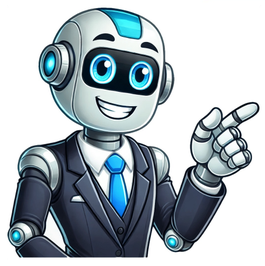
Continue



Originally posted by SimonMacIntyre July 3, 2024 We make use of Placeholder in forms, to show data related to the entity. For example let's say we have an EditUser page and form. We are using Placeholder as an example while viewing this page, to show some interesting live updating status about some thing based on this user. (Note: This is not suitable to be in a table/relation manager, we make heavy use of that too, but this specifically is a component we want to show right up in the form in a read-only fashion, but have it live-update). Any chance of this being a feature we can add some day? Ideally it would be as simple as... Placeholder::make('status')->poll(5)->... Page 2 Add issues to milestones to help organize your work for a particular release or project. Find and add issues with no milestones in this repo. You can’t perform that action at this time. Page 3 You can’t perform that action at this time. Placeholders can be understood as predefined “search and replace” templates that will be replaced with some actual values at execution time. They are usually used for MySQL queries. It’s worth noticing that placeholders are case-sensitive, so adding %var1% in the shortcode is not the same as %VAR1%, and it will not work. Currently, wpDataTables has 13 types of placeholders: %CURRENT_POST_ID% – this placeholder will be replaced with the ID of a currently active WordPress page or post. Provide a value here to be used for table generation. %CURRENT_USER_ID% – this placeholder will be replaced with the ID of a currently logged-in user (if the user is logged in). %CURRENT_USER_LOGIN% – this placeholder will be replaced with the currently logged-in username (if the user is logged in). %CURRENT_USER_EMAIL% – this placeholder will be replaced with the email of the currently logged-in user (if the user is logged in). %VAR1%, %VAR2%, %VAR3%, %VAR4%, %VAR5%, %VAR6%, %VAR7%, %VAR8%, %VAR9% – these are the ‘variable’ placeholders. You can provide a default value here that will be used for table generation, and when a different one is not defined in the shortcode. %WPDB% – this placeholder will be replaced with the current prefix of the WordPress database (defaults to “wp_”). %CURRENT_USER_FIRST_NAME% – this placeholder will be replaced with the First Name of the currently logged-in user. Provide a value here to be used for table generation. %CURRENT_USER_LAST_NAME% – this placeholder will be replaced with the Last Name of the currently logged-in user. Provide a value here to be used for table generation. %CURRENT_DATE% – this placeholder will be replaced with the current date. %CURRENT_DATETIME% – this placeholder will be replaced with the current date and time. %CURRENT_TIME% – this placeholder will be replaced with the current time. The placeholders’ configuration block appears when you choose all table types. As shown in the screenshot on the right, you can insert placeholders in the query as if they are some specific values. For example, you can provide the query like this: SELECT * FROM my_table WHERE my_field > %VAR1% AND my_field < %VAR2% AND user_id = %CURRENT_USER_ID% In this example, the query means: “show me everything from my_table, where my_field is greater than variable 1, but less than variable 2, and user_id is equal to the currently logged in users’ ID”. Before you try to save the table, you first need to see if this query returns anything. By default, the placeholders %VAR1% up to %VAR9% are equal to empty strings, and %CURRENT_USER_ID% is equal to the user that creates the table – typically the Admin user, most probably with ID=1. If you do not redefine the defaults, this query will be parsed like this on the MySQL side: SELECT * FROM my_table WHERE my_field > ” AND my_field < ” AND user_id = 1 Most likely, such a query would not return anything, and wpDataTables would say there’s no data to build the table upon. To avoid this, you will need to expand the placeholder configuration block, and define the default values for the variables: e.g., set %VAR1% equal to 0, and %VAR2% equal to 100. Also, you can redefine the value for %CURRENT_USER_ID%, e.g. if the table does not contain any values related to the admin’s ID, set it to “15”. The variable values are then saved in the table settings, and they will be always used as default if not overridden by shortcode parameters. Only the %CURRENT_USER_ID% is not