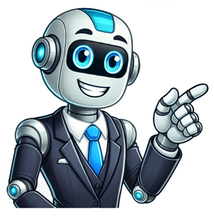
Click Here



A customizable and reusable chat component built with Next.js 25 January 2024 React developer portfolio template use framer motion to animate components 12 December 2023 Amazing Button Click Animation With Framer Motion's New Animation Sequences 08 May 2023 AI-Powered 3D Website with AI (DALL-E) on React, Framer Motion, ThreeJS and TailwindCSS 17 April 2023 Customizable 3D text hover animation using Framer Motion 06 February 2023 A macOS desktop clone built with TypeScript React, SASS and Framer Motion 26 December 2022 A web-based app built on React, utilising custom Framer Motion animations, ParticlesJS 09 September 2022 Facebook Reaction Animation made with React JS and FramerMotion 15 July 2022 A movie application with React JS and Framer Motion 03 July 2022 Personal Portfolio build using Next js, Three js, and Framer Motion 26 June 2022 Game Store using RAWG API and React 19 June 2022 CSS Generator Tools Built with React and Framer Motion 22 April 2022 A Stunning portfolio build with React JS and framer-motion for cool transitions and animation 16 April 2022 Dots game Built With React and Framer Motion. 24 March 2022 "Rock Paper Scissors" game built with React.js and framer motion. 23 March 2022 Time Counter project Using React and Framer Motion 17 February 2022 Create framer-motion Filter Animation using React.js 12 February 2022 Stock metrics - A web(mobile-focused) app for checking the companies listed on the stock exchange and see their live metrics 04 February 2022 Typescript Next.js Chakra-UI Framer-Motion Starter Kit Start with a powerful template ⚡ Table of contents Getting started Paths & Files Useful dependencies (Important) Components Internationalization Theme Framer Motion SEO See a DEMO here. 26 January 2022 A pretty cool country selector using TailwindCSS and Framer Motion. 13 September 2021 Animations are a powerful tool to enhance user experience, and when it comes to creating animations in React, Framer Motion is a popular library of choice. In this blog post, we'll explore 5 cool animations you can create in React using Framer Motion. You can also learn more complex animations with Framer Motion in this article!. Framer Motion is a versatile animation library for React, known for its simplicity and powerful animations. If you're looking for an alternative, you can explore React Spring, which I've covered in my article here. Now, let's dive into five cool animations you can create in React using Framer Motion! 1. Button Tap Animation This is a really simple yet fun effect to add to your buttons. It adds a fun touch to your buttons by scaling them when tapped. To add this, we can make use of the whileTap property. We can scale the button up or down within this property: import { motion } from "framer-motion"; function App() { return (Click me!); } export default App; The result will look somewhat like this: Prerendered Text typing animations, created with Framer Motion, add a playful touch to engage users, useful for loading messages and chat simulations. To achieve this effect, we will take a sentence we want to display and convert it into an array of words. Next, we will map this array out and add an animation and transition on the opacity, as well as a delay that creates the typing effect. import "./styles.css"; import { motion } from "framer-motion"; function App() { const text = "Framer Motion is a really cool tool".split(" "); return ({text.map((el, i) => ({el}{" "})}}); } export default App; Prerendered Stagger animations, as demonstrated in dropdowns and navigation menus, involve animating elements in a sequential manner, each appearing one after the other, with a subtle delay distinguishing their appearances. You can create much more complex effects using this method as opposed to the ones mentioned above. In this example, we will animate in some menu items when a menu button is clicked. We will use the stagger function alongside motion and useAnimate. Learn more about useAnimate here. import { useAnimate, stagger, motion } from "framer-motion"; Start by creating an array that holds the list of items to be displayed in the menu. To achieve this effect, the list items are initially mapped within a element. When the button is clicked, the is first made visible. Then, the list items will animate in a staggered fashion one by one. We will utilize the useAnimate hook to generate the animations for both the and elements. These animations will be triggered by a change in state, which can be toggled using the button. import "./styles.css"; import { useState, useEffect } from "react"; import { useAnimate, stagger, motion } from "framer-motion"; export default function App() { const [open, setOpen] = useState(false); const [scope, animate] = useAnimate(); const items = ["Item 1", "Item 2", "Item 3", "Item 4"]; // the stagger effect const staggerList = stagger(0.1, { startDelay: 0.25 }); // create the animations that will be applied // whenever the open state is toggled useEffect(() => { animate("ul", { width: open ? 150 : 0, height: open ? 200 : 0, opacity: open ? 1 : 0 }, { type: "spring", bounce: 0, duration: 0.4 }); animate("li", open ? { opacity: 1, scale: 1, x: 0 } : { opacity: 0, scale: 0.3, x: -50 }, { duration: 0.2, delay: open ? staggerList : 0 }); }, [open]); return (setOpen(!open)} whileTap={{ scale: 0.95 }}> Menu {items.map((item, index) => ({item})}}); } Prerendered This effect is commonly used for displaying numbers and statistics and is often used in countdown timers. To implement this effect, we'll utilize MotionValues to monitor the state and velocity of the values, and we'll employ the animate function to specify the animation duration. import { motion, useMotionValue, animate } from "framer-motion"; We will pass it an initial state or value which will be 0. Then we will count up to 50. Keep in mind that you need to round the count to make sure no decimal points are shown. For this, useTransform will be used. import "./styles.css"; import { motion, useMotionValue, useTransform, animate } from "framer-motion"; import { useEffect } from "react"; export default function App() { const count = useMotionValue(0); const rounded = useTransform(count, Math.round); useEffect(() => { const animation = animate(count, 50, { duration: 2 }); return animation.stop; }, []); return {rounded}; } In