INTERACTIVE CSS

PSEUDO CLASSES

A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states

PSEUDO CLASSES

:hover

Applies css properties when an element is hovered over

```
.box:hover{
  background-color: red;
}
```

PSEUDO CLASSES

:focus

Applies css properties when an element is in focus. (usually when an input is selected)

```
input:focus{
 border: 1px solid red;
}
```

PSEUDO CLASSES

:active

Applies css properties when an element is active or clicked

```
p:active{
  color: green;
 }
```

PSEUDO CLASSES

Can also be used to select elements more specifically

PSEUDO CLASSES

:first-of-type

Selects the first element that matches the selector given

```
.box:fist-of-type{
 color: purple;
}
```

PSEUDO CLASSES

`:nth-of-type()`

Selects the first element that matches the pattern given within the parentheses

```
.box:nth-of-type(2)
.box:nth-of-type(even)
.box:nth-of-type(3n)
```

PSEUDO CLASSES

There are a lot more pseudo classes that can be found

HERE

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes#Index_of_standard_pseudo-classes

Pseudo Classes
:hover
:focus
:active
:nth-of-type()
Transition
Transition Property
Duration
Timing Function
Delay
Key Frame
Iteration Count
Animation Direction
Translate
Scale
Rotate
Perspective

YOUR TURN

Fork this Codepen

TRANSITIONS

CSS transitions allows you to change property values smoothly (from one value to another), over time.

TRANSITIONS

`transition-property`

The CSS property you want to add an effect to

```
.box {
  transition-property: width;
}
```

NOT ALL PROPERTIES ARE "ANIMATABLE"

ANIMATABLE PROPERTIES

Animatable properties have numerical values.

background
background-color
background-position
background-size
border
border-bottom
border-bottom-color
border-bottom-left-radius
border-bottom-right-radius
border-bottom-width
border-color
border-left
border-left-color
border-left-width
border-right
border-right-color
border-right-width
border-spacing
border-top
border-top-color
border-top-left-radius
border-top-right-radius
border-top-width
bottom
box-shadow
clip
color
column-count
column-gap
column-rule
column-rule-color
column-rule-width
column-width
columns
filter
flex
flex-basis
flex-grow
flex-shrink
font
font-size
font-size-adjust
font-stretch
font-weight
height
left
letter-spacing
line-height
margin
margin-bottom
margin-left
margin-right
margin-top
max-height
max-width
min-height
min-width
object-position
opacity
order
outline
outline-color
outline-offset
outline-width
padding
padding-bottom
padding-left
padding-right
padding-top
perspective
perspective-origin
right
text-decoration-color
text-indent
text-shadow
top
transform
transform-origin
vertical-align
visibility
width
word-spacing
z-index

TRANSITIONS

`transition-duration`

How much time the effect should take

```
.box {
  transition-property: width;
  transition-duration: 2s;
}
```

TRANSITIONS

`transition-timing-function`

the speed curve of the transition effect

```
.box {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
}
```

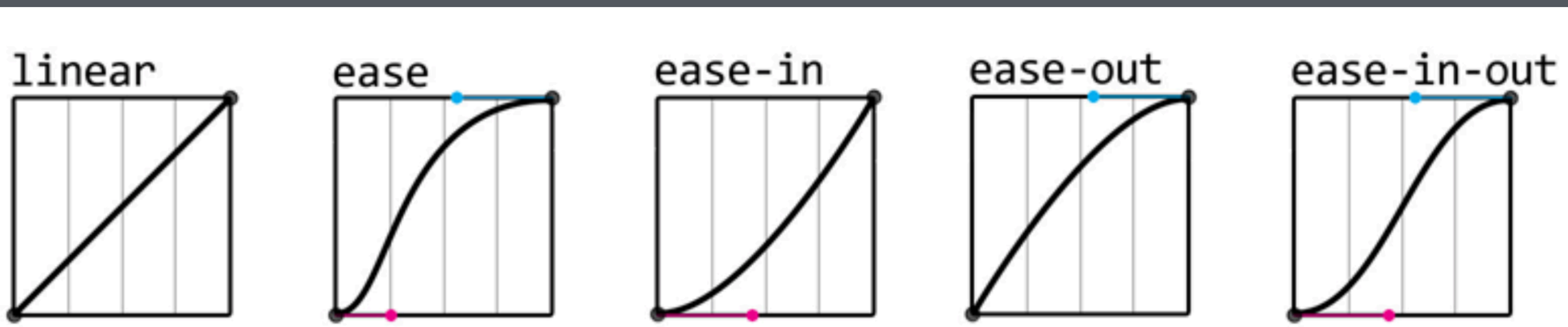
TRANSITIONS

`transition-timing-function`

ease: slow start, then fast, then end slowly (default)
linear: same speed from start to end
ease-in: slow start
ease-out: slow end
ease-in-out: slow start and slow end