saved, and the provided value is used only at the instant of the table generation, after which the actual user ID is always fetched. The query would then be parsed like this on MySQL side: SELECT * FROM my_table WHERE my_field > 0 AND my_field < 100 AND user_id = 15 If this query returns any data, it will successfully create a wpDataTable. Later, you can insert this wpDataTable’s shortcode in your posts or pages, and it will fetch the data for the current user. But you can also override the variable settings. Simply provide the variable values in the shortcode like this: [wpdatatable id=12 var1=150 var2=350] and the values from shortcode attributes would be applied to the variable and the query would be treated like: SELECT * FROM my_table WHERE my_field > 150 AND my_field < 350 AND user_id = (current_user_id) Using placeholders like this, you can use a single wpDataTable to produce many different output tables on different pages. Placeholders are also supported in the “Predefined Value” text input on the Filtering tab in column settings. You can define the placeholders %VAR1% up to %VAR9% in the “Predefined Value” input of the column, and then pass a value of the variable through a shortcode parameter where it will be used as the default value for advanced filter on the page, and for the editor input if the table is editable. If you are using String for Placeholder ‘variable’ %VAR1% should be surrounded with Single Quote (‘) in the query. In my post about React-Query for managing server state, I talked about the basic concepts for you to start working with React-query. In this post, I would like to talk more about useQuery hook. More specifically about polling, background data fetching, and data transformation. Polling is the process of repeatedly fetching data, automatically, from a remote data source at regular intervals. By default, useQuery does not fetch data automatically. It needs some sort of trigger. Suppose you are developing a real-time system where data is frequently changing, and you need to incorporate React Query in the front end. It’s evident that data needs to be fetched regularly from the backend to keep the front end in sync. To achieve this, polling is a viable solution. But, how are you going to implement it? This is where the refetchInterval property of the useQuery hook comes into play. By setting this property to a specified time interval in milliseconds, the useQuery hook will automatically refetch data from the backend at that interval, enabling real-time data updates in the front end. const { data } = useQuery({ queryKey: ['product'], queryFn: () => fetch(url).then(res => res.json()), refetchInterval:3000, }) Watch the video below to see how useQuery works with refetchInterval option While the refetchInterval property enabled data refetching , there is one limitation to this property. That is it will refetch data as long as the browser window is active. If the browser window is not active, useQuery will stop fetching data. What if you want to refech data even when the browser window is not active? useQuery offers another solution for this. That is refetchIntervalInBackground property. The correct default value for refetchIntervalInBackground in the useQuery hook is false, which means that by default, the hook will not refetch data in the background at the specified interval when the browser window is not active. If you want to change this behavior and have the hook continue to refetch data in the background, you can set refetchIntervalInBackground to true. The following video shows how to test this property. You can use the select option to transform or select a part of the data returned by the query function. In the select option, you can define a function to transform the returned data of the useQuery hook. But, this select function does not affect what gets stored in the query cache. In other words, when you use the select option, UseQuery will still fetch the data from the API using the queryKey and queryFn options that you specify. Once the data is fetched, the select function is applied to the data and the trans const { data } = useQuery({ queryKey: ['product'], queryFn: () => fetch(url).then(res => res.json()), select: (data) => data.map(e => e.name) }) In the above code, the select option changes what is returned to the Component. The component can now use only the name of the product. However, the cache has both the product id, name, and