COMP4021
Internet Computing

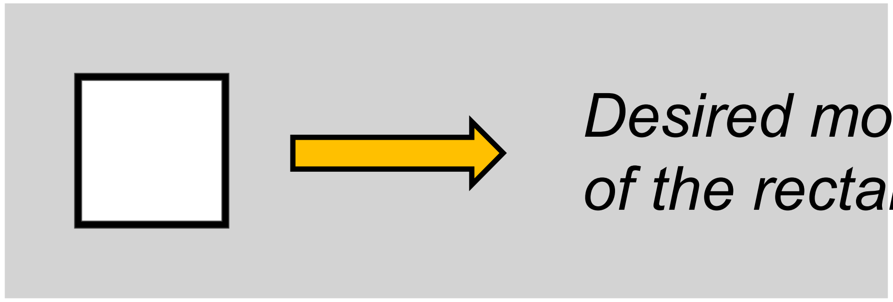
Web Animation

David Rossiter & Gibson Lam

What We Want

```
<!DOCTYPE html>
<html>
  ...
<body>
  <svg xmlns="http://www.w3.org/2000/svg"
       width="600px" height="200px"
       style="background: lightgray">
    <rect x="50" y="50" width="100" height="100"
          stroke="black" stroke-width="5" fill="white"/>
  </svg>
</body>
</html>
```

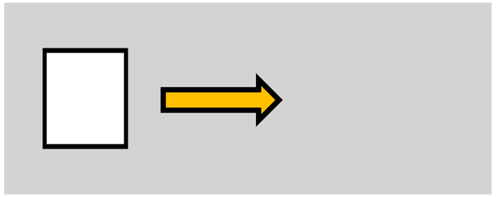
- We want to make this SVG rectangle move from left to right



Different Approaches

- To make the desired animation, we will quickly show the code to do that in these ways:
  1. Using JQuery
  2. Using SVG Animations
  3. Using CSS Animations
- We will then give a brief introduction to each of them

1) jQuery Example



- You can do the animation using JavaScript or JQuery
- Here we use JQuery because the code is simpler than writing our own JavaScript:

we will see the movement once the webpage is loaded.

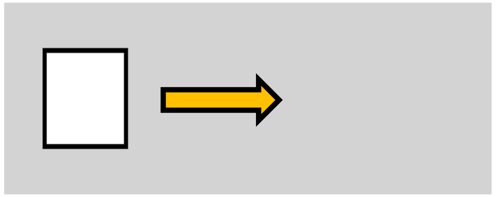
```
$(document).ready(function() {
    $("rect").animate({ x: 450 }, 2000);
});
```

selector: 这里用的tag name，因为只定义了一个rect

Move the rectangle to x = 450 in 2 seconds

syntax： .animate({ attribute : movement }， duration)

2) SVG Animation Example



- SVG has its own animation commands
- SVG animation controls are done using *animation* :

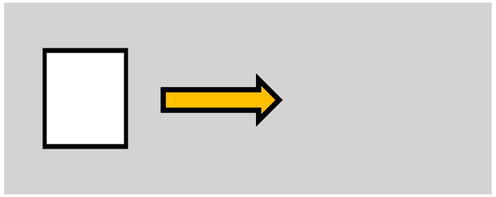
```
<rect x="50" y="50" width="100" height="100"
      stroke="black" stroke-width="5" fill="white">
  <animate attributeType="XML" attributeName="x"
           from="50" to="450" dur="2s"
           fill="freeze"/>
</rect>
```

SVG own animation tag, 放在rect tag 里面

at the end of the animation, freeze! (means do not do anything further. (As opposite to repeat or reverse, etc.)

This animates the rectangle by changing the x attribute value over 2 seconds

3) CSS Animations Example



- You can use CSS animation instructions
- To move the SVG rectangle, this style is applied to it:

```
@keyframes move-rect {
    from { transform: translateX(0px); }
    to   { transform: translateX(400px); }
}

rect { selector: all rectangles
    animation: move-rect 2s;
}
```

These are defined inside the style area

Using JQuery

- The simplest way to use `animate()` is:

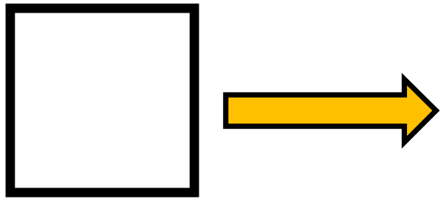
终点的properties

`animate(` *CSS properties* `,` *time* `)`

- It allows you to animate one or more CSS attributes over a certain time here in pure html, the attribute names are left and top, rather than x and y.

