FRAMER MOTION

- animation library for react.

- <u>Installation</u>

  npm install framer-motion

- <u>Basic Animations</u>

  <u>import</u>

  import { motion } from 'framer-motion'

```
<h2>  →   <motion.h2                          # on load size increases
              animate = {{ fontSize : 50}}>
                Hello world  ↑         ↑
                           JSX        = 50px
                           format.
          </motion.h2>

  # can have    x:100 , y:-100       (to move from original position)
                rotateX , rotateY , rotateZ    ( rotateZ : 180 )

<button>  →   <motion.button
                 animate ={{ Scale :1.2 }}
                   Create Your Pizza
              </motion.button>
```

- <u>Initial Value</u> (setting start point)

1) Move head title from screen top down to its real position -10 px

```
<motion.div   className = "title"
  initial  = {{ y: -250 }}
  animate  = {{ y:-10 }}
>
  <h1> Pizza Joint </h1>
</motion.div>
```

# note
-20 , -10 etc are offset value
- -10 means -10 px relative to initial position

2) on selection of pizza base bring select button from left of screen

```
{ pizza.base && (
   <motion.div  className = "next"
      initial = {{ x: "-100vw" }}
      animate = {{ x : 0 }}
   >
     <Link to = "/toppings">
        <button> Next </button>
     </Link>
   </motion.div>
)}
```

- <u>Adding speed, delay using transition</u>

specifies time from initial to animate

types = "spring" "tween" "inertia"

for some components default is spring & for some it is "tween" (smooth one).

eg)
```
<motion.div className = "title"
   initial = {{ y: -250 }}
   animate = {{ y : -10 }}
   transition = {{ delay : 0.2, type : 'spring', stiffness : 120 }}
>     <h1> Pizza Joint </h1>
</motion.div>
```

more stiffness more spring effect.
stiffness can only be added with spring

eg2)
```
<motion.div className = "home container"
   initial = {{ opacity : 0 }}
   animate = {{ opacity : 1 }}
   transition = {{ delay : 0.4, duration : 3 }}
>
```

duration can only be added with tween type
for this div default was tween.

Hover Effect

1) add shadow on hover to button

```
<motion.button
    whileHover = {{
        scale : 1.1,
        textShadow : "0px 0px 8px rgb(255, 255, 255)",
        boxShadow : " 0px 0px 8px rgb (255, 255, 255)'
        }}
    >
        Create your Pizza
    </motion.button>
```

2) on hover change color

```
<motion.li key = {topping} onClick = {() => addTopping (topping)}
    whileHover = {{ scale : 1.3 , originX : 0 , color : "#f8e112" }}
    transition = {{ type : "spring" , stiffness : 300 }}
    >
        - -
    </motion.li>
```

origin : 0   so scale on right side.

Variants

variants allow you to define animation states and organise them by name. They allow you to control animations throughout a component tree by switching a single animate prop.

eg outside return

any Name

```
const containerVariants = {
    hidden : {
        opacity : 0,
        x : '100vw'
    },
    visible : {
        opacity : 1,
        x : 0
    }
}
```

can give any name like hidden, start etc
This is just eg

in div

```
<motion.div
    variants = {containerVariants}
    initial = "hidden"
    animate = "visible"
    transition = {{ type : "spring" }}
    >
```

◦ Can even send transition to it

```
Const nextVariant ={                         <motion.div
  start: {                                      variants = {nextVariant}
     opacity :0,                                initial = 'Start'
     x: "100vw"                                 animate = 'final'
  },                                           >
  final : {
     Opacity: 1,
     x :0,
     transition : {
        type: "spring",
        delay : 0.5 }
     }
}
```

working with parent & children

no need to give initial & animate property to child if variables are defined with same name in parent & child.

eg
```
<motion.div                                   Const containerVariants = {
  variants = {containerVariant}                  hidden : {
  initial = "hidden"                                opacity :0,
  animate = "visible"                               x :'100vw'
>                                                },visible: {
<h2> Thank you for the order :) </h2>              opacity : 1,
 <motion.p                                          x:0,
   variants = {childVariant}                        transition : {
 >                                                     type : 'spring',
   You ordered a {pizza.base} with :                   mass : 0.4,
 </motion.p>                                           damping : 9,
 <motion.div                                           when :"beforeChildren",
   variants = {childVariant}  # child not              staggerChildren :0.4
                                given initial        }
                                & animate          }
 >                                               }
  {pizza.toppings.map (topping =>              const ChildVariant = {
    <div key ={topping}> {topping}</div>}        hidden: {
 </motion.div>                                      opacity : 0
 )                                               },
}                                                visible : {
                                                    opacity :1
                                                 }
                                               }
```

when → decide parent animation wrt children
staggerChildren : time betn each children

KeyFrames

used when want several transitions together.
eg left right left right . . .

eg

in a variant

```
visible : {
    x : [0 , -20 , 20 , -20, 20 , 0 ],
    transition : { delay : 2 }
},
```

```
< motion . button
    variants = - - -
    animate = "visible"
>
```