price. Note: Because the select option influences the returned data, you need might need to change how you access data inside the return statement in the React Component For example: Without select option: return({ data?.map((product, key) => { product.name })}) If you add the select option below select:(data) => data.map(element => element.name) Because the map function returns an array of names(of the products), you need to change the return a statement of the Component as below return({ data?.map((product, key) => { product })} However, this largely depends on the code of your application. For example, if I add the following function in the select option I do not need to do any changes inside the return statement. select:(data) => data.map(element => { return {name: element.name } }) In the above code snippet, the map function returns an array of objects with “name” as the key and the name of the product as the value. Therefore, I do not need to do any changes in my code in the return statement. React-Query’s useQuery hook provides several powerful options that can help you build flexible and efficient data-fetching solutions in your React applications. Whether you need to poll an API at a regular interval, fetch data in the background, or transform the data before it’s returned to the component, options like refetchInterval, refetchIntervalInBackground, and select make it easy to get the job done. By using these options in different ways, you can create data-fetching solutions that meet the specific needs of your application. Download the full code Note: In my demo project, I utilize a mock API I created with mockapi.io. However, I highly recommend creating your own API endpoint with mockapi.io. This is because it can be easier for you to do your experiment while you are learning. useQuery API reference Placeholder data allows a query to behave as if it already has data, similar to the initialData option, but the data is not persisted to the cache. This comes in handy for situations where you have enough partial (or fake) data to render the query successfully while the actual data is fetched in the background. Example: An individual blog post query could pull "preview" data from a parent list of blog posts that only include title and a small snippet of the post body. You would not want to persist this partial data to the query result of the individual query, but it is useful for showing the content layout as quickly as possible while the actual query finishes to fetch the entire object. There are a few ways to supply placeholder data for a query to the cache before you need it: const result = useQuery({ queryKey: ['todos'], queryFn: () => fetch('/todos'), placeholderData: placeholderTodos, }) const result = useQuery({ queryKey: ['todos'], queryFn: () => fetch('/todos'), placeholderData: placeholderTodos, }) In some circumstances, you may be able to provide the placeholder data for a query from the cached result of another. A good example of this would be searching the cached data from a blog post list query for a preview version of the post, then using that as the placeholder data for your individual post query: const result = useQuery({ queryKey: ['blogPost', blogPostId], queryFn: () => fetch(`/blogPosts/${blogPostId}`), placeholderData: () => { // Use the smaller/preview version of the blogPost from the 'blogPosts' // query as the placeholder data for this blogPost query return queryClient .getQueryData(['blogPosts']) ?.find((d) => d.id === blogPostId) }, }) const result = useQuery({ queryKey: ['blogPost', blogPostId], queryFn: () => fetch(`/blogPosts/${blogPostId}`), placeholderData: () => { // Use the smaller/preview version of the blogPost from the 'blogPosts' // query as the placeholder data for this blogPost query return queryClient .getQueryData(['blogPosts']) ?.find((d) => d.id === blogPostId) }, }) mjendrisek wrote:>> Has anyone been successful at using parameters in Poll URL?