the code above, animation.stop is being used to stop the animation when the component is unmounted. This ensures that the animation is terminated, and any resources associated with the animation are released. Prerendered Scroll reveal animations are a great way to make websites engaging and appealing to users. For instance, you can create effects like fading an element in or smoothly sliding it into view as it becomes visible on the screen. To create this effect, we can use the whileInView and viewport props. Here's an example of how to use these props to achieve a scroll reveal effect: We can add animations and effects within the initial and whileInView props. Variants can also be used for this purpose. In the example below, we are going to slide in some elements when they come into view: import React from "react"; import { motion } from "framer-motion"; function Card({ text, index }) { return ({text}); } export default Card; You can set viewport={{ once: true }} to false if you want the effect to occur every time the element is scrolled into or out of view. Prerendered Thanks for reading! AboutA simple and powerful JavaScript animation library7,625,893Weekly DownloadsExternal Linksgithub.com/motiondivision/motion#readmegithub.com/motiondivision/motiongithub.com/motiondivision/motion/issuesCollaboratorsProductCodingPrototypingKnowledge SharingFeedbackWhat’s NewExploreFeatured SandboxesSearch SandboxesForIndividualsTeamsEnterpriseAboutPricingCompanyBlogPodcastsCareersTerms Of UsePrivacy PolicySupportDocumentationContact SupportStatusCopyright © 2025 CodeSandbox BV When a tool like Framer Motion is available, why write many lines of CSS code to create an animation? Web applications look more exciting and beautiful when they are animated.Many developers avoid adding animation to their web applications since animation codes can be rigorous to write. However, with Framer Motion, you can execute animations with only a few lines of code. This external library in React.js makes animation incredibly simple, allowing the developer to concentrate on other aspects of the project.In this article, we'll examine Framer Motion's functionality, installation process, and usability by using it to animate text and images.Steps we'll cover:PrerequisitesTo follow along with this tutorial, you need to have:A clear understanding of basic frontend technologies, i.e., HTML, CSS, and JavaScript.Knowledge of React.JSNode Js installed.What is Framer Motion?Framer motion is a powerful production-ready library from Framer that can create different animation styles in the DOM as elements. Framer Motion is an excellent alternative to making complex CSS animations. To use Framer Motion, you need to install the library and integrate it into your React project.Getting started with Framer Motion in ReactReact requires that we install any external libraries using the npm package before utilizing them. Install Framer motion with the following command.npm install framer-motionThe next step is to import the Framer Motion component you wish to use into your React app.We are going to break this article into two sections first, the text animation, and second, the image animation. In the end, we will combine both into one React application.Getting started with text animationIn this section, we'll demonstrate how to use Framer Motion when animating text using the motion components.Add the following code to your App.js file.App.jsimport { motion } from "framer-motion";function App() { return (Animation made easy with Framer Motion);}export default App;We imported Framer Motion's motion component. Look closely at the h1 element you will notice how it differs from the standard h1 element.The first step in using Framer Motion to animate is to use the motion component, which can be used with any standard HTML element.We will give our text a cool entry animation to show gradually when the page reloads by utilizing the motion component in our h1 and a Framer Motion animate prop to accomplish the animation.With this approach, we can animate our text to fulfill our desires. First, we assign a position to our text, indicating where the animation will take place. Then we will animate our text on the x-axis. You can set x to have a value of [0, 150, 50].Next, let's give it an opacity of 1.We need to assign a scale to the cool entering animation. We don't want anything wild. We can set the scale to 1. So when the text animation reaches scale 1, it stops increasing in size.Finally, we use the Framer Motion transition prop because every effective animation requires a transition process. We will specify the duration, delay, and ease of our text animation inside this prop. Give it a duration of 5, a delay of 3, and an ease of [0.5, 0.71, 1, 1.01].The easing shows the scale at which the text animation enters before going to its original scale.However, we don't want to stop with the cool entrance in our text animation. Let's add a beautiful hover effect to our text. When your mouse hovers over it, it will appear to bounce or grow bigger in size.To accomplish this, we'll use the whileHover prop and scale it to 1.2.App.jsfunction App() { return (Animation made easy with Framer Motion);}In this section, we'll demonstrate how to use Framer Motion when animating images.First, you need to create an image folder inside the "src" folder to keep your images. For the sake of this project, I used some random images. Feel free to use any image of your choice.Create an “image.js” file inside the “src” folder and add the following code inside:src/images.jsimport fox from "./images/pexels-fox-1071249.jpg";import jill from "./images/jill.jpg";import cha1 from "./images/cha1.jpg";import cha2 from "./images/cha2.jpg";export default [fox, jill, cha1, cha2];We are going to import all our images into the image.js file mentioned above before using them in our App.js file. This will help us avoid redundant importation where we will use these images.Add the following code to your App.css file.App.css.item { min-height: 10rem; min-width: 30rem; padding: 5px;}.item img { width: 60%; height: 50%; border-radius: 1rem; pointer-events: none; padding: 15%;}.inner-carousel { display: flex; background: rgb(76, 76, 76); height: 23rem;}.carousel { cursor: grab; overflow: hidden; background: rgb(215, 216, 215); height: 23rem;}Change App.js file with the