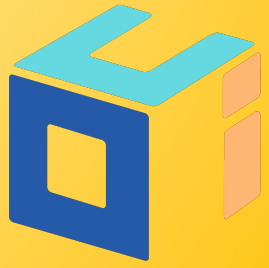
Svelte Animations

slides at https://github.com/mvolkmann/talks

R. Mark Volkmann
Object Computing, Inc.
http://objectcomputing.com
Email: mark@objectcomputing.com
Twitter: @mark_volkmann
GitHub: mvolkmann



OCI | TRAINING

Topics

- Why? How? Kinds? of animations
- Easing functions
- **`svelte/animate`** package
  - **`flip`**
- **`svelte/motion`** package
  - **`spring`** and **`tweened`**
- **`svelte/transition`** package
  - **`blur`**, **`draw`**, **`fade`**, **`fly`**, **`scale`**, **`slide`**, and **`crossfade`**
- Custom transitions
- Transition events

Why and How

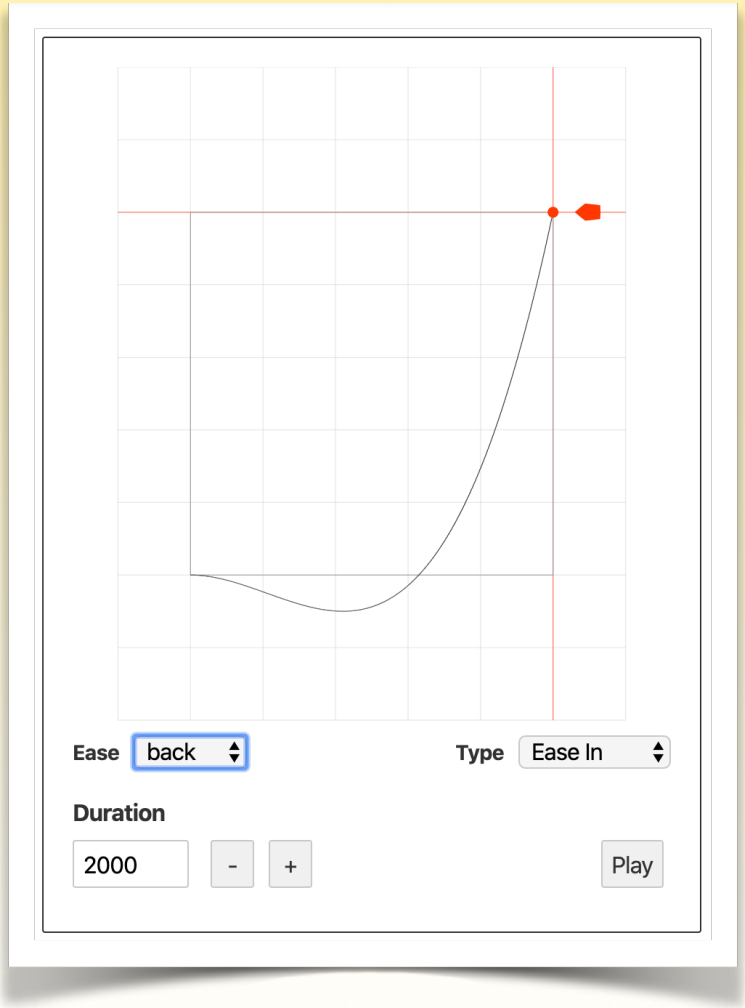
- Why add animations?
  - can make applications more enticing to users
  - can make some operations more intuitive
- How can animations be added?
  - other web frameworks require add-on libraries
  - Svelte provides many **`transition`** directive values and functions that make it easy to add CSS-based animation to elements
  - being CSS-based rather than JavaScript-based means they don't block the main thread, which is good for performance

Two Kinds of Animations

- When an element is added to or removed from the DOM
  - a special effect can occur over a given number of milliseconds
  - for example, added elements can fade in
    and removed elements can slide out of the browser window
  - more visually appealing than abruptly adding or removing an element
- When a value changes
  - can gradually change from its current value to a new value
    over a number of milliseconds
  - when the variable is part of the state of a component,
    the DOM is updated for each intermediate value
  - for example, the value of a given bar in a bar chart can be changed
    from a value of 0 to a value of 10 over a duration of 500 milliseconds
  - rather than the bar jumping from a height of 0 to say 300 pixels,
    it gradually increases, resulting in an animated change

Easing Functions ...

