P.PORTO

POLITÉCNICO
DO PORTO
ESMAD

ANIMAÇÃO GRÁFICA
TSIW

Syllabus

- CSS Transitions
- CSS Animations
- Timing functions
- Dealing with events

SVG animations

- **SMIL** (*Synchronized Multimedia Integration Language*): XML language created to define multimedia elements
  - It is, however, being slowly deprecated

```
<svg width="800" height="500" >
        <circle cx="200" cy="200" r="100" stroke="black" stroke-width="5" fill="red">
                <animate attributeName="cx" from="200" to="400"
                begin="0s" dur="5s" repeatCount="indefinite" />
        </circle>
</svg>
```

https://codepen.io/teresaterroso/pen/mdyjdWp

SVG animations

- **Web Animations API**: new standard, to provide access to the animation engine of the browsers, allowing more complex and fluid animations
  - It aims to bring the power of CSS performance, add the benefits and flexibility of JavaScript
  - Still in a draft stage

```
<div id='photo'>
   <img src= 'photo.jpg' />
</div>

<script>
        document.getElementById("photo").animate( [
                { transform: 'translate3D(0%, 0%, 0)', backgroundColor: '#000'  },
                { transform: 'translate3D(10%, 10%, 0)', backgroundColor: '#ff0000' }
        ], { duration: 3000, iterations: Infinity});
</script>
```

https://codepen.io/teresaterroso/pen/PowBoOa

SVG animations

- **CSS Animation**: allows for animations described in CSS language, to animate CSS properties

- **Externals libraries**: a number of JavaScript libraries exist that offer a variety of animation methods
  - While outside the scope of this course, a more curious reader can be pointed to, for example, the GreenSock Animation API

CSS animation

- Provides animation to almost all HTML elements, without the need for JavaScript
  - As it would be expected, SVG falls on the category of a HTML element, and thus it can be animated using CSS animation
- CSS animation describes **how the CSS properties** (e.g. `fill`, `stroke`, `background-position`, `transform`, …) change over time
- The animation can either be triggered by a state transition (e.g. the user hovers an element), or it can be an explicit property of an element

CSS transitions

- The CSS specification allows the definition of CSS styles that are used on specific conditions
- Theses conditions are defined by pseudo-classes, and specify a special state of an element
- Examples of pseudo-classes (and thus states) are:

  `:hover` - selects links on mouse over

  `:active` - selects the active link

  `:focus` - selects the <input> element that has focus

- CSS syntax for pseudo-classes:

```
selector.pseudo-class {
        property : value ;
}
```

- For a complete list of pseudo-classes, visit this link

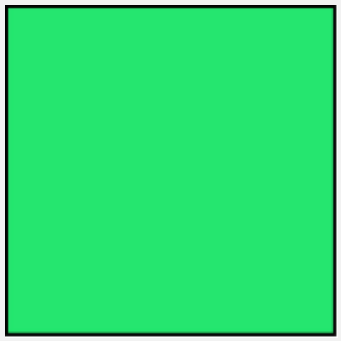
CSS transitions

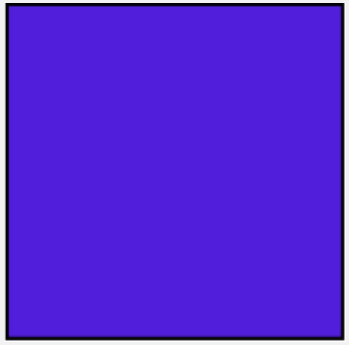
- It is a typical situation to create CSS styles for when a special state occurs
  - For example, when the mouse hovers a link, a feedback is usually given
- With CSS, when a state transition is triggered, it is possible to instruct the browser to perform a **smooth transition between the properties**, instead of just changing them, as it normally does
- For a transition to take place, an element must have a **change in state**, and a **different property value** has to be declared on that state change
- Typical example: highlight menu items (example)

CSS transitions

- Immediate state change applied to a SVG square

```
svg rect {
        fill: #00E969 ;
}
svg rect:hover {
        fill: #5100DF ;
}
```