→

movement in x direction

- Scale up & down 3 times on hover.

```
< motion . button
    whileHover = {{
        scale : [1 , 1.1 , 1 , 1.1 , 1 , 1.1 , 1 ],
        textShadow : "0px 0px 8px rgb(255, 255, 255)"
    }}
>
```

- apply infinite time using yoyo transition

```
const buttonVariables ={
    hover : {
        scale : 1.1 ,   # can give keyframe too [1, 2, 0, 3] - -
        = =
        transition : {
            duration : 0.3,
            yoyo : Infinity     # apply infinite time keyframe value
        }                         can do yoyo : 10  so go through transition
    }                             10 times
}
```

° Animate Presence

eg → want to remove h2 after 4 sec with animation go above

```
→ const [showTitle, setShowTitle] = useState (true);
  Set Timeout (() => {
      SetShowTitle (false);
  }, 4000);
```

now, import {motion, AnimatePresence} from 'framer-motion';

↙ see format for conditional rendering

```
  <AnimatePresence>
    { ShowTitle && (
       <motion.h2
          exit = {{ y: -1000 }}
       > Thank you for order </motion.h2>
    )}
  </AnimatePresence>
```

• Animate Routes

when page changes have transition

• inside App.js where routing is done.

```
import {Route, Switch, useLocation} from "react-router-dom";
import {AnimatePresence} from "framer-motion";
const location = useLocation();
<AnimatePresence>
   <Switch location = {location} key = {location.key}>
      <Route path = "/base">
          <Base />
      </Route>
      <Route path = "/toppings">
          <Toppings pizza = {pizza} />
      </Route>
   </Switch>
<AnimatePresence>
```

in Base, Toppings etc components add exit

now must have to apply exit to all. else they will be shown at one page (can't be removed)

If no routing exit is applied transition ~~Routing~~ will occur

in eg. <Base>, < Topping >

in Base . js

```
const containerVariants = {
    hidden : {
        opacity : 0,
          x : '100vw'
    },
    visible : {
        opacity : 1,
        x : 0,
        transition : {
            type : "spring",
            delay : 0.5 }
        },
        exit : {
          x : '-100vw',
          transition : { ease : 'easeInOut' }
    }
}
```

```
<motion.div
    variants = { containerVariants }
    initial = "hidden"
    animate = "visible"
    exit = "exit"
>
```

Svg animation

```
const svgVariants = {
  hidden : { rotate : -180 },
  visible : {
     rotate : 0,
     transition : { duration : 1 }
  }
}
const pathVariants = {
  hidden : {
    opacity : 0,
    pathLength : 0
  },
  visible : {
    opacity : 1,
    pathLength : 1,
    transition : {
      duration : 2,
      ease : "easeInOut"
    }
  }
}
```

```
<motion.svg className = "pizza-svg"
  xmlns = "http://www.w3.org/2000/svg"
  viewBox = "0 0 100 100"
  variants = { svgVariants }
  initial = "hidden"
  animate = "visible"
>
 <motion.path
   fill = "none"
   d = "M40 40 L80 C80 40 80 80 40 80 C40
      80 0 80 0 40 00 40 00 40 0Z'
   variants = { pathVariants }
 />
 <motion.path
   variants = { pathVariants }
   fill = "none"
   d = 'M50 30 L50 -10 C50 -10 90 -10 90 30
      Z"
  />
</motion.svg >
```

• Making a Loader

(ball bouncing)

```
import React from "react";
import {motion} from "framer-motion";

Const loaderVariants ={
    animateOne : {
        x: [-20,20],
        y : [0,-30],
        transition : {
            x : {
                yoyo : Infinity,
                duration :0.5
            },
            y : {
                yoyo : Infinity,
                duration : 0.25,
                ease : "easeOut"
            }
        }
    }
}

Const Loader = () => {
    return (
        <>
        <motion.div className ="loader"
        variants ={loaderVariants}
        animate ="animationOne"
        >
        </motion.div>
        </>
    )
}

export default Loader;
```

in index.css

(no need to call, already applied to all elements since called in index.html)

```
.loader {
    width :10px;
    height : 10px;
    margin : 40px auto;
    border-radius : 50%;
    background : #fff;
}
```

Use Cycle hooks
(to switch betn animations)

• import { motion , useCycle } from "framer-motion" ;

```
Const Loader Variants = {
    animation One : {
        - - -
    }
    animation Two : {
        - - -
    }
}
```

→ const [animation , cycleAnimation] = useCycle ("animationOne", "animationTwo")

```
    <- ——    animate = { animation }
  &  <div onClick ={()=> cycle Animation () } > Cycle Loader </div>
  &
```

dragging items & wrap Op

eg drag logo

```
   <motion.div className ="logo"
     drag
     dragContraints ={{ left :0 , top:0 , right:0 , bottom:0 }}
     drag Elastic ={0.7}
   >
     - -
     — —
   </motion.div>
```

↘ more the value easier to drag.

↖ brings back to this position