- Animations can proceed at varying rates over their duration
  - specified with easing functions
  - each animation has a default easing function;
    override with **`easing`** option
- Definitions
  - **`svelte/easing`** package currently defines 11 easing functions
  - can also define custom easing functions
  - just functions that take a number between 0 and 1
    and return a number in that same range
- Ease Visualizer
  - browse https://svelte.dev/examples#easing
    to learn about provided easing functions
  - after selecting Ease and Type (Ease In, Ease Out, or Ease In Out),
    it displays a curve that describes its effect and
    animates movement through the duration



... Easing Functions ...

- **`linear`**
  - most basic; provides smooth, constant rate of animation
- **`sine`**, **`quad`**, **`cubic`**, **`quart`**, **`quint`**, **`expo`**, and **`circ`**
  - all are simple curves with only minor differences in acceleration in middle of animation
  - **`expo`** is the most extreme
- **`back`**, **`elastic`**, and **`bounce`**
  - more interesting because they move forward and backward
  - **`back`** changes direction only once and so is the least bouncy
  - **`elastic`** changes direction five times
  - **`bounce`** changes direction seven times

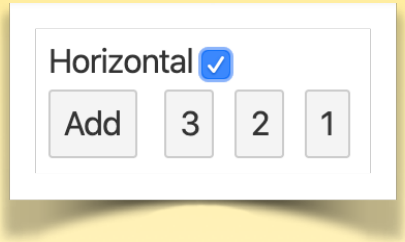
... Easing Functions

- Actual names end with **`In`**, **`Out`**, or **`InOut`**
  - for example, **`bounceIn`**, **`bounceOut`**, and **`bounceInOut`**
  - those ending in **`In`** mostly affect the beginning of a transition
  - those ending in **`Out`** mostly affect the ending of a transition
  - those ending in **`InOut`** affect the beginning and ending of a transition

`svelte/animate` Package

- Provides `flip` function
  - stands for first, last, invert, play
  - doesn't actually flip anything
  - determines new position of element and
    animates changes to its x and y values
    from old to new position
  - common use is to animate changing location of list items

`flip` Example



- Example REPL at https://tinyurl.com/y6awtx5m
  - "Add" button adds a new number to a list of numbers that are each displayed in a button
  - new numbers are added to beginning of list so each must move to make room
  - clicking a number button removes it, causing all the buttons after it to slide toward beginning of list to close up vacated space
  - the list can be toggled between horizontal and vertical, and this change is also animated
  - toggling the Horizontal checkbox before the animation completes cancels the in-progress animations and starts new animations that cause the elements to return to their previous locations

`flip` Animation Options

- **`animate`** directive must be applied to an HTML element
  - applying it to a custom component has no effect
- **`flip`** animation options
  - **`delay`**
    - how long to wait in milliseconds before beginning the animation; defaults to 0
  - **`duration`**
    - how long it should take in milliseconds to complete the animation
    - can also be a function that takes distance to move in pixels and returns duration to use
    - defaults to function `d => Math.sqrt(d) * 120`
  - **`easing`**
    - an easing function that defaults to `cubicOut`
    - many more easing functions can be imported from `svelte/easing`