A Quick jQuery Example

```
<style>
#me {
  position: relative;
  left: 0;
  top: 0;
  width: 100px;
  height: 100px;
  border: 5px solid black;
}
</style>
...
<div id="me"></div>
...
```



- In this example, a div is moved to the right by changing the CSS `left` property

```
$("#me").animate({ left: 200 }, 2000);
```

$\underbrace{\texttt{\{ left: 200 \}}}_{\text{}}$

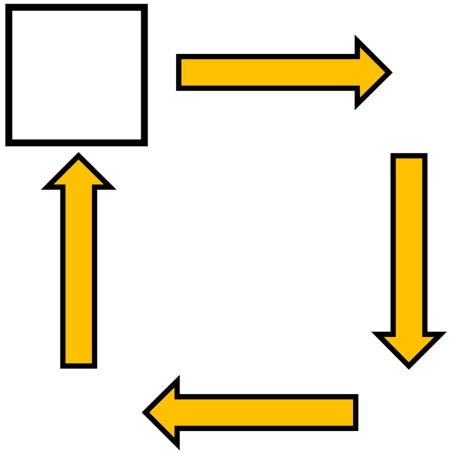
One or more properties can be used

Queuing Animations Together

- The animations can be sequenced like this:

```
$("#me").animate({ left: 200 }, 2000);
$("#me").animate({ top : 200 }, 2000);
$("#me").animate({ left: 0   }, 2000);
$("#me").animate({ top : 0   }, 2000);
```

- The animations will be played one after another, in a queue



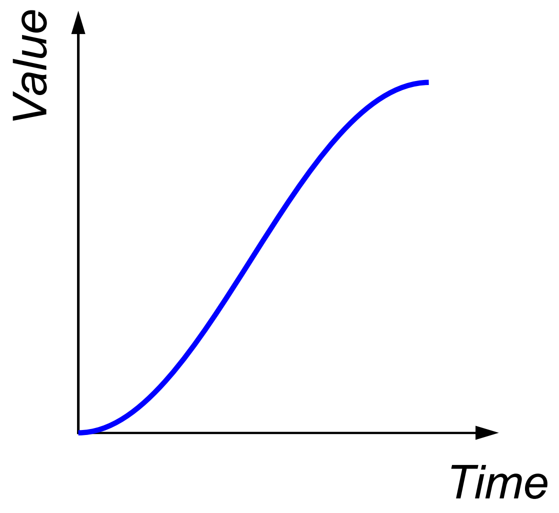
Easing – Rate of Animation

- If you look at the animation now, you can see that the movement does not have a constant speed
- The div moves relatively slower at the start and at the end within one cycle of animation
- This is called *easing*
- The default value for easing is `swing` but you can change it to `linear` so that the speed of movement is constant

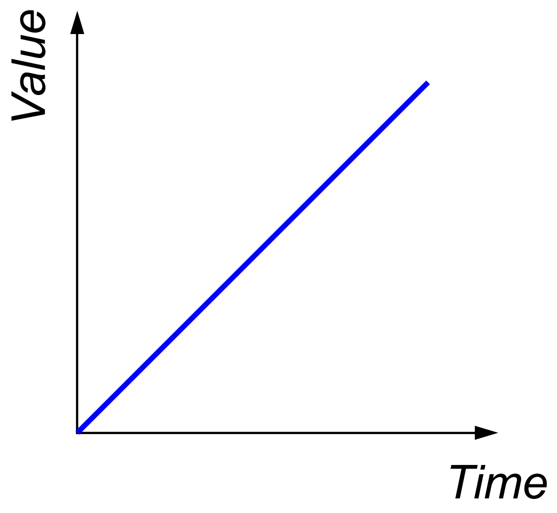
Controlling the Animation Speed

Using JQuery

- Swing



- Linear



Using Linear Easing

- This third parameter is optional
- It adjusts the easing value of the animations, e.g.:

```
$("#me").animate({ left: 200 }, 2000, "linear" );
$("#me").animate({ top : 200 }, 2000, "linear" );
$("#me").animate({ left: 0   }, 2000, "linear" );
$("#me").animate({ top : 0   }, 2000, "linear" );
```

- A linear movement is especially useful when there are multiple animations, so they all have equal speed

A Note About the Properties

Using JQuery

- You can provide more than one property to the `animate()` function, like this:

```
$("p").animate(
    { fontSize: "150px", borderWidth: "5px" }, 2000
);
```