following code:App.jsimport { motion } from "framer-motion";import "./App.css";import images from "./image";function App() { return ({images.map((image) => { return ( ); })});}export default App;The image animation will take the form of a carousel with spinning effects each time the page is refreshed, hover effects each time the mouse is moved close to them, and tap effects each time the image is touched.For the carousel, create three containers, one for the outer carousel, one for the inner carousel, and one for the images. All three containers should be created using Framer Motion motion components, one inside the other.The inner carousel will perform the spinning and sliding effects. The framer motion animate prop will be called in at this point, scaled to [1,1,1,1,1], and rotated to [0,30, 60, 240, 360].This will give it a unique spinning effect, as you can see below:We'll use dragConstraint prop and framer motion drag to give it a carousel-like feel. “X” will be passed as a parameter in the drag prop because we need our carousel to move along the x-axis.To set a stopper in place, utilize the dragConstraint prop. For our carousel to travel to the left and then return to its previous location, we set the dragConstraint:{ right: 0, left: -1100 }.It appears too boring to move with just the drag prop and dragConstraint props; instead, let's add a bounce-like effect at each end of the drag. This can be achieved with the drag transition props from Framer Motion.We'll take advantage of the "bounceStiffner" and "bounceDamping" for drag transition attributes. Give the bounceStiffner and bounceDamping values of 600 and 8, respectively. Finally, we will navigate through our imported images in this container and add each image back to the carousel.Add highlighted code to App.jsfunction App() { return (...);}The last div in the carousel will house the returned images. To create the appropriate animation inside this div, we will use the framer motions whileHover, whileTap, and a transition prop.Use the framer motion whileHover with a scale of 1.2 to give our image a nice hover effect. When your mouse is over an image, it will only scale up to this maximum.Then, by giving the whileTap prop a scale of 2, we will give our images a tap effect so that they bounce when we tap on them. We must define a transition duration so that hovering and tapping become more engaging. Let's choose 3 seconds as the duration.App.jsfunction App() { return ({images.map((image) => { return ( ); })});}Contrary to ordinary CSS, we can see that framer motion enabled the above animation to be created quickly and with a few lines of code. Some programmers like to use their own animation with raw CSS, but if you're like me and prefer a simpler solution, you can use Framer Motion for your animation.You can access the app hereConclusionThis article taught us how to use Framer Motion to create a simple text and image animation for a React application. Compared to regular CSS, we saw how easy it is to implement these animations with a few lines of code. We also saw how to use some Framer Motion components and props. Framer Motion has other powerful features we didn't cover in this article. You should check them out. I hope this article becomes valuable to you. SiteAboutBlogDocsExamplesMotion+SponsorSocialDiscordGitHubX/TwitterBluesky Previously, I covered React Spring, a spring-physics based animation library. We made some easy to make toggle and then a somewhat complex animation/transition on hover. Well, that was all good and cool until I came up with Framer Motion! This is a really great library to make your prototypes come to life with double the ease of use from React Spring. Let's take a look at what it has to offer and why you should be using it in your next React project. Framer Motion is a production-ready motion library for React. It utilizes the power of the Framer prototyping tool and is fully open-source. There are some out-of-the-box features or selling points: My favorite one is the Variants, click below to interact: Also, It uses server-side rendering. It has CSS variables support. You can unmount animations easily. It has great accessibility options. You can handoff designs from Framer to Framer Motion. Before we do any of this we need to understand some basics, most probably its API. The Motion API This is what the heart of Framer Motion is. It provides us with a variety of options including those which you saw in the above points. ➡ The motion component. It's a React component built-into the library and is available for almost any of the HTML or SVG element you'll be using in your UI. These are DOM optimized for 60fps animation and gesture support. What's good is that we can easily convert a static HTML/SVG element to a motion component. For example, if we have a usual div, then just add motion. in front of that HTML tag and you have a motion component! Hence becomes . It allows you to: Declaratively or imperatively animate components. Animate throughout React trees via variants. Respond to gestures with animations. Add drag, pan, hover, and tap gestures. Here's an example: ➡ The animation prop. As you saw in the above code snippet, motion components are animated via the animate prop. When any value in animate changes, the component will automatically animate to the updated target. If you use x or scale values then they will be animated via a spring simulation. Whereas values like opacity or color will be animated with a tween. You can set different types of animation by passing a transition prop. Here's an example: ➡ Gestures. All the motion components can detect hover, tap, pan, and drag gestures. Each of these has event listeners you can attach. Two of the commonly used gesture props provides by the motion component are whileHover and whileTap. Here's an example: motion.button whileHover={{ scale: 1.2, transition: { duration: 1 }, }} whileTap={{ scale: 0.9 }} /> ➡ MotionValue. This is used to track the state and velocity of animating values. These are created automatically. But for advanced use-cases, it is possible to create them manually. It allows you to: Set and get the state. Chain MotionValues via the useTransform hook. Pass to multiple components to synchronize motion across them. Here's an example: export function MyComponent() { const x = useMotionValue(0) return } Yes! We're