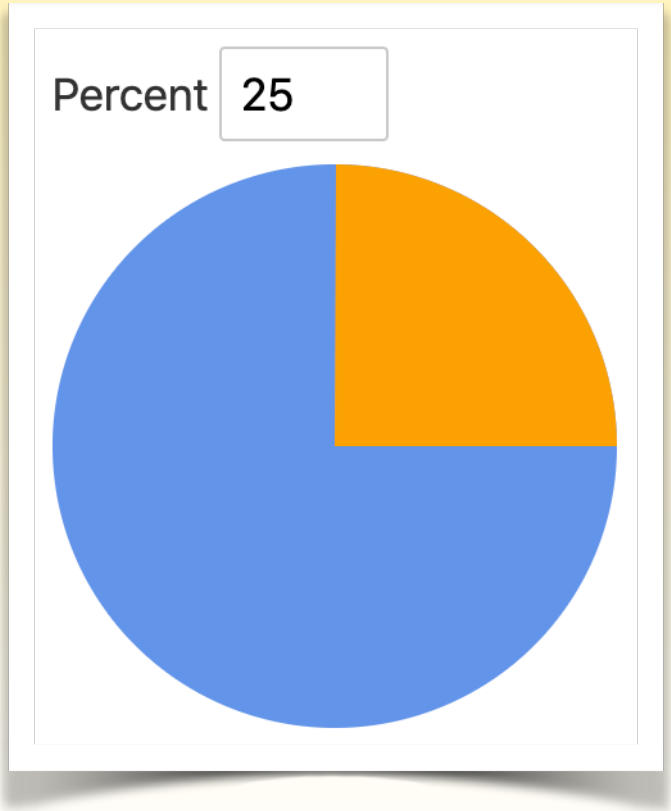
```
// Example of specifying options for flip animation:
<script>
  import {bounceInOut} from 'svelte/easing';
</script>
...
<div animate:flip={{delay: 200, duration: 1000, easing: bounceInOut}}>
```

`svelte/motion` Package

- Provides **`spring`** and **`tweened`** functions that create writable stores whose values animate from old to new values
  - as with all writable stores, the stored value can be changed by calling the store's **`set`** and **`update`** methods
  - **`set`** is passed new value
  - **`update`** is passed a function that computes new value based on current value
- Typically these functions are used to interpolate between two numbers
  - however, they can also be used to interpolate between multiple numbers held in two arrays or between two objects that have the same shape and only have number values in their primitive properties

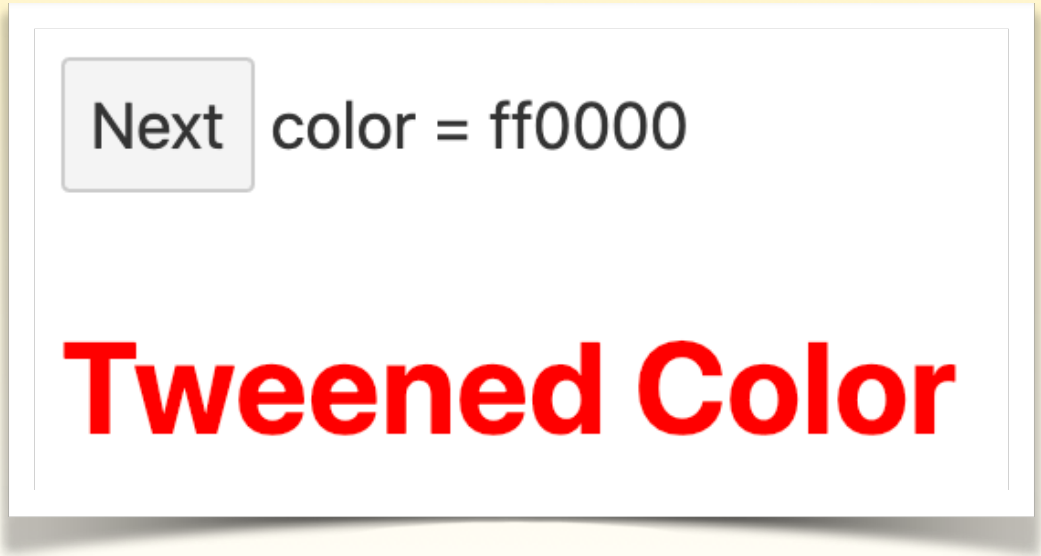
`spring`/`tweened` Example

- Useful for rendering changes in a pie chart
  - when value changes from say 10% to 90%,
    we might want the pie chart to animate showing many values in between
  - rather than changing immediately, the value changes smoothly
- Example REPL at https://tinyurl.com/yyy96pvv
  - SVG-based pie chart component that displays a given percentage value
  - recall from geometry class that
    0 degrees corresponds to 3 o'clock on a clock face,
    and angles increase counterclockwise from there
- **`spring`** is similar to **`tweened`**, but utilizes **`stiffness`**, **`damping`**, and **`precision`** parameters to give a spring-like effect to the animations
  - does not use **`duration`** parameter
  - modify REPL code to switch between **`spring`** and **`tweened`**