TRANSITIONS

transition-timing-function



TRANSITIONS

transition-delay

Amount of time before the transition starts

```
.box {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 3s;
}
```

TRANSITIONS

`transition`

All 4 properties in 1!

property, duration, timing-function, delay

```
.box {
  transition: width 2s linear 3s;
}
```

TRANSITIONS

`transition`

You can do multiple transitions in one, or all!

```
.box {
  transition: width 2s,
              background-color 3s;
}

.box {
  transition: all 2s;
}
```

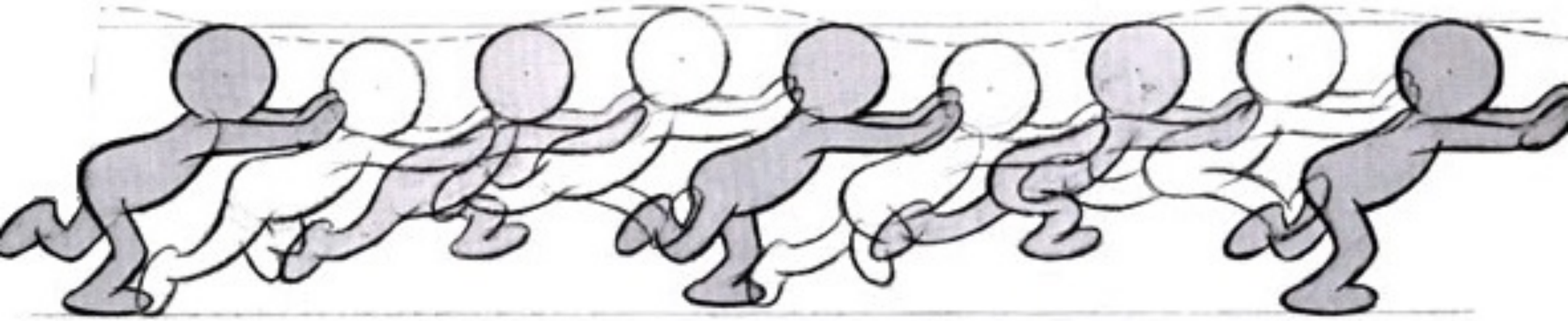
Pseudo Classes
:hover
:focus
:active
:nth-of-type()
Transition
Transition Property
Duration
Timing Function
Delay
Key Frame
Iteration Count
Animation Direction
Translate
Scale
Rotate
Perspective

EXAMPLE

Lets add some transitions to the codepen from earlier

KEYFRAMES

Keyframes are usually found in hand-drawn animation. They define the starting, ending, or middle point of a smooth transition



The other drawings are **inbetweens**- they don't have to be drawn on a storyboard, because the animator can assume what they will look like without a visual reference.

KEYFRAMES

In web-based animations, keyframes work the same way - they represent the beginning, ending, or midpoint state of the element being animated. However, our inbetweens will be generated by code, instead of being filled in by hand later.

KEYFRAMES

The transition property allows us to define a beginning and end point for a state change. However, sometimes you'll want to have an element move through multiple states during an animation. That's where the CSS keyframes rule comes in.

KEYFRAME SYNTAX

CSS animation takes 2 steps:

1. Define the animation and what it will do
2. Apply it to an element

KEYFRAME SYNTAX 1

```
@keyframes move_box {
   0% {
      top: 0px;
   }
   50% {
      top: 20px;
   }
   100% {
      top: 40px;
   }
}
```

Animation Name

Keyframes

Can be as many as you want, but needs AT LEAST 0% and 100%

KEYFRAME SYNTAX 2

animation-name

animation-duration

animation-timing-function

animation-delay

animation-iteration-count

animation-direction

KEYFRAME SYNTAX 2

`animation-name`

The name of the animation

```
.box {
  animation-name: move_box;
}
```

KEYFRAME SYNTAX 2

`animation-iteration-count`

How many times the animation should run (can be infinite)

```
.box {
  animation-name: infinite;
}
```