taking the boilerplate interface that comes when we create a React app and adding a little bit of interaction fun to it. As you can see, these are a few of the things happening: First, when the page loads, it fades in. The only animation happening. Next comes the interactions. When the React logo is clicked, we see it acts as a button. It pushes back on mouse press and when released, it comes to its normal state. We can also click and drag the React logo horizontally and it keeps fading as it moves away from the center. When hovered, the text below the logo scales up. To move the text from its position horizontally, we have a slider from which it can be controlled. Finally, we can fade in and out the same text with the toggle button. There's so much to cover. Let's dive straight into the development! Step 1: Create a React project and add Framer Motion After you're done with creating a React app, simply install the Framer Motion dependency with the following command: Step 2: Import the library and convert the elements! For this demo, we need to import these three API functions: motion, useMotionValue, useTransform. import { motion, useMotionValue, useTransform } from 'framer-motion'; We already talked about the first two. Now the useTransform is a hook through which we can pass the latest MotionValue through an update function that takes the latest parent value and transforms it. After the import, we need to change some of the default HTML tags that come with React boilerplate to the new Framer Motion ones. Do the following changes in App.js: Parent element to . Wrap the React logo tag inside a newly created . The tag to . Add a new element which will be our range slider with min and max values as -100 and 100 respectively. Next to this create a new with the text as "Toggle Fade". Here's what we did so far: Edit src/App.js and save to reload. Toggle Fade Nothing will happen yet as we haven't written any props and the Motion code doesn't have its properties to work on. Step 3: Add the animations and transitions! The page fade animation: For the initial fade animation, we use the initial, animate, and transition properties over the new motion.div. The initial initializes a value for the animate. The animate has the actual values to animate to The transition is used to add a default transition of one frame change to another. As we need a simple fade animation where the animation takes place half a second, we give the following properties to the motion.div: initial={{ opacity: 0 }} animate={{ opacity: 1 }} transition={{ duration: 0.5 }} Now the entire page fades! The tap and drag interaction on React logo: This is achieved by the whileTap and drag helper animation props on the motion component. The whileTap animates while the element is pressed/clicked. The drag enables the dragging gesture of an element and is set to false by default. So, on tapping the logo, we first need it to scale a bit, hence we add the scale property to the whileTap prop and for the drag, we need to pass on which axis dragging is to be done. Here, we do it horizontally so it's the x axis. To achieve the actual value of the x we passed on the drag prop, we'll be using the useMotionValue hook which tracks the state and the velocity of the dragged element. Initially, we don't want the drag to enable, hence we pass in 0. As for defining the coordinates of how much drag is to be done, the useTransform hook will help us. Through this, we can pass in the latest value of the x axis. It can be any number you like depending on the expensiveness of the drag you want to achieve. const x = useMotionValue(0); const opacity = useTransform(x, [-200, 0, 200], [0, 1, 0]); Now, for both of these to work, we need to pass the style prop which takes in the different constants we provided above. Hence out image drag and tap interaction code looks like this: const x = useMotionValue(0); const opacity = useTransform(x, [-200, 0, 200], [0, 1, 0]); . . . . . Now this interaction works! Interactions on the text: We have a scale on hover and tap, a drag on moving the slider, and finally a fade toggle using a button to finish off. The scale is done exactly like we did the tap, it's just that here the interactions are both tap and hover so for the new hover interaction we use the whileHover prop. The x variable is used for the horizontal drag as we need the same values. Now, to constraint its values we can fine-tune it using the dragConstraints prop which allows us to pass the left and right constraints of the drag gesture. For the final interactions, we need to use the useState React Hook as we are changing the drag and fade states of the text. Hence, we define the following two variables for states: const [value, setValue] = useState(0); const [toggle, setToggle] = useState(1); On the element we created at the beginning, its onChange event uses the setValue() method from the useState Hook and we pass in the current value chosen by the user when they drag the slider. A similar event is fired on the 's onClick but here we're toggling by interchanging the states from 0 to 1 or vice-versa. For the actual fade to kick in, we simply get the value from the created state (adding the 'px' string so that it works as an actual pixel unit) and use the opacity value equal to the toggle we created. const [value, setValue] = useState(0); const [toggle, setToggle] = useState(1); . . . Edit src/App.js and save to reload setValue(e.target.value)} /> setToggle((prevValue) => (prevValue ? 0 : 1))} className='toggle-button'>Toggle Fade . . . The button styles are simple in CSS to look better from the default one: .toggle-button { margin-top: 1.5em; width: 10em; border: 0; outline: none; padding: 1em; border-radius: 10em; font-weight: bold; } .toggle-button:hover { color: #282c34; background-color: #61dafb; } And now our final interaction also works! If you're not impressed by this library then you can check out React Spring. I wrote a tutorial regarding the same: Thanks for reading, I appreciate it! Have a good day. (✿◕‿◕✿) And if you want to impress developers, you can try the new open-source Windows Terminal. 