`interpolate` Function

- An option accepted by the **`spring`** and **`tweened`** functions is an **`interpolate`** function
  - supports interpolating between values that are **not** numbers, dates, arrays of them, or objects whose properties only have number or date values
  - takes starting and ending values
  - returns another function that takes a number between 0 and 1 and returns a value of the same type as the starting and ending values that is "between" them
- Example REPL at https://tinyurl.com/y456crg2
  - use an **`interpolate`** function to tween over hex color values in the format "rrggbb"
  - when changing from one color to another, such as red to green, we want to pass through colors that are in a sense between them



`svelte/transition` Package

- Provides **`transition`** directive values **`blur`**, **`draw`**, **`fade`**, **`fly`**, **`scale`**, **`slide`**, and the **`crossfade`** function
  - specified using directives **`in`**, **`out`**, and **`transition`**
  - **`in`** effects are applied when an element is added to the DOM
  - **`out`** effects are applied when an element is removed from the DOM
  - **`transition`** effects are applied in both cases
- Like **`animate`**, the **`in`**, **`out`**, and **`transition`** directives must be applied to an HTML element
  - applying them to a custom component has no effect

`fade` Transition

- Animates a change in opacity between 0 and the current opacity value, which is typically 1
- Goes from 0 to the current opacity when an element is added to the DOM, and from the current opacity to 0 when an element is removed from the DOM
- Accepts the options **`delay`** and **`duration`**
  - **`delay`** is number of milliseconds to wait before starting transition
  - **`duration`** is number of milliseconds over which the transition should occur

`blur` Transition

- Animates an amount of blur in pixels
- In addition to **`delay`** and **`duration`** options, it accepts **`easing`**, **`opacity`**, and **`amount`** options
  - **`easing`** is an easing function described earlier
  - **`opacity`** specifies starting opacity value
    - defaults to 0, which is typically the desired value
  - **`amount`** specifies size of blur in pixels and defaults to 5

`slide` Transition

- Like a window shade
- Animates hiding and showing an element
  by gradually changing its height
  - when hiding an element, it is removed from the DOM after height reaches 0
  - elements below it in normal DOM flow move up to occupy vacated space
- Accepts **`delay`**, **`duration`**, and **`easing`** options

`scale` Transition

- Animates the size and opacity of an element
- Accepts **`delay`**, **`duration`**, **`easing`** , **`start`**, and **`opacity`** options
  - **`start`** specifies the smallest scale to use before element is removed
    - defaults to 0, which is typically the desired value