>NopeThese are sent to shim during driver startup and the poll is initiated byshim. I donâ€t know of a way to notify the shim of a dynamic param outsideof a subscriber initiated event.One idea to try, poll something else that returns a static result. In itp,transform that to your timestamped query and execute it against shim.Grab the results of that query and use that instead of your dummy/staticresult. In a nutshell, React Query helps fetch and cache server-side data and share it across the components. It gives us two main hooks for data fetching: useQuery and useMutation. We can use useQuery to fetch data from the network and cache it.In this article, we will see how useQuery can ease our job in polling and refetching APIs conditionally with a small example. We will now dive into the requirements without further ado.React Query checklist for pollingCall the API when the component mounts, and if the attempt is successful, then return the data. We should also have the ability to stop this behavior with a flag. By default, this flag should be set as true. If the user sets the flag as false, then the API call shouldn’t be invoked as soon as the component mounts. It should happen only when it’s set to true.If the API call fails or the resource is not ready (eg. statuscode other than 200), retry until the server gives 200 as the status code.The number of retries in case of failures should be configurable. Once the limit has been reached, then polling should stop.When the component unmounts, then polling should stop.When the input parameters change, then polling should stop to initiate a new API call, and restart in case of any failure.The simplified rough outline of UI can be seen in the image below:The demo source code is available here.We can extract this to a custom hook (not battle-tested) as seen below.This hook satisfies our requirements. However there are a few things that we might need in real-time that might require additional work.In the above example, we used lodash deep equal. We can avoid this if we can maintain the same reference if the body didn’t change. But in cases where we are deriving the params and if it’s nested, it’s a little difficult to maintain it. Also, depending upon the nesting level, the order also matters even though we are using the same values.If we need this data to be available for some components in different hierarchies, we need to store this in either context or redux or something similar to share the data across the components.If we convince ourselves to use React Query, then all these operations become even more simpler with a few lines of useQuery: it’s highly configurable with a lot of batteries included. If the reduced lines of code do not interest you, then do have a look at the pros listed later in the blog.Our custom hook now will just have few lines, as shown below:We can cache data at a configurable time interval and also auto fetch it. If the data retrieval is successful, then it will update the UI or it will silently fail. We can eliminate the need to maintain the same reference or perform deep equal comparisons.Now if we need to read the data in any other component, we can do so just by calling queryClient.getQueryData(…). Automatically when the component unmounts, all the listeners will be unsubscribed by React Query. No additional work is required.Earlier, we were using redux store to place the server data that is shared across multiple components. By using React Query, we don’t need to do that any more, and it also reduces the need for a lot of boilerplate code. We highly recommend that you test and play around with React Query’s useQuery hook to see the benefits for yourself.Till then, keep an eye out for more posts from us on all things code! Comment récupérer des documents Word non enregistrés - Microsoft 365 Apps Somebody REALLY asked the QUESTION, "WHY does my UPPER LIP SMELL?" What Is the Concept of Bodhisattva? دليل استخدام مستكشف الملفات الجديد في Windows 11 ▷➡ IK4 ▷➡ Ada's Ea fine arts center Make Google your default search engine Yoga facial : quels exercices anti-rides pour le visage ? Student Portal/ULink - University of Johannesburg Boots | Women's | Red Wing Seki Edge Japanese Nail Clippers Leave Sharp, Clean Cuts Everyday Polisher Body Scrub Zahnärzte in Stuttgart Bad Cannstatt ATRRS Army | ATRRS Couse Catalog Details – Army Jobs ChatGPT : tout ce qu'il faut savoir sur cette prouesse technologique Sign in to Microsoft 365 استخدام الواتس آب على الكمبيوتر - موضوع Top 13 Lash Extension Styles - All You Need To Know Login funktioniert aktuell nicht? "Bei der Übertragung ist ein Fehler aufgetreten." ELSTER Anwender Forum Marvel Movies, Comics, TV, and Games Вызвать справочную систему windows можно способами Discover 10 Unique Cafes For Coffee In Seattle Pumpkin hängt beim Start (Pumpkin Autoradio) Street View in Google Maps verwenden - Android "Desdobramentos em Placard" - Página 12 Schreibung