😎 Download here: source: pic.twitter.com/WQfKOWTjfa Discover inspiring Framer motion examples to create smooth, realistic animations for your web projects. Try our web design solution today. Animations can make or break a web design project. If your animations could be smoother or more natural, users might feel like they are navigating a broken website. However, getting animations just right can be a challenge. Fortunately, Framer motion makes creating beautiful, realistic animations for React projects easy. In this blog, we’ll explore some Framer motion examples to inspire your own animations. We’ll also introduce you to NUMI’s Framer design solution to help you start quickly.What is Framer Motion?Framer motion is an open-source animation library built by Framer. This powerful tool adds slick, professional animations to web projects, especially those built with React. One of its standout features is its ability to simplify the animation process. With Framer Motion, developers can build smooth, visually stunning animations without writing a lot of CSS. As a React-based library, Framer Motion provides a simple and intuitive API, making it easy to create animations. Creating Dynamic Web ExperiencesWhether trying to animate a button on click, create a draggable element, or handle complex gesture-based interactions, Framer motion has you covered, all with just JavaScript. This eliminates many of the headaches associated with traditional CSS-based animations, which often require multiple lines of code and a lot of trial and error. Framer Motion ConnectionWith its easy integration, seamless connection with Framer design tools, and thriving community, Framer motion makes creating dynamic, interactive web experiences faster, easier, and much more fun. Whether you’re animating UI elements or adding complex interactions, Framer motion delivers smooth, production-ready results with minimal hassle.5 Best Features of Framer Motion1. Create Animations Easily with Framer MotionFramer motion simplifies creating animations with React. The library allows developers to define their animations using the animate prop. When values within this prop change, the corresponding motion component automatically animates to the new state. This feature streamlines the animation process, enabling developers to focus on their designs rather than on implementation details. ```javascript```2. React to User Gestures with Framer MotionFramer motion supports various gestures that can trigger animations responding to user interactions. For instance, you can use props like whileHover, whileTap, and whileDrag to define how a component should animate in response to these gestures. ```javascript```3. Use Variants To Manage Animation StatesVariants allow you to define multiple animation states for components and orchestrate animations across entire sub-trees with a single animate prop. This declarative approach makes managing complex animations and transitions between different states easy. ```javascriptconst variants = { open: { opacity: 1, x: 0 }, closed: { opacity: 0, x: "-100%" },};```4. Server-Side Rendering SupportFramer motion supports server-side rendering (SSR), ensuring that the animated state of components is rendered on the server side. This feature prevents flashes of re-styled content after JavaScript loads, providing a smoother user experience.5. Use Motion Values for Fine-Grained ControlMotion Values create a declarative and reactive chain of values that can be used to update the results of animations or gestures. They allow for fine-grained control over animations and can be used with other features like scroll-linked animations. ```javascriptconst { scrollYProgress } = useScroll();```Additional FeaturesFramer motion combines these features to provide a comprehensive solution for creating rich, interactive animations in React applications, making it a popular choice among developers looking to enhance user experiences with minimal effort.Hardware-Accelerated Animations: Framer motion intelligently decides when to run animations on the GPU, ensuring smoother performance. Exit Animations: Components can have exit animations that play when removed from the DOM.Keyframes: Support for animating through an array of values over time. Scroll-Linked Animations: Use scroll position to drive animations, creating effects like parallax scrolling.Your One-Stop Shop for Startup Design NeedsNUMI is a framer web design agency that world-class framer developers and product designers power. Backed by Y Combinator, NUMI handles all of your startup's sourcing/vetting/hiring needs for design. Our fabuolous design team ensures all your design work is done well. NUMI helps with:Product designWeb designFramer developmentWebflow developmentMobile designPrototypingUX designAll of your startup's design needs! Subscribe to a guild of world-class designers ready to embed on your team today. Schedule a call with us today to learn more!Related Reading8 Benefits Of Adding Animations To Your Application1. Increased User EngagementAnimation helps capture and retain user attention by creating visually pleasing and interactive experiences. Transitions and animations can make the app feel more intuitive and responsive, thus encouraging users to spend more time using the app.2. Improved User NavigationBy utilizing animation, developers can guide users through the app's interface, making navigating between different sections and features easier. Smooth transitions and subtle animations provide visual cues, giving users a sense of direction and making the overall experience seamless and enjoyable.3. It Makes Your Website Stand OutMany websites still have static content, and adding animation to yours can make it stand out. While there are different ways to use animation to give your website a unique look, if you want your website to deliver the desired results, you also need excellent web design skills and innovative content.4. Mobile-First ApproachIf you add animation, remember that the mobile-first approach has become the industry standard in:According to Statista, more than half of all global website traffic already comes from mobile devices, so mobile-friendliness should be considered when adding animated content and features to your website.5. People Remember Animated Messages LongerAny message you convey through visual and audio presentation will last longer in the minds of your audience. A short animated video introducing your brand or its products/services can significantly impact your website's traffic. Various studies have indicated that people are more