

Open in app ↗

Sign up

Sign in

Medium

 Search

Ways of Routing



Graeme Byrne · Follow

3 min read · Sep 5, 2023



Listen



Share

As I mentioned in the previous post, I would be writing more about the different ways of routing which the developer has at their disposal in SvelteKit.

Nested Routes

As mentioned before, SvelteKit is a file-based routing system. This means that the URL paths users can access in the browser are defined by files and folders in your codebase. This means that if I create a folder called products and add a file called +page.svelte, a SvelteKit convention, whatever is added to this file will appear when the user adds /products to the end of the URL.

However, the name products suggests that you probably have more than one, and you might want to have separate pages for each product. Nested routes can help here. To nest routes, create a new folder inside folder and then put a +page.svelte file in there. So in your products folder you could have, for example, separate folders called product-one, product-two, and product-three. Each of these folders would have their own +page.svelte file and adding the folder name to /products in the URL, for example /products/product-one, will take the user to the page for that specific product.

Dynamic Routes

Defining routes by pre-defined routes such as first, as in the nested routes above, isn't always feasible in complex apps. If you have 100 products, it can be difficult to keep track of so many routes, so the correct solution here is to use dynamic route segments. The product value should be a dynamic value that maps to a particular file in the products folder and in SvelteKit you can wrap a folder name with brackets to create a dynamic route.

To create a dynamic route, create a folder in the folder and put the name inside [], for example [productId], and put a +page.svelte file inside it. The [productId] is the dynamic part.

To extract the route parameter productId, we need to import a module that SvelteKit makes available for use in our app, which is the store module.

To import stores module, add the following at the top of the page in the script block:

```
<script>
  import {pages} from '$app/stores'
</script>
```

This page store has access to the parameters of a given route. To access that value, in this case productId, we have to destructure it by storing it in a constant:

```
<script>
  import {pages} from '$app/stores'

  const productId = $page.params.productId
</script>
```

The params object will contain a key called productId as this is the same dynamic segment specified for the folder name. Once we have the productId, we can render it as part of the HTML:

```
<script>
  import {pages} from '$app/stores'

  const productId = $page.params.productId
</script>

<h1>Details about product {productId}</h1>
```

Dynamic routes are useful when implementing the list detail pattern in any UI app.

Nested Dynamic Routes

So, we have our individual products thanks to dynamic routes. Now, perhaps, we would like to add a review for each product. Similar to nested routes, it's possible to create a new folder in [productId] called reviews. In reviews, create a new folder called [reviewId].

In [reviewId], import the following:

```
<script>
  import {pages} from '$app/stores'

  const {productId, reviews} = $page.params
</script>
```

The developer now has access to the reviews route and can use it similar to how productId was used in the dynamic routes section.

There are other examples of routing in SvelteKit which I will also cover in future posts.



Follow

Written by Graeme Byrne

2 Followers

More from Graeme Byrne



Graeme Byrne

TypeScript Objects, Arrays, Tuples, and Enums.

In this blog post, we'll dive into TypeScript's support for objects, arrays, tuples, and enums. These foundational data structures are...

Sep 28, 2023



Graeme Byrne

Rock, Paper, Scissors Game in Python Explained

As I continue learning and progressing, I have decided to learn how to use Python. Python can be used for web development, data analysis...

Oct 12, 2023 🖱 5



Graeme Byrne

TypeScript Functions, Literal Types, Type Aliases, and the “Never” Type

In the world of TypeScript, understanding functions, literal types, type aliases, and the elusive ‘never’ type can greatly enhance your...

