CSS ANIMATION

OBJECTIVES

- Understand pseudo classes like ‘hover’ and ‘active’. Understand they exist without us having to ‘apply’ the classes.
- Understand @keyframe animations. Know when you would need to use one

PSEUDO CLASSES

- Predefined classes in CSS.
- Are ‘called’ in a CSS file via pseudo class selectors:
  - :hover
  - :active

- They are applied to elements after the type, class or id. For example:

```
#specialBox:hover {
 background-color: pink;
}
```

TRANSITION PROPERTY

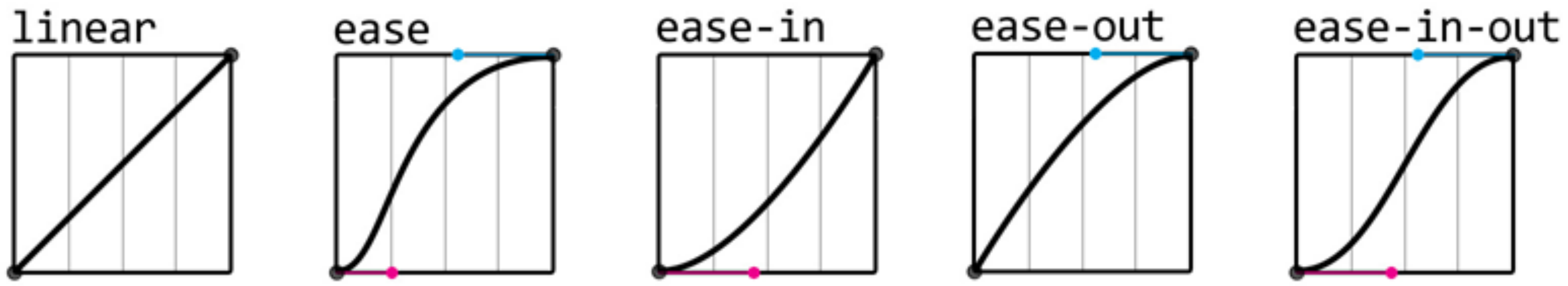
•We define the style of the element for these ‘hover’ and ‘active’ scenarios.

The new styles are can be put on a timer via the transition property.
Example:

```
 div {
  transition: all 1s;
 }
```

CSS TRANSITION PROPERTY

- transition: [property] [duration] [timing-function] [delay]
- Property - what property you are defining the transition for. Can be opacity, height, width, or even ‘all’
- Duration - how long for the transition to the new style to complete
- Timing Function - how the transition animates. Valid values:
  - ease - a transition with a slow start, then fast, then end slowly (default)
  - linear -a transition effect with the same speed from start to end
  - ease-in - a transition effect with a slow start
  - ease-out - a transition effect with a slow end
  - ease-in-out - a transition effect with a slow start and end
- Delay - Time before start.


linear
ease
ease-in
ease-out
ease-in-out

CODE ALONG

Sample pseudo classes

MULTI-STEP ANIMATIONS

- What do we use when we want more than one ‘step’ in our animation?
- Keyframes!
- There are two parts to a keyframe animation.
- Keyframes - defines the steps of the animation
- Animation CSS Property - assigns keyframes to an element, defines what is animate.

KEYFRAMES SYNTAX

- Keyframes contain:
  - Name
  - Stages (percentage of animation completed)
  - Properties to apply.

```
@keyframes bounceIn {
  0% {
    transform: scale(0.1);
    opacity: 0;
  }
  50% {
    transform: scale(1.5);
    opacity: 1;
  }
  100% {
    transform: scale(1);
  }
}
```

CSS ANIMATION PROPERTY

- Defines HOW an element will animate, not what actually animates.
- SYNTAX:

```
animation: [name] [duration] [timing-function] [delay] [iteration-count]
[direction][fill-mode];
```

  - **name - the keyframes you've defined.**
  - **duration - time to get from 0% - 100%**
  - timing-function- how it eases. (ease, linear, ease-in, ease-out, ease-in-out.)
  - delay - how long before it starts
  - **iteration-count - how many loops (infinite, #)**
  - direction - normal (0-100%), reverse(100-0%), alternate(0-100-0)
  - fill-mode:
    - backwards - Before the animation (during the animation delay), the styles of the initial keyframe (0%) are applied to the element.
    - forwards - After the animation is finished, the styles defined in the final keyframe (100%) are retained by the element.

- EXAMPLE:

```
div {
  animation: bounceIn 2s linear 1s infinite normal forwards;
}
```

CODE ALONG

Keyframe Animations

THINGS TO REMEMBER

- There are multiple ways to achieve the same goal.
- We’ve explored using pseudo classes to trigger animation
- We’ve explored using @keyframe animations to control animation.

YOUR TURN

Solar System