用JS的时候也是这样

- When you use the CSS property names with jQuery if the property has a '-' then you need to change it a little e.g. `background-color` becomes `backgroundColor`

SVG Animation Tags

- For SVG, you can do these:
  - `<animate>`
  - `<animateTransform>`
  - `<animateMotion>`
- Here we will focus on the `<animate>` tag

Using the Animate Tag

- To animate an SVG element, an animate tag goes inside the element
- For example, to animate an SVG circle an animate tag is placed inside the `<circle>` tag

```
<circle ...>
    <animate .../>
</circle>
```

- If there's no animation then you don't usually write the code `<circle ...> … </circle>` - instead you do `<circle … />`

end of the tag

Information in the Animate Tag

- Here are some essential information that you provide to the `<animate>` tag:
  - The attribute that you want to animate
  - The range of values that the attribute changes from/to
  - The duration of the animation
  - The number of times that you want to repeat the animation

at the end, it jumps back to the beginning if you don't use "freeze"

- An example is shown on the next page

Using Animate

```
<circle cx="250" cy="250" r="200"
        stroke="black"
        stroke-width="5" fill="red">

  <animate attributeName="r"
           from="200" to="50"
           dur="1s"
           repeatCount="1"
  />
</circle>
```

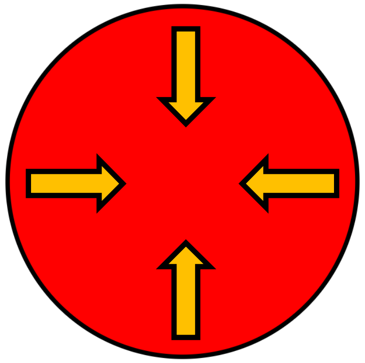
全部都要加引号~

Attribute to be animated

Range of values

Duration

Number of times to repeat



Animating a Circle

Attribute Value After Animation

- After performing the animation, you have the option to set the attribute to either the starting value or the ending value of the animation
  - For the former, you set the `fill` attribute for the animation to `remove` (it's the default option) (or don't provide the attribute at all)
  - For the latter, you set the `fill` attribute to `freeze`

Animating Over Values

- In addition to the `from` and `to` attributes, the range of values that you can animate can be set using the `values` attribute
- So this:

```
<animate attributeName="r" from="200" to="50"
          dur="1s" repeatCount="1"/>
```

and this:

```
<animate attributeName="r" values="200;50"
          dur="1s" repeatCount="1"/>
```

are equivalent

Making a Repeating Animation

- You can then adjust the `values` and `repeatCount` attributes to make a repeating animation

```
<animate attributeName="r"
         values="200;50;200"
         dur="1s"
         repeatCount="indefinite"
/>
```

This means repeating forever

- The attribute values have to go around and back to the starting value

Using Multiple Animations

- You can make as many animations as you want by putting them inside the SVG element, like this:

```
<circle ...>
  <animate attributeName="r"
           values="200;50;200" dur="1s"
           repeatCount="indefinite"/>

  <animate attributeName="stroke-width"
           values="5;40;5" dur="0.4s"
           repeatCount="indefinite"/>
</circle>
```

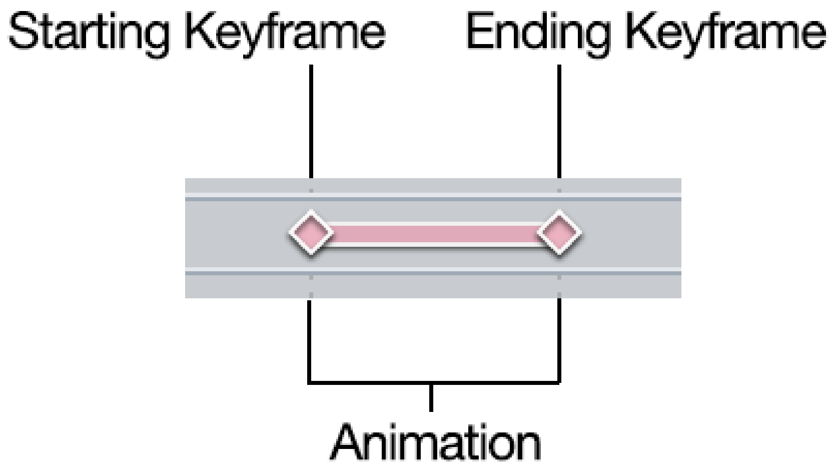
Animating radius

Animating outline

both animations work together, but they do not need to have the same timing.