likely to remember information they got through audio-visual formats than through reading alone, and this is plain to see in the learning pyramid. From the web design perspective, animation can help you communicate better and sell more products and services through your website.6. It Provides Great FlexibilityAnimation is flexible and can help you introduce several products or present much information in a short and simple video.Animated IntroSome visitors need more time to browse through entire websites, looking for specific:ProductsServicesInformationUsing an animated video can help you introduce all the services at once.7. It is Good for SEOAnimation can have a positive effect on search engine optimization (SEO) by:Improving user experienceIncreasing engagement timeThe longer users spend on your website, the better its search engine ranking. Animation can help:Decrease bounce ratesCreate better backlink opportunitiesIncrease social engagementEnhance mobile users' experience8. It Makes Your Message Easier to Convey It’s common for consumers to look for different types of information before making a purchase. The process of researching your company and its products/services can be time-consuming for new customers who know little to nothing about your business.Animation is an easy way to convey your message in a short span of time while allowing you to include plenty of information.Animated PersuasionWhile your animated video can attract the attention of prospective customers, it can also help convince them to perform the desired action, be it making a purchase or subscribing to a service.6 Components Of React Framer Motion1. Motion Components: The Core of Framer MotionMotion components form the foundation of Framer motion. They are simple replacements for standard HTML and SVG elements that enable you to add animations to your React applications. Every HTML and SVG element has a corresponding motion component. For example: for a div element for an SVG circleAnimated ComponentsThese components function like their counterparts but come with additional props that enable animations, gestures, and more. They allow developers to: Animate properties like position, scale, and opacity. Add gestures such as drag, pan, hover, and tap. Respond to viewport changes and deep animations through variants.2. AnimatePresence: Exit Transitions Made EasyAnimatePresence is a key component for managing exit animations for components that are conditionally rendered. It detects when direct child components are removed from the React tree and animates them accordingly. This is particularly useful for smooth transitions when elements enter or leave the DOM.3. LayoutGroup: Grouping For Smoother Layout AnimationsLayoutGroup allows you to group motion components that should perform layout animations together. When motion components with a layout prop are rendered, they automatically detect and animate layout changes, providing a cohesive transition effect across grouped elements.4. LazyMotion: Boost Performance By Reducing Bundle SizeLazyMotion helps reduce bundle size by allowing asynchronous loading of motion features. This means that only the necessary parts of the library are loaded when needed, optimizing performance without sacrificing functionality.5. MotionConfig: Global Configurations for Motion ComponentsMotionConfig sets global configuration options for all child motion components. This can include default transition settings or other shared properties that apply to multiple components within a specific context.6. Reorder: Drag-and-Drop Animations Made EasyThe Reorder component facilitates drag-to-reorder functionality for lists or groups of items. This makes it easy to create interactive interfaces where users can rearrange elements, such as tabs or to-do items, through simple drag-and-drop interactions.Your One-Stop Shop for Startup Design NeedsNUMI is a framer web design agency that world-class framer developers and product designers power. Backed by Y Combinator, NUMI handles all of your startup's sourcing/vetting/hiring needs for design. Our fabuolous design team ensures all your design work is done well. NUMI helps with:Product designWeb designFramer developmentWebflow developmentMobile designPrototypingUX designAll of your startup's design needs! Subscribe to a guild of world-class designers ready to embed on your team today. Schedule a call with us today to learn more! Framer Motion Examples: Animating In React Using Framer MotionOne of Framer motion's standout features is the ability to create fluid page transitions. Using the AnimatePresence component, you can manage the entrance and exit of components as pages change. You might: Fade the outgoing page to transparent, then have the new page fade in. Create a slide transition, where pages move horizontally or vertically off-screen as new ones slide in. Add scaling effects, where the page content grows or shrinks during transitions. These transitions enhance user flow, making your website feel like a cohesive experience rather than a series of disconnected pages.Hover Effects: Add Interactivity to Your Website with Framer MotionHover animations are a simple but effective way to add interactivity to your elements. With Framer motion’s while-over property, you can animate buttons, images, or icons when users hover over them: Enlarge a button when hovered to make it feel more clickable. Rotate or tilt icons to make them visually engaging. Change colors smoothly to indicate interactivity or call-to-action. These subtle hover effects make your site more dynamic and inviting for users.Drag-and-Drop Interactions: Use Framer Motion to Create Interactive ExperiencesFramer motion allows you to easily add drag-and-drop functionality to any element using the drag property. Here’s how you can use this feature: Create draggable cards in a gallery, allowing users to rearrange them. Build interactive sliders where users drag handles to adjust values. Design custom file-upload areas where users drag files to a designated zone to upload. You can also restrict dragging to the X or Y axis using drag="x" or drag="y", providing complete control over the interaction.Animated Modals and Pop-ups: Use Framer Motion to Enhance Your UIAnimated modals or pop-ups can make your UI more engaging and visually appealing. Using Framer motion: Modals can fade in