Sep 29, 2023



Graeme Byrne

Svelte Logic

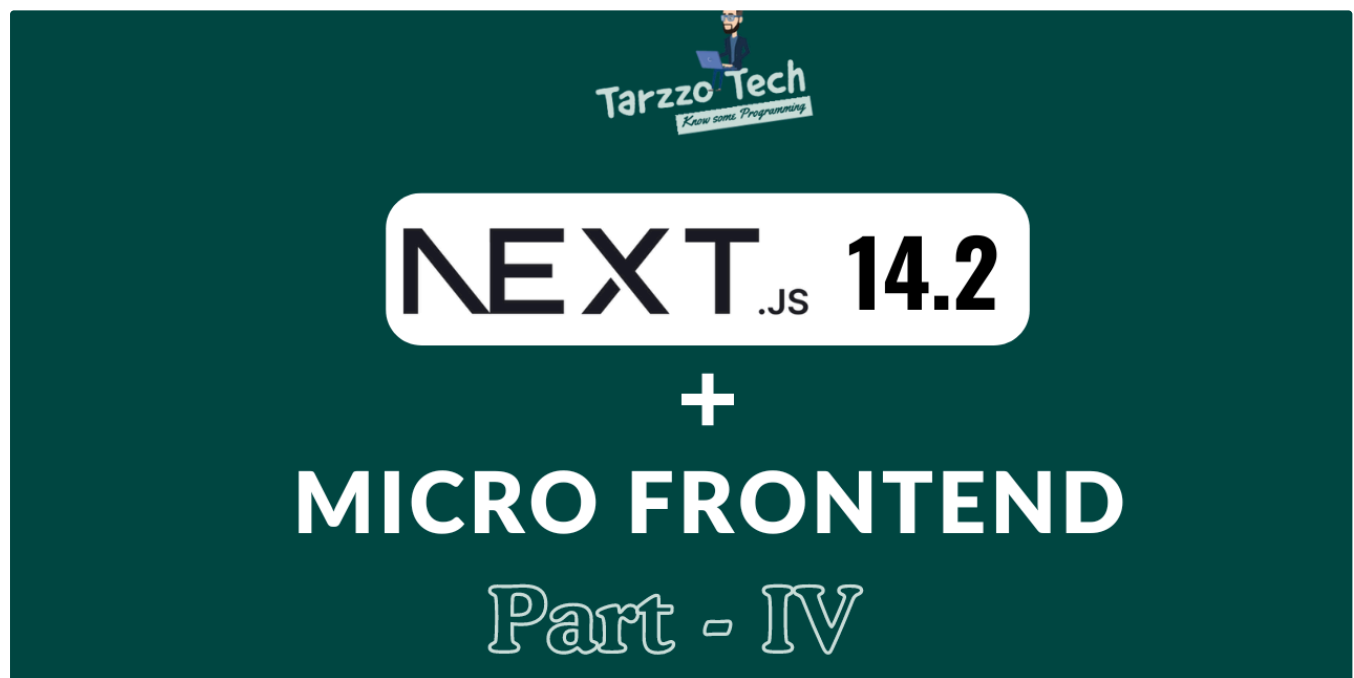
As I continue learning Svelte, I learn how to do more and more with it. In recent days, I have completed another tutorial on Svelte, this...

Aug 22, 2023



See all from Graeme Byrne

Recommended from Medium



Vijayasekhar Deepak in Stackademic

Building a Micro Frontend Application with Next.js 14.2 and Tailwind CSS —Part-IV

In my previous blog no:2, for sharing and using the micro frontend components, i have build the node server with express.js with cors npm...

4d ago 15



- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a \$300,000 job at Google.

1-page. Well-formatted.

★ Jun 1 🤝 15.1K 💬 231



Lists



Staff Picks

698 stories · 1177 saves



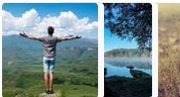
Stories to Help You Level-Up at Work

19 stories · 711 saves



Self-Improvement 101

20 stories · 2417 saves



Productivity 101

20 stories · 2118 saves



Afan Khan in JavaScript in Plain English

Microsoft is ditching React

Here's why Microsoft considers React a mistake for Edge.