KEYFRAME SYNTAX 2

animation-direction

Which way the animation will run

normal, reverse, alternate, alternate-reverse

```
.box {
  animation-direction: alternate;
}
```

KEYFRAME SYNTAX 2

`animation`

All properties in one

name duration timing-function delay iteration-count direction

```
.box {
  animation: move_box 4s linear 0s infinite alternate;
}
```

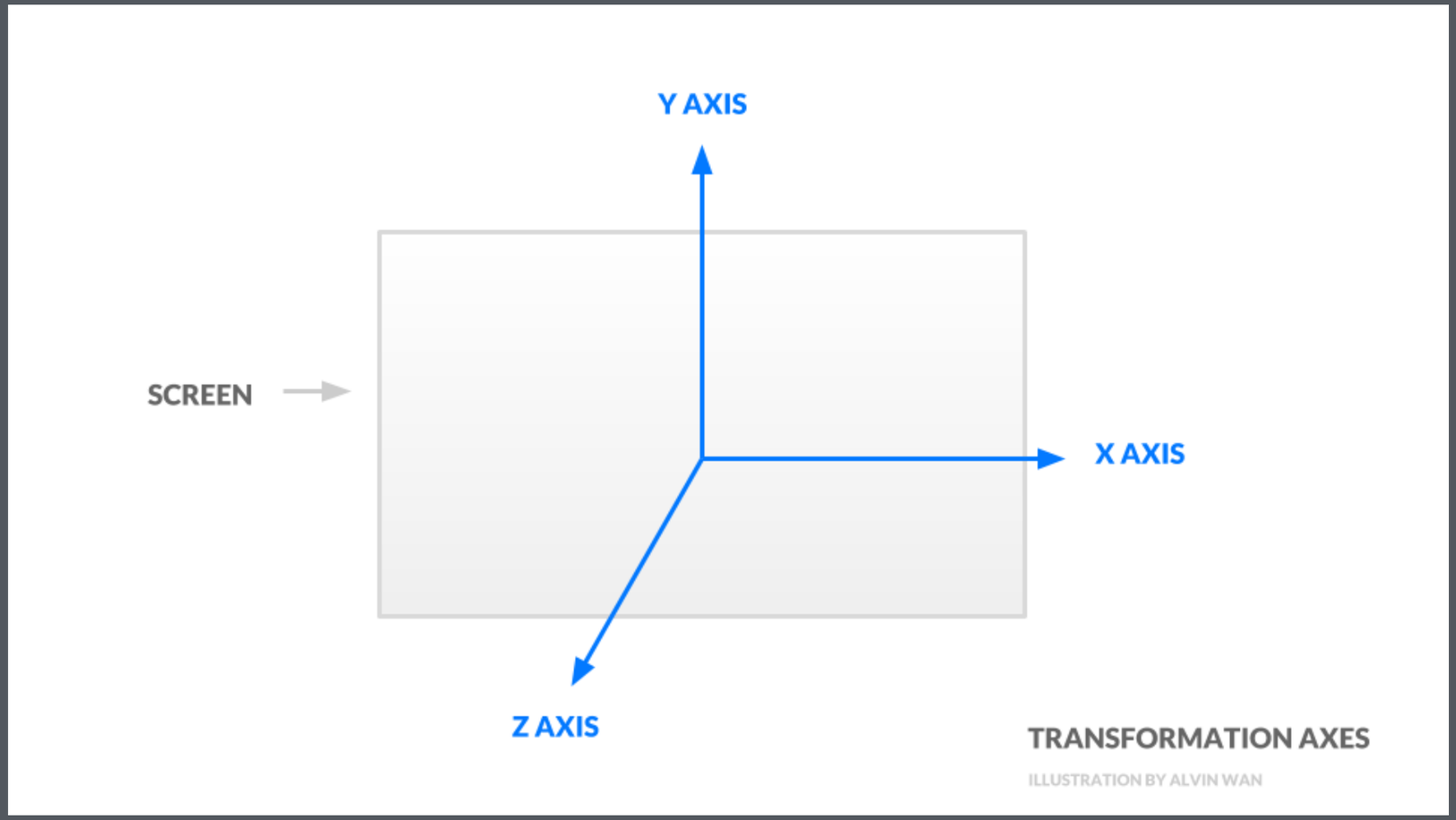
Pseudo Classes
:hover
:focus
:active
:nth-of-type()
Transition
Transition Property
Duration
Timing Function
Delay
Key Frame
Iteration Count
Animation Direction
Translate
Scale
Rotate
Perspective