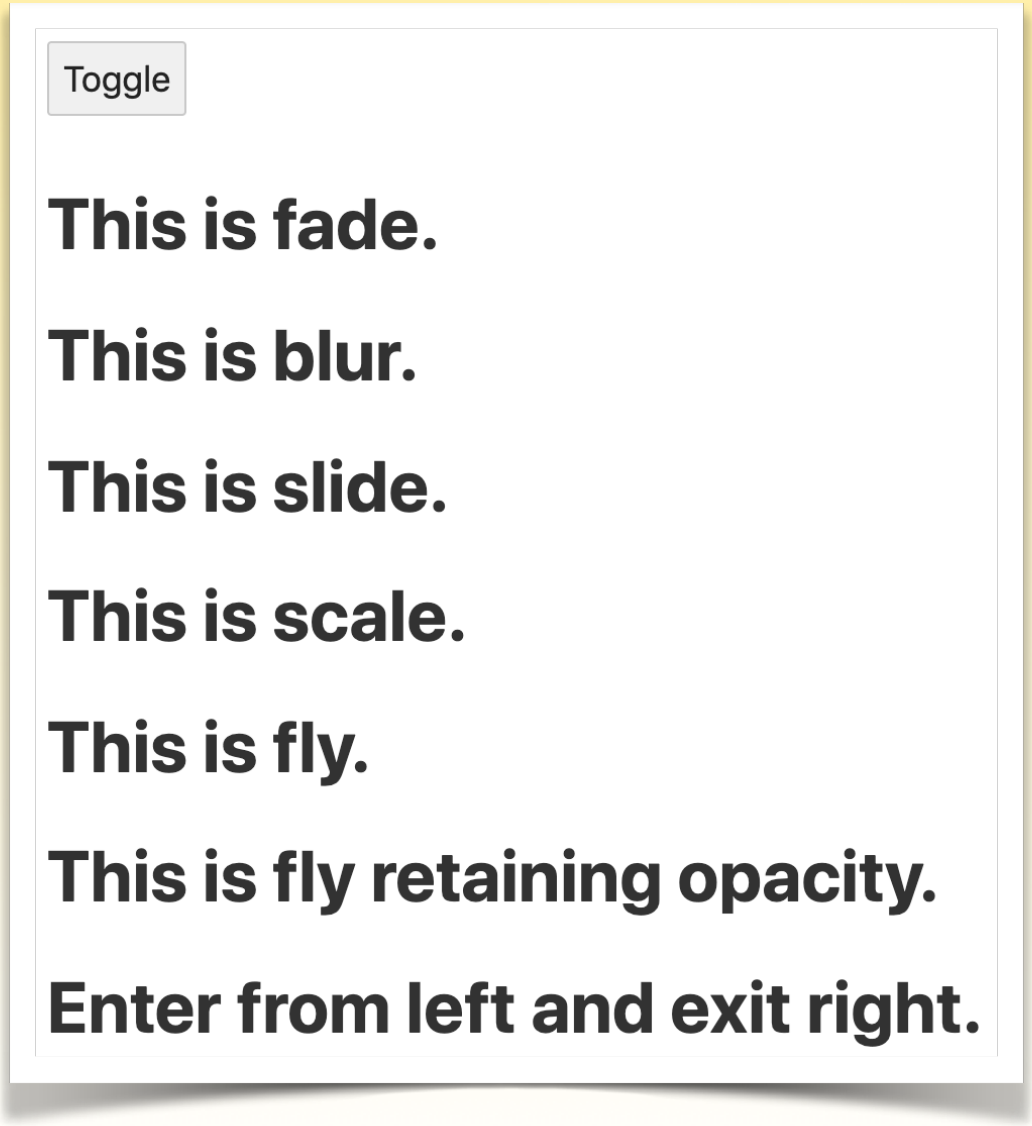
`fly` Transition

- Animates x and y location of an element
- Accepts **`delay`**, **`duration`**, **`easing`** , **`opacity`**, **`x`**, and **`y`** options
  - **`x`** and **`y`** can be set to negative values to slide element off left and top sides of page
- By default, also animates opacity to 0
  - but this can be changed by specifying **`opacity`** option
  - to move an element without changing its opacity during the movement, set **`opacity`** to 1

Transition Examples

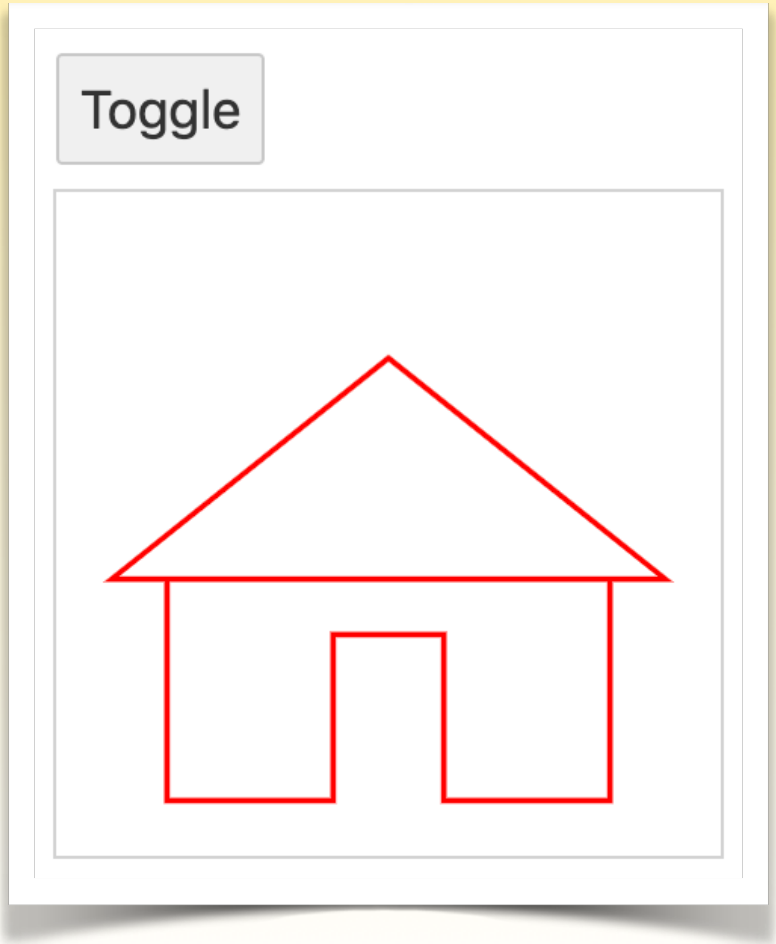
- Demonstrates all but draw
- Focus your eyes on one line at a time to get a clear understanding of the effect of a particular transition