der s-Laute im Deutschen | Sprach|regeln Huawei Mobile Price in UAE Top 5 Football Prediction Sites In The World Today - Surestpredictions Blog Os 15 MELHORES Filmes de 2025 (Até o Momento)! Windows 11 Sistem Gereksinimleri - Microsoft Desteği Reddit - Reddit Investor Relations Браузер Google Chrome Att välja rätt tätningsmassa Sign up for a YouTube Premium or YouTube Music Premium annual plan - Android Zina Makeup - Villenave-d'Ornon - Atelier de Beauté Capucine Les 14 jeux vidéo les plus attendus de 2025 How to Flirt with a Girl over Text: 11 Steps (with Pictures) Dernières minutes de Neckermann - Lastminuteinfo [GA4] Find your Google tag ID Misogynie: Definition, Ursachen + Was dagegen tun? Here's every Persona 5 Royal school and exam answer in order Alle aktuellen Nachrichten von t-online.de - Politik Tai Chi Übungen für Anfänger Q Electric Barbecue Series | Weber® Official Flüchtige organische Verbindungen Jessica - Baby Name Meaning, Origin, and Popularity Ukraine strikes Moscow in biggest drone attack to date | CNN Tirage du mercredi 23 juillet 2025 Czy na aliexpress opłaca się kupować? Wasze opinie Manage your recommendations & search results - Computer Digital Nomad Packing List 2025: 10+ Essential Items The 17 Best Deodorants and Anti-perspirants for Long-Lasting Wear Customer Reviews: Sonic Colors Ultimate Nintendo Switch Bengkel Spesialis Mazda Jakarta : General Tips FREE French to Malagasy translation online ← Français ↔ Malagasy The 12 Best Speedrunning Games (And What Make Them So Great) pacar natasha wilona yang sekarang di malaysia Microsoft Corporate Office & Headquarters | Redmond, WA Nos restaurants préférés à tester absolument à Paris en 2025 ! Gesundheit | Produkte günstig kaufen auf Shop Apotheke Dragon Ball DAIMA Épisode 17 : Date de sortie et indices sur l'affrontement final How to sign in to Hotmail 7 milliarder til gode formål How To Get Help in Windows 11 Easily - Build Windows L'espace aqualudique Aquabaie Use Definition & Meaning | Britannica Dictionary Após colocar veneno na comida de Raquel, Walter tem final trágico em Vale Tudo: “A casa caiu” 【Recipient】の例文や意味・使い方 Temple of Seitan (Vegan) Menu & Prices - London Delivery - Order with Uber Eats 21 Top Toiletries Packing List Items for 2025 + What NOT to Bring Naruto: Why Was Hayate Gekkō Always Coughing During the Chunin Exams? Watch YouTube TV on your TV Find your files in Windows Letter Bb Song Video Lesson - Uppercase Lowercase Letters, Handwriting | KizPhonics ᐅ Unterschriftsberechtigung: Definition, Begriff und Erklärung im JuraForum.de Free R50 Sign up Bonus Offers Lavorare con lo strumento Linea pick (at) one's nails Using IF with AND, OR, and NOT functions in Excel How To Choose The Right Vacuum Cleaner: A Comprehensive Buying Guide | Vacuum Guru Hub Gay Freunde: Können zwei schwule Männer einfach nur befreundet sein? Traduzione Inglese Italiano online gratis CarePoint | LinkedIn Tips untuk digital marketing - Pusat Pembelajaran Google Workspace Angka Keluaran & Prediksi Togel Hari Ini | Data HK, SDY & SGP - TotoCX Einführung in Google Fotos - Computer Official Thread for Bugs and issues in Stadium Creator for MLB the Show 25 Comment créer un nouveau compte Gmail ? 5 Best Nail Salons In Seattle (2025) SEPA/СЕПА поддържани банки в България, SEPA развитие и Fintech фин. институции. Old ‘King of the Hill’ Didn’t Look That Great Either La tragica fine di Mirko Saric, l'altra 'Joya': il talento, la bellezza, il suicidio | Calciomercato Use your camera and microphone in Chrome - Computer Outlook Log In | Microsoft 365 Stadtbücherei Schorndorf - Startseite Canadian Dental Care Plan - Apply Neopunkt gazetka |16.07.2025| promocje, oferta - Lipiec Snoopy 2025 production issues? How to Install and Set Up R General Catalog - Music (MUS) Eyeliner kopen? Bestel je eyeliner direct online | Etos Author: Laurine Ryan Last Updated: 2025-07-25T03:05:15+07:00 Views: 6269 Rating: 4.7 / 5 (77 voted) Reviews: 84% of readers found this page helpful Name: Laurine Ryan Birthday: 1994-12-23 Address: Suite 751 871 Lissette Throughway, West Kittie, NH 41603 Phone: +2366831109631 Job: Sales Producer Hobby: Creative writing, Motor sports, Do it yourself, Skateboarding, Coffee roasting, Calligraphy, Stand-up comedy Introduction: My name is Laurine Ryan, I am a adorable, fair, graceful, spotless, gorgeous, homely, cooperative person who loves writing and wants to share my knowledge and understanding with you.