CSS Animations

- Now we will look at making animations inside CSS using the `@keyframes` rule
- Keyframes are the important values of an animation, from start to finish
- Once the keyframes are ready, you can use them through the `animation` property



Keyframes

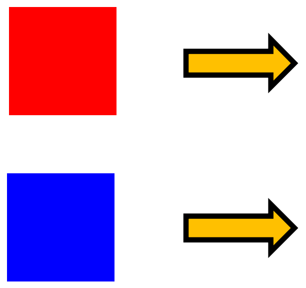
- Keyframes are sets of CSS properties that represent their values along a timeline
- For example, the following rule, called `move-right` , changes the `left` property from `0px` to `50px`

```
@keyframes move-right {
    from { left: 0px }
    to   { left: 50px }
}
```

Applying the Rule

- The previous keyframes rule can then be applied to different elements
- For example, it can be applied to two separate divs as shown below:

```
<div id="box1"></div>
<div id="box2"></div>
```



```
#box1, #box2 {
  position: relative;
  left: 0;
  width: 50px;
  height: 50px;
}
#box1 {
  top: 0;
  background: red;
  animation: move-right 2s;
}
#box2 {
  top: 80px;
  background: blue;
  animation: move-right 4s;
}
```

Using CSS

Animation Name and Duration

- For each div, the `animation` property specifies the animation name and duration

```
#box1 {
  animation: move-right 2s;
}
#box2 {
  animation: move-right 4s;
}
```

所以可以用同一个keyframe定义的animation但是不同的duration

- In the example, both boxes move to the right, but with a different speed

More Animation Properties

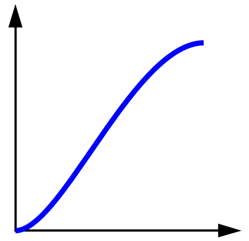
- Optionally, you can use more animation properties:
  - `animation-timing-function` , the easing to be used for the animation – next slide
  - `animation-iteration-count` , the number of times the animation is repeated, which can be `infinite` to make it non-stop
  - `animation-fill-mode` , the property values to retain after the animation finished, which can be `none` or `forwards` (meaning the end value)

Animation Timing Function

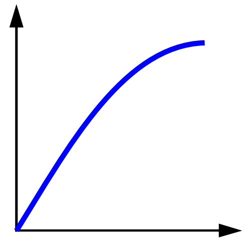
- Similar to the easing in the JQuery animation, you can adjust the rate of animation for the CSS animation
- Some values available are `ease` (similar to `swing` in JQuery), `linear`, `ease-in`, `ease-out`, `ease-in-out`
- The default easing is `ease`, which is similar to what JQuery does

Easing in CSS Animations

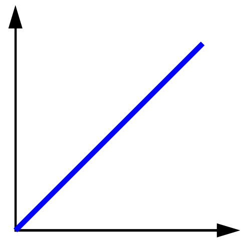
- Ease



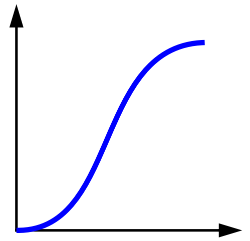
- Ease-Out



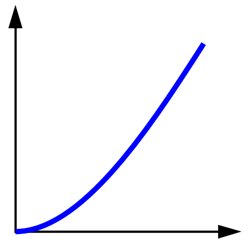
- Linear



- Ease-In-Out



- Ease-In



The Transform Property

Using CSS

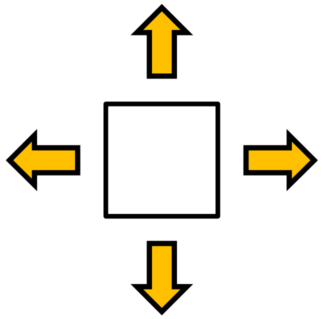
- Instead of animating the regular CSS property, you can use the `transform` property to make all sorts of animations

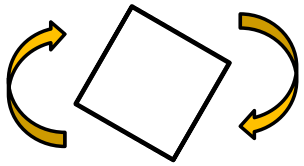
- You can either translate (=move), rotate or scale the elements

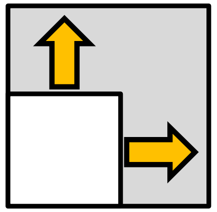
    transform：translate/rotate/scale()

Transforming Elements

- Translation
  - `translate(x, y)`, or
  - `translateX(x)` and `translateY(y)`
- Rotation
  - `rotate(angle)`
- Scaling
  - `scale(x, y)`, or
  - `scaleX(x)` and `scaleY(y)`