REPL at https://tinyurl.com/y4xprcdn



`draw` Transition

- Animates the stroke of an SVG element
- Accepts **`delay`**, **`duration`**, **`easing`** , and **`speed`** options
  - **`speed`** is an alternate way to specify duration, computed based on SVG path length using **`length / speed`**
- Example REPL at https://tinyurl.com/yxk3x8jm
  - draws a house with a single SVG **`path`** element that uses **`transition:draw`**
  - click Toggle button to cause drawing and erasing of the house to be animated

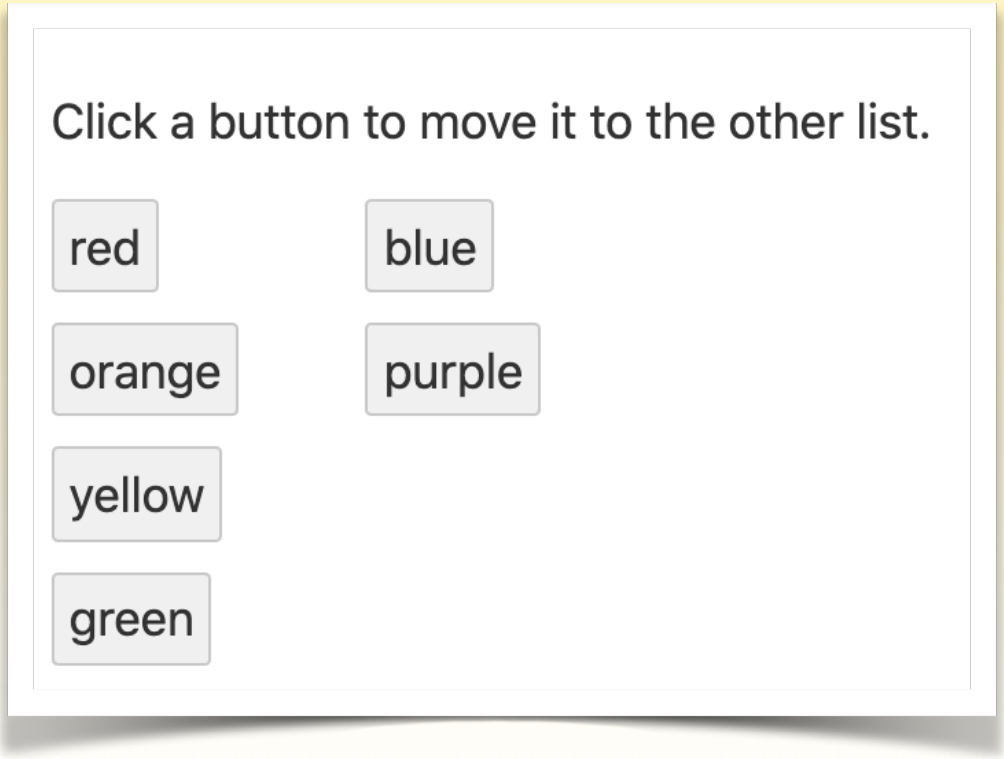


Canceling Transitions

- Canceling a transition means the element returns to its previous state of either being in the DOM or not being in the DOM
- Only transitions specified with the **`transition`** directive can be canceled
- Transitions specified using the **`in`** and **`out`** directives cannot be canceled
- Makes sense because, for example,
  it would be odd to stop the addition an element
  with a **`blur`** transition partway through,
  and to remove the element using a **`fly`** transition