and scale up from the center, providing a smooth transition into focus. Exit animations can slide the modal off-screen or fade away when users close it. Add dimmed background effects that fade alongside the modal to enhance focus on the modal content. These animations ensure a clean and professional presentation of important information or interactions.Scroll-based Animations: Use Framer Motion to Create Engaging Storytelling ExperiencesScroll-triggered animations can add depth and interactivity to a website by animating elements as the user scrolls down the page: Elements like images or text can fade or slide into view as they become visible, creating engaging storytelling experiences on long-scrolling pages.Use parallax effects where background images move slower than foreground elements, giving a sense of depth. Trigger animations on scroll by detecting when specific sections enter the viewport and animating elements within those sections.Using useInView or similar hooks lets you finely control when and how elements animate as users scroll through content.Staggered Animations: Create Smooth User Experiences with Framer MotionStaggered animations bring elements to life sequentially, and Framer motion makes this effortless using staggerChildren: When displaying a list of items (like product cards), each item can fade or slide into view one after another, creating a cascading effect. Animate navigation links, where each link appears in a staggered sequence when the menu opens. Use staggered animations for onboarding steps or feature lists, guiding users' attention through steps. This technique makes the user experience smoother and draws attention to content in a visually pleasing way.Complex Keyframe Animations: Use Framer Motion to Create Intricate EffectsFramer motion supports keyframe animations, allowing you to animate through a sequence of values. This is useful for creating more intricate effects: A bouncing ball effect, an element moves from one side of the screen to another, scaling down and back up as it bounces. Color-changing animations, where an element transitions between multiple colors in a loop (e.g., for loading indicators or attention-grabbing buttons). Rotating elements through different angles and applying easing functions to give it a smooth, lifelike motion. Keyframe animations let you specify multiple stages of an animation, providing greater control over how elements behave.Gesture-controlled Animations: Make Your Web Applications More Interactive with Framer MotionWith Framer motion, you can build animations that respond to user gestures, such as taps, clicks, or drags: Use the whileTap property to shrink buttons slightly when clicked, giving users immediate feedback. Create interactive image galleries where users can drag to navigate between images, swipe, or pinch to zoom. Build animated sliders that respond to dragging, making UI components more tactile and responsive. Gesture-controlled animations make your web applications feel more interactive and intuitive, especially on touch-enabled devices.Animated Headers and Footers: Improve Your Website’s Navigation with Framer MotionHeaders and footers don’t have to be static! Using Framer motion, you can animate these elements to enhance user experience: Create a sticky header that animates when scrolling (e.g., shrinking the header as users scroll down and expanding it when they scroll up). Animate a footer section that slides into view when users reach the end of a page, highlighting important links or calls to action. Reveal content in your footer with a smooth slide-up effect when users hover over specific sections. These animations make the navigation experience more engaging while ensuring critical elements like the header or footer remain functional and eye-catching.Hero Section Animations: Use Framer Motion to Make a Strong First ImpressionThe hero section is typically the first thing users see on a website, so animating it can make a solid first impression: You can have the hero text fade in or slide in from the side as the page loads, instantly drawing attention. Add animated background images or gradual color transitions to create a visually appealing entry. Have hero buttons or CTAs pulse or bounce, making them more prominent and encouraging clicks. Animating the hero section can captivate visitors from the start and set the tone for the rest of the site.Bonus Example: Loading AnimationsAdding a loading animation can improve perceived performance and keep users engaged while content loads:Use spinners, progress bars, or pulsing elements to indicate loading status. Animate your company logo or other branded elements to reinforce your brand identity during loading. These animations ensure that users remain engaged, even during moments of downtime.Getting Started With React Framer MotionTo get started with Framer Motion in a React project, first install the Framer-motion package from npm. Ensure that your project is using React version 16.8 or greater.```bashnpm install framer-motion```Import the Motion ComponentOnce installed, you can import the motion component into your React components:```javascriptimport { motion } from 'framer-motion';```Use Motion Components to Animate Your AppFramer motion provides motion components that are enhanced versions of standard HTML and SVG elements. You can replace primitive DOM elements (like , , etc.) with their motion counterparts, such as and . These components come with additional props for adding animations and gestures.Example: To animate a div element, you can use the animate prop:```javascript```Animate Elements with Framer MotionYou can animate elements using the animate prop, which accepts an object of CSS properties to animate. You can also use variant labels or imperative animation controls.Example:```javascript```Add Gestures to Your Animated ComponentsFramer motion supports various gestures like:You can use props such as whileHover, whileTap, and drag to define how your components should respond to user interactions.Example:```javascript```Use Variants for Coordinated AnimationsYou can define variants to manage animations across multiple components or sub-trees with a single animated prop, allowing for coordinated animations.Example:```javascriptconst variants = { hidden: { opacity: 0 }, visible: { opacity: 1 },};```Server-Side