YOUR TURN

Create animations with a partner

TRANSFORMS

CSS Transforms can move, rotate, scale and skew elements in 2d or 3d without affecting the document flow



TRANSFORMS SYNTAX

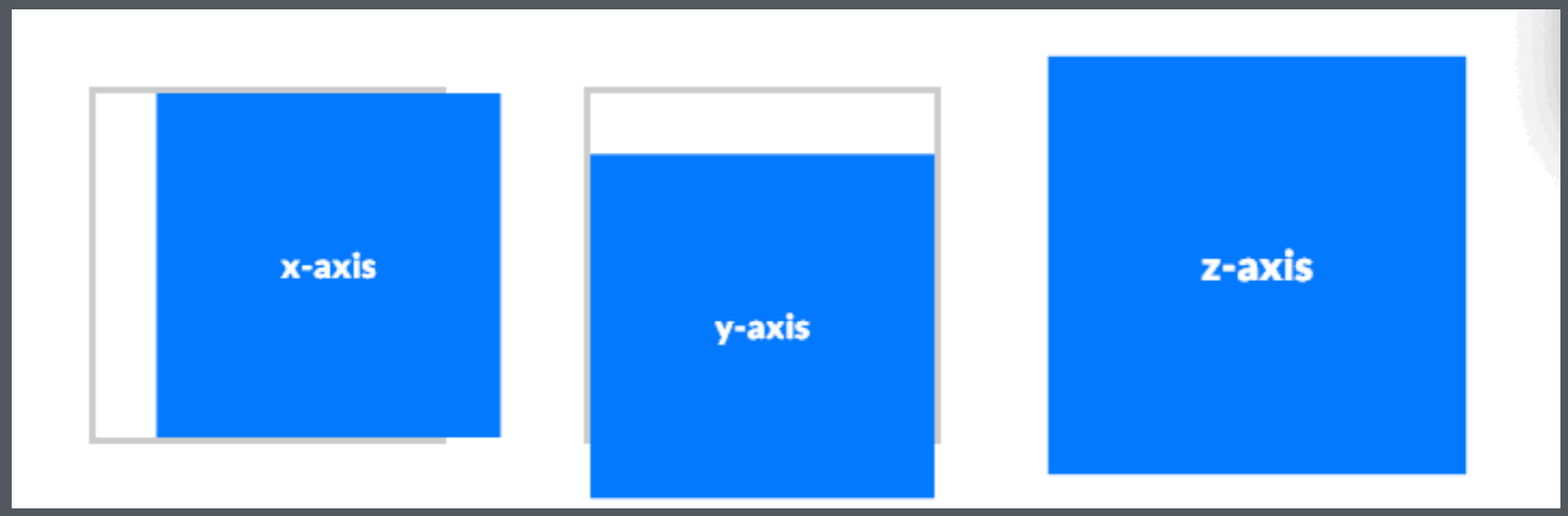
```
selector {
  transform: property(value);
}

.box {
  transform: rotate(45deg);
}
```

TRANSFORM

translate

Move elements along x, y and z axes

```
translateX(x)
translateY(y)
translateZ(z)
translate(x,y)
translate3d(x,y,z)
```



TRANSFORM

`scale`

Grow and shrink along x, y or z

```
scaleX(x)
scaleY(y)
scaleZ(z)
scale(x,y)
```

TRANSFORM

rotate

Rotates elements along x, y and z axes

```
rotateX(x)
rotateY(y)
rotateZ(z)
rotate(z)
```

TRANSFORM

perspective

Gives an element a 3D-space by affecting the distance between the Z plane and the user.

The further away the object, the less drastic 3D effects are.

`transform: perspective(200px)`

Pseudo Classes
:hover
:focus
:active
:nth-of-type()
Transition
Transition Property
Duration
Timing Function
Delay
Key Frame
Iteration Count
Animation Direction
Translate
Scale
Rotate
Perspective

YOUR TURN

And transforms to your animations!