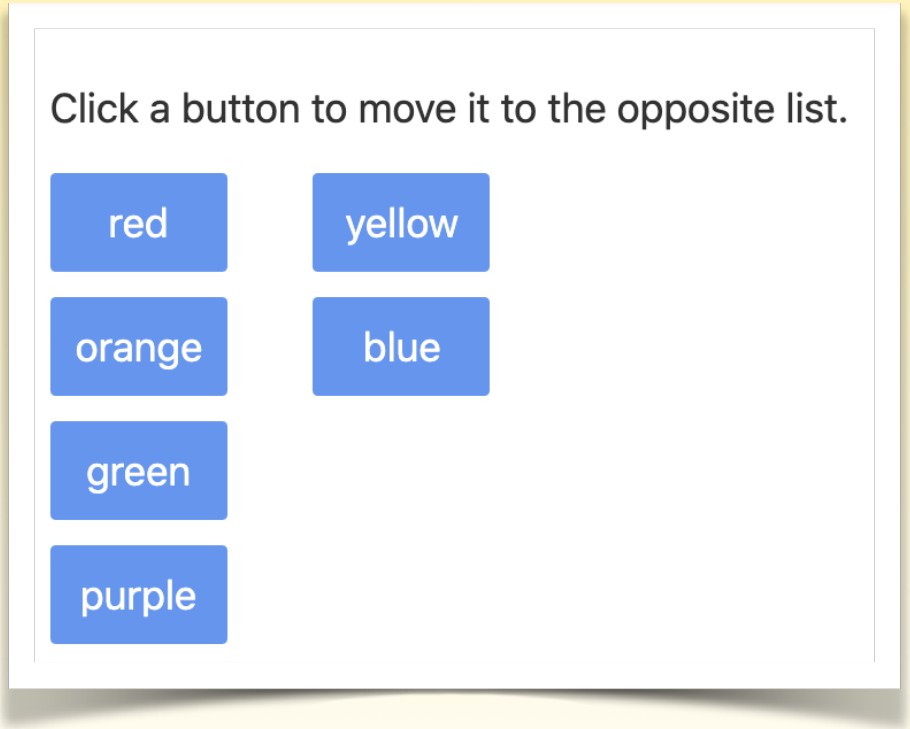
`fade` and `flip` Combined

- Example REPL at https://tinyurl.com/y5bmvj2e
  - moves buttons between left and right lists as each button is clicked
  - uses **`fade`** transition, so the clicked button fades out of its current list and fades into its new list
  - also uses **`flip`** animation so buttons below the button being moved slide up to fill the vacated space
  - implemented using two components, **`App`** and **`ButtonList`** to avoid duplicating code
  - **`ButtonList`** component is used for both the left and right lists.



`crossfade` Transition

- Creates **`send`** and **`receive`** transitions, used to coordinate movement of an element from one parent to another
  - also referred to as a deferred transition
- Example REPL at https://tinyurl.com/y6bcspwm
  - moves items between left and right lists
  - an item is sent out of one list and received into another
  - **`send`** transition defers to see if the element is being "received" in the other location
  - it then animates a transition of the element from its current location to its new location
  - provides a much nicer visual effect than the previous example using **`fade`**
  - also uses **`flip`** animation so remaining list items animate closing up vacated space