Animating a Transform

Using CSS

- For example, here we animate the `transform` property:

```
@keyframes transform-animation {
  0%   {
    transform: translate(0px, 0px)
               rotate(0deg);
  }
  50%  {
    transform: translate(100px, 100px)
               rotate(720deg);
  }
  100% {
    transform: translate(200px, 200px)
               rotate(0deg);
  }
}
```

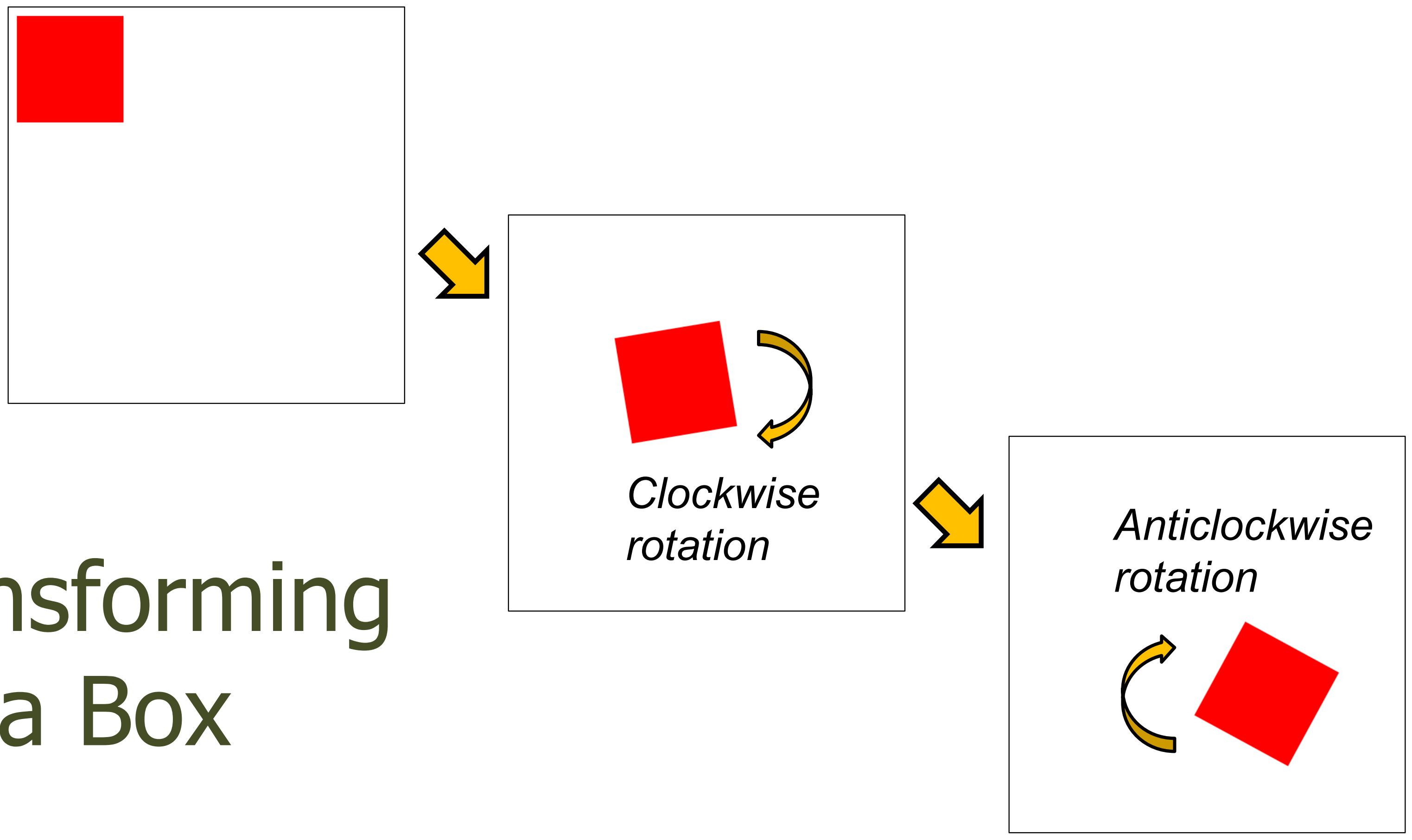
加単位！

Percentages have been used to specify the keyframes in this example

Applying two transforms at the same time

Using CSS

Transforming a Box



reverse back to "rotate(0deg)"