- Smooth state change applied to a SVG square

```
svg rect {
        fill: #00E969 ;
        transition: all 2s;
}
svg rect:hover {
        fill: #5100DF ;
}
```

https://codepen.io/teresaterroso/pen/jOEpOvg

CSS transitions

- To create an animated transition between CSS animatable properties, one uses the **`transition` property**
- The transition property is a shortcut for a number of sub-properties that are:
  - `transition-delay`: time in seconds (s) or milliseconds (ms) before the transition animation starts
  - `transition-duration`: (mandatory) time in seconds or milliseconds that the animation takes from start to finish; no duration means no transition
  - `transition-property`: property, or properties, to animate; the special keyword `all` represents all properties at the same time
  - `transition-timing-function`: specifies the speed curve for the transition, and can take different values (next slide)

CSS timing functions

- `transition-timing-function`: specifies the speed curve for the transition, and can take the values:
  - `linear`: the same speed from start to end
  - `ease`: (default) transition effect with a slow start, then fast, then end slowly
  - `ease-in`: transition effect with a slow start
  - `ease-out`: transition effect with a slow end
  - `ease-in-out`: transition effect with a slow start and end
  - `cubic-Bezier(n,n,n,n)`: define the values in a cubic-bezier function

https://codepen.io/vailjoy/live/PWWbpW

CSS transitions

- Not all of the four transition-related properties are required to build a transition
- CSS transitions can be controlled using only the shorthand `transition` property or by explicitly declare its sub-properties

```
transition: property duration timing-function delay
```

```
#delay {
        font-size: 14px;
        transition-property: font-size;
        transition-duration: 4s;
        transition-delay: 2s;
}

#delay:hover {
        font-size: 36px;
}
```

```
#delay {
        font-size: 14px;
        transition: font-size 4s 2s;
}

#delay:hover {
        font-size: 36px;
}
```

https://codepen.io/teresaterroso/pen/xxbJwPE

CSS transitions

- More than one property can be animated with transitions

```
#heart {
        transform-origin: 25% 25%;
        fill: red;
        transition: fill 1s, transform 4s;
}

#heart:hover {
        transform: scale(1.5);
        fill: green;
}
```

```
#heart {
        transform-origin: 25% 25%;
        fill: red;
        transition-property: fill, transform;
        transition-duration: 1s, 4s;
}

#heart:hover {
        transform: scale(1.5);
        fill: green;
}
```

https://codepen.io/teresaterroso/pen/BayPovv

CSS transforms

- With the CSS `transform` property one can perform the following 2D or 3D transformation methods on HTML elements:
  - `translate(x,y)` / `translateX(x)` / `translateY(y)`: 2D translations
  - `translate3d(x,y,z)` / `translateZ(z)`: 3D translations
  - `scale(x,y)` / `scaleX(x)` / `scaleY(y)`: 2D scaling
  - `scale3d(x,y,z)` / `scaleZ(z)`: 3D scaling
  - `rotate(θdeg)`: 2D rotation
  - `rotateX(θdeg)` / `rotateY(θdeg) )` / `rotateZ(θdeg)` : 3D rotation
  - `skewX(θdeg)` / `skewY(θdeg)`: 2D skews
  - `initial`: sets the property to its default value

CSS transforms playground

CSS transforms

- While trivial to apply, some transform (namely scaling and rotation) may not perform as expected
- The reason why is the position of the **transform origin**, by default the top left corner
- To transform an element by its center, for instance, it is mandatory to change the transform origin
- Luckily, CSS provides a simple property to change the transform origin: **`transform-origin: x-axis y-axis z-axis`**
  - accepted values for each axis are

```
x-axis: left | center | right | length | %
y-axis: top | center | bottom | length | %
z-axis: length
```