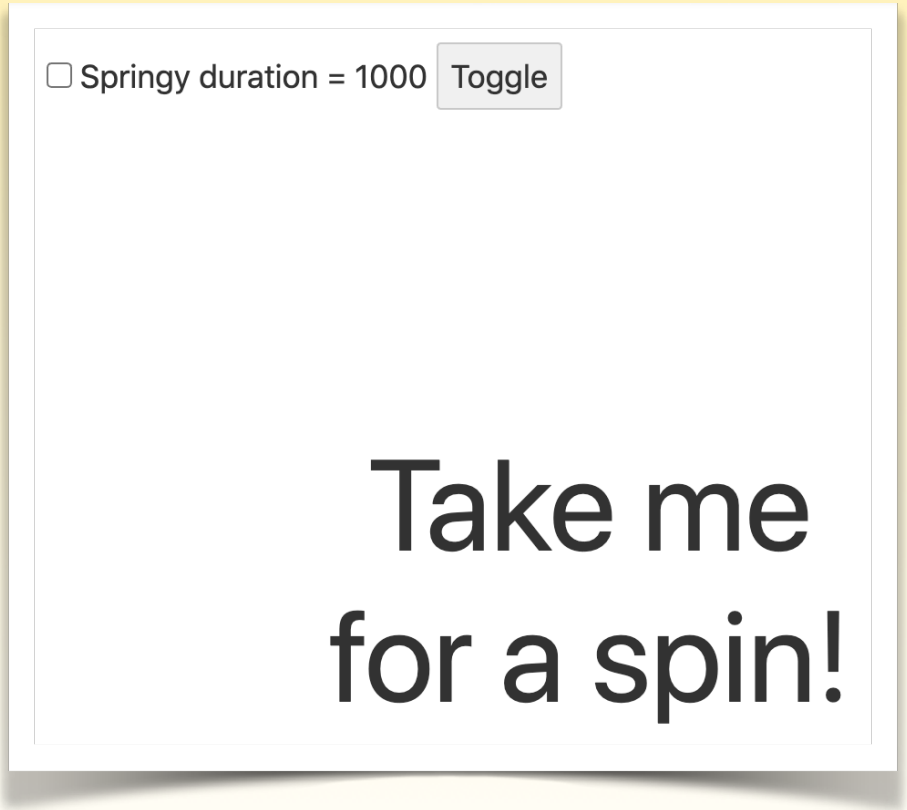


Custom Transitions ...

- To implement a custom transition, write a function that follows a few basic rules
  - the function should take two arguments,
    the DOM node to be transitioned and an options object
  - examples of options include **`delay`**, **`duration`**, and **`easing`** that we saw earlier
  - options specific the custom transition can also be supported
  - the function must return an object whose properties include
    the transition options and a **`css`** method
  - the **`css`** method must return the appropriate CSS string for the number
    between 0 and 1 that is returned by calling the **`easing`** function
  - Svelte takes care of honoring the **`delay`** and **`duration`** options
  - transition options returned can be given default values
    that are used when options are not passed to the custom function
  - the **`css`** method is passed a time value between 0 and 1 and
    must return a string containing CSS properties to be applied to the DOM node for that time value
  - examples of CSS properties that might vary over time include
    `opacity`, **`size`**, **`font-size`**, **`position`**, **`rotation`**, and **`color`**

… Custom Transitions

- Example REPL at https://tinyurl.com/y4kkd22b
  - animate the scale and rotation of an element to make it appear to spiral down a drain when removed from the DOM
  - apply this to a **`div`** element containing the text "Take me for a spin!" sized to wrap to two lines
  - click Toggle button to toggle between hiding and showing the **`div`**
  - check the "Springy" checkbox to use the **`backInOut`** easing function instead of the **`linear`** easing function



Transition Events

- Events are dispatched at specific points in a transition
  - **`introstart`** when an "in" transition begins
  - **`introend`** when an "in" transition ends
  - **`outrostart`** when an "out" transition begins
  - **`outroend`** when an "out" transition ends
- Like with other events, the **`on`** directive can be used to register a function to be called when these events are dispatched
  - for example, to run a function when the transition for an element being removed from the DOM completes, use **`on:outroend={someFunction}`**
  - these functions can trigger changes to the affected component or to other components
  - for example, focus can be moved to a particular **`input`** element after the transition for an element being added to the DOM completes using **`on:introend`**

Conclusion

- Svelte has great built-in support for CSS-based, rather than JavaScript-based, animation, which means animation won't block the main thread
- Easing functions
  - control the rate of change in an animation over its duration
  - many are provided, and custom easing functions can be defined
- Animation is supported by the provided packages
  - `svelte/easing`
  - `svelte/animate`
  - `svelte/motion`
  - `svelte/transition`
- Custom transitions can be easily implemented
- Check out my book, "Svelte and Sapper in Action"
  - https://www.manning.com/books/svelte-and-sapper-in-action