Rendering with Framer MotionFramer motion supports server-side rendering (SSR), ensuring that a component's animated state is rendered on the server to prevent flashes of re-styled content after JavaScript loads.Related Reading8 Best Practices for Animating With Framer Motion1. Feedback and Guidance Through AnimationUse subtle animations to provide real-time feedback. Animations can signal users about their input (such as a button press or form submission) to:Reduce user uncertaintyEnhance the interface's usabilityStrategic use of motion can guide users through actions or highlight changes in application states, improving the navigational flow.2. Purposeful MotionEvery design animation should have a clear purpose. Excessive or irrelevant motion can overwhelm users or detract from the user experience. Ensure animations are smooth, quick, and cohesive, enhancing functionality without causing distractions.3. Consistency Across the BoardConsistency in motion and animation ensures a coherent user experience across all aspects of a product. The uniform use of animation in design reinforces user familiarity and comfort with the interface, promoting a seamless interaction from start to finish.4. Accessibility ConsiderationsMake animations accessible by providing options to reduce or remove motion for users who are sensitive to it. This inclusivity ensures the design is user-friendly for a broader audience, meeting:Individual user needsPreferences5. Empirical ValidationObserving real interactions can reveal how effectively animations contribute, allowing for refinements that closely match user preferences and behaviors. 6. Adaptive Animations for Diverse PlatformsUpdate design animations to fit the specific characteristics of different platforms. Mobile interfaces require animations that are responsive to touch interactions and optimized for smaller screens.7. Early Integration in the Design ProcessIntegrate UX motion and animation design early in development to identify potential usability issues. Experimentation with various animation styles and speeds can help determine the most effective approach for the product’s target audience.8. Stay Updated with Animation Design TrendsNew trends and motion design technologies can inspire you to update animation applications. Advances in Framer enhance animations’ capabilities and open up new possibilities for creating engaging user experiences. Related ReadingSubscribe To a Guild of World Class Framer Developers with Our Framer Web Design Agency TodayFramer design is a powerful tool for building stunning, highly interactive websites and apps. NUMI is a dedicated framer design agency that can help your startup harness the full power of a framer for your next project. With our talented team of framer specialists, we can help you build a gorgeous, functional website or app that your users will love. We can also help you with ongoing needs or future iterations to ensure your site or app stays fresh and up-to-date. Animations play a crucial role in modern web applications, enhancing user experience by providing visual feedback and improving the overall aesthetics. In React, creating animations can be a complex task, but with the right tools, it becomes much more manageable. Framer Motion is one such tool that makes building animations in React straightforward and powerful. Framer Motion is a popular animation library for React that provides a simple and declarative API for creating animations. It offers a wide range of features, including support for keyframes, gestures, spring animations, and more. With Framer Motion, developers can create sophisticated animations with minimal code, making it a go-to choice for many React projects. In this comprehensive guide, we will explore how to build animations in React using Framer Motion, covering everything from basic animations to advanced techniques and performance optimization. Framer Motion is an open-source animation library for React developed by Framer. It provides a declarative syntax for creating animations and interactions, allowing developers to add complex animations to their applications with ease. Framer Motion leverages the power of the Framer animation engine to deliver smooth and performant animations. Key features of Framer Motion include: Declarative API: Simplifies the creation of animations with a clear and intuitive syntax. Powerful Animations: Supports a wide range of animations, including keyframes, springs, and transitions. Gestures and Interactions: Provides built-in support for gestures like dragging, tapping, and hovering. Animation Controls: Allows fine-grained control over animations with hooks like useAnimation. Performance Optimization: Optimizes animations for smooth performance, even on lower-end devices. With these features, Framer Motion is well-suited for building interactive and dynamic user interfaces in React applications. Before we start building animations with Framer Motion, we need to set up our development environment. This involves installing Node.js and npm, creating a React application using Create React App, and installing Framer Motion. Node.js is a JavaScript runtime that allows you to run JavaScript code outside of a browser. npm (Node Package Manager) is included with Node.js and is used to manage JavaScript packages. To install Node.js and npm, follow these steps: Download and install Node.js from the official website. Verify the installation by running the following commands in your terminal: This should display the version numbers of Node.js and npm, indicating that they have been installed correctly. Create React App is a command-line tool that sets up a new React project with a sensible default configuration. To create a new React application, open your terminal and run the following command: npx create-react-app framer-motion-example This command creates a new directory named framer-motion-example and installs all the necessary dependencies. Once the installation is complete, navigate into the project directory and start the development server: cd framer-motion-example npm start Your default browser should open a new tab displaying the React welcome page, indicating that your React application is up and running. To use Framer Motion in your React application, you need to install it using npm: npm install framer-motion This command will add