Jun 6

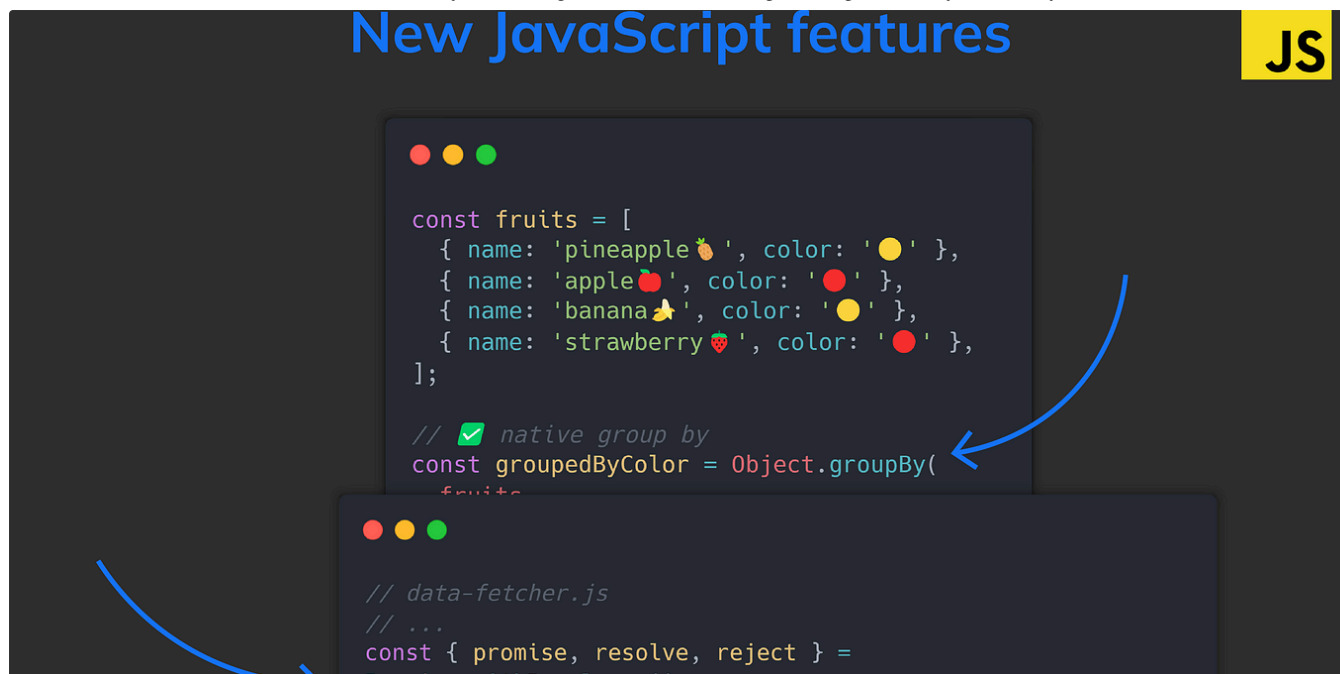


2.9K



66





Tari Ibaba in Coding Beauty

5 amazing new JavaScript features in ES15 (2024)

5 juicy ES15 features with new functionality for cleaner and shorter JavaScript code in 2024.



Jun 2




1.93K



10



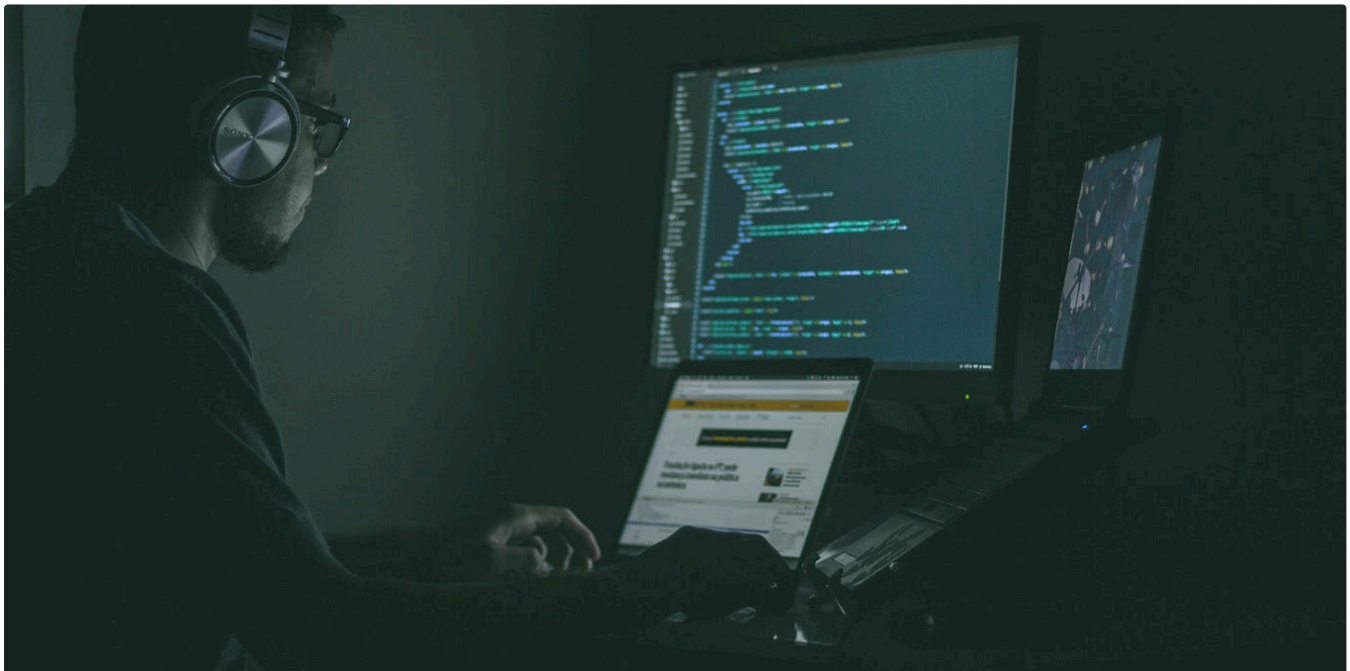


 Sufyan Maan, M.Eng in ILLUMINATION

What Happens When You Start Reading Every Day

Think before you speak. Read before you think.— Fran Lebowitz

★ Mar 12 🖱 28K 💬 634



 Mate Marschalko

28 JavaScript One-Liners every Senior Developer Needs to Know

Learn how to implement complex logic with beautifully