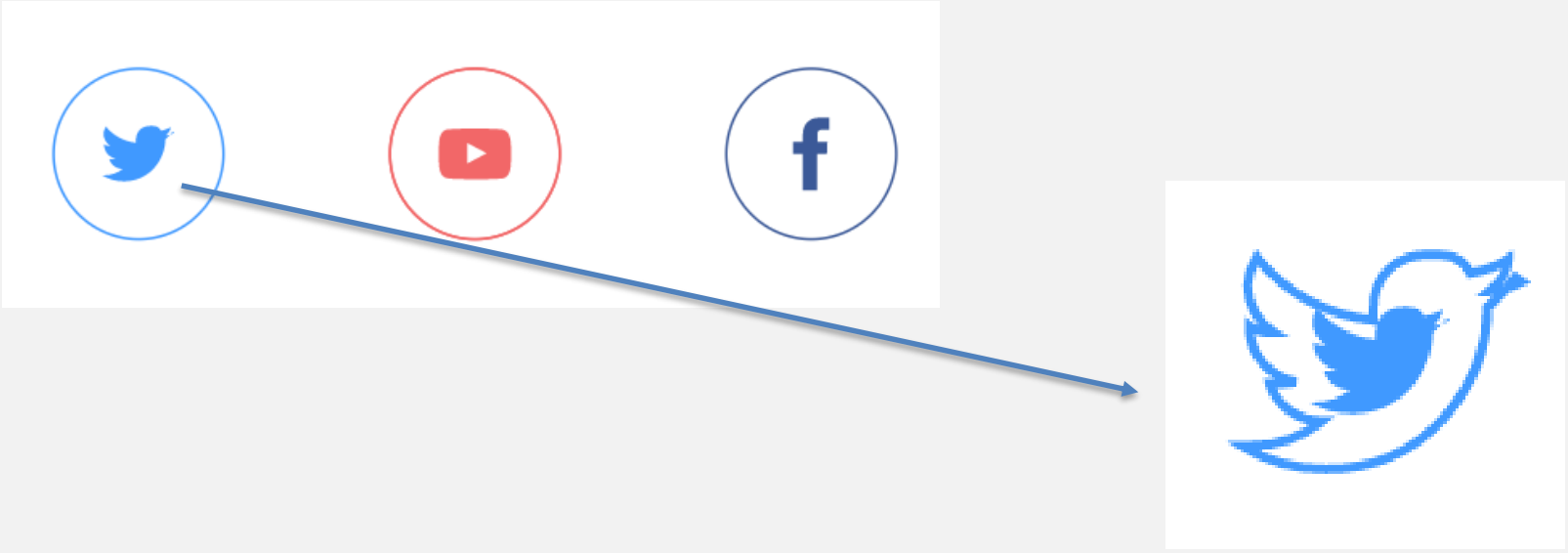
  - E.g., `transform-origin: 50% 50%` places the origin at the center of the element

Try yourself….

1. In SVG, draw two balls, like the image below. Using CSS transitions, when user hovers the mouse cursor over the SVG element, move the small ball to the right when “hit” by the large ball

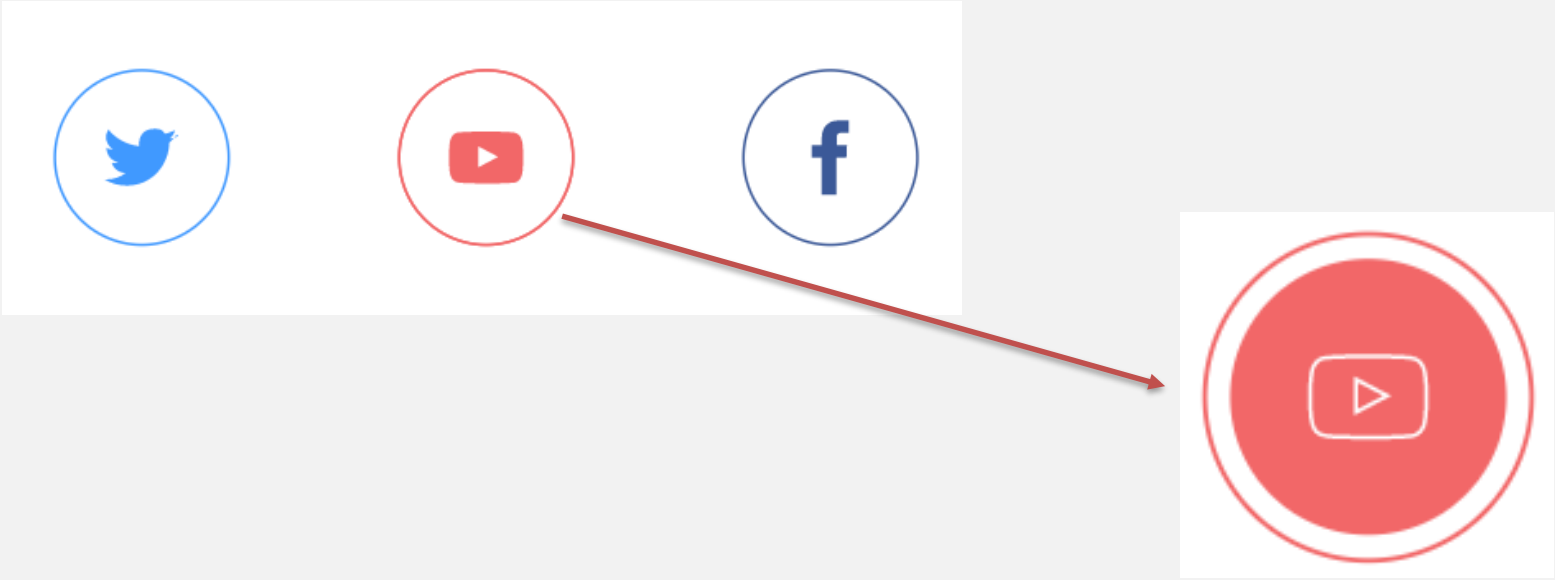
Try yourself....

2. Consider the following well-knowed icons, in SVG. Use CSS transitions to animated the hover state (duration for all: 1s).
   - Twitter icon group: when the user hovers the mouse cursor over, make the outline scale down to 0, and a second icon appear by change the stroke color from transparent to #4099ff, and at the same time scale it up to 2



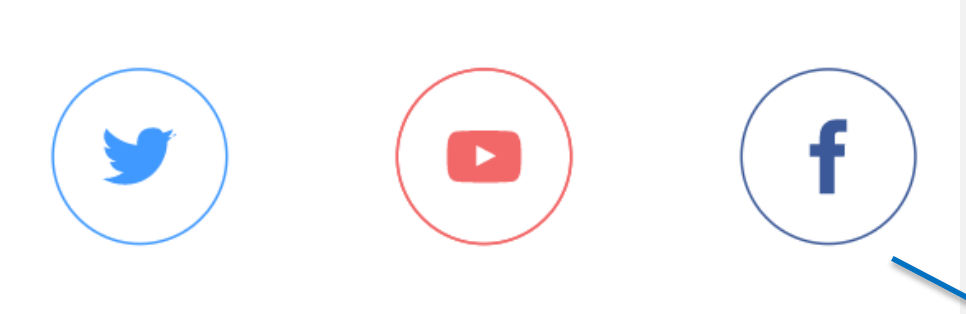
Try yourself....

2. Consider the following well-knowed icons, in SVG. Use CSS transitions to animated the hover state (duration for all: 1s).
   - Youtube icon group: when the user hovers the mouse cursor over, make the outline scale up to 1.2, change the inner circle fill color to #f26768 and the icon stroke color to white



Try yourself....

2. Consider the following well-knowed icons, in SVG. Use CSS transitions to animated the hover state (duration for all: 1s).
   - Facebook icon group: when the user hovers the mouse cursor over, make the outline circle disappear by change its opacity to 0 and at the same time scale it up to 1.8, make the detail appear and at the same time scale it down to 0.8



CSS animations

- But what to do when one just wants to animate one or more elements, without a state transition?
- The CSS standard specifies one property and one rule, that, when used together, produce animation
- The property is **`animation`** and the rule is **`@keyframes`**
- Keyframing is an animation technique in which special frames are marked, and the properties of an object are explicitly given
  - For the remaining frames, the values in between the keyframes are interpolated
- E.g.: animate a ball from left to right, linearly
  - 2 keyframes would be required: the initial, with the position of the ball on the left, and the final, with the position of the ball on the right

CSS animations

- CSS rule `@keyframes` syntax:

  ***@keyframes** animation_name {keyframes-selector {css-styles;}}*

  animation_name: name by which this particular animation will be referred

  keyframes-selector: percentage along the animation (0%-100%)

  css-styles: values of the properties at those places

- See the following keyframed animation that, once applied will change the fill color (of whatever element it is applied to)

```
@keyframes square-animation {
    0% {
        fill: #00E969;
    }
    100% {
        fill: #5100DF;
    }
}
```

```
@keyframes square-animation {
    from {
        fill: #00E969;
    }
    to {
        fill: #5100DF;
    }
}
```

CSS animations

- This 2nd example shows a more complex animation, named `scaleit`, there are three keyframes:

```
@keyframes scaleit {
        0% {
                fill: red;
        }
        80% {
                transform: scale(1);
                fill: green;
        }
        100% {
                transform: scale(2);
                fill: green;
        }
}
```

CSS animations

- Defining keyframes animations, per-se, does not animate anything
- The actual animation occurs when the **`animation`** property of CSS is used
- Like `transition`, **`animation`** is a shortcut for a number of properties:
  - `animation-delay`: time in seconds (s) or milliseconds (ms) before the animation starts
  - `animation-duration`: (mandatory) time in seconds or milliseconds that the animation takes from start to finish; no duration means no transition
  - `animation-direction`: whether an animation should be played forwards, backwards or in alternate cycles. Alternate supposes repetition:

    `normal` | `reverse` | `alternate` | `alternate-reverse`

CSS animations

- **`animation`** sub-properties:
  - `(…)`
  - `animation-fill-mode`: element style when the animation is not playing (before it starts, after it ends, or both), or in which CSS style is the animation going to end:

    `none` | `forwards` | `backwards` | `both`
  - `animation-iteration-count`: number of times an animation should be played:

    `number` | `infinite`
  - `animation-play-state`: whether the animation is running or paused:

    `paused` | `running`
  - `animation-timing-function`: speed curve for the animation (same values as in transitions – slide 11)

CSS animations

- If one wants to play the `scaleit` animation for 2 seconds, infinitely and alternating, one writes:

https://codepen.io/teresaterroso/pen/QWwBZbd

```
@keyframes scaleit {
        0% {
                fill: red;
        }
        80% {
                transform: scale(1);
                fill: green;
        }
        100% {
                transform: scale(2);
                fill: green;
        }
}

element {
        animation: 2s scaleit alternate infinite;
}
```

Try yourself….

1. Implement keyframe animations on the SVG of file SVGanimations_ex1.html:
   a) SVG group element `rim` is animated by the animation named `rectangle`, that scales up from 0 to 1 the red element, and at the same time, makes it from totally transparent to totally opaque
   duration: 1s delay: 5s

Try yourself....

1. Implement two keyframe animations on the SVG of file SVGanimations_ex1.html:
   a) SVG group element `rim` is animated by the animation named `rectangle`, that scales up from 0 to 1 the red element, and at the same time, makes it from totally transparent to totally opaque
   duration: 1s delay: 5s

Try yourself....

1. Implement two keyframe animations on the SVG of file SVGanimations_ex1.html:
   b) SVG group element `text` is animated by the animation named `text-anim`, that animates CSS properties stroke-dashoffset, opacity and fill. In the beginning: stroke-dashoffset 210, totally transparent, no fill color. At 50%, stroke-dashoffset changes to 0, fill color #fafafa opaque. At 75%, fill color #F7931E and in the end, a 2D rotation transformation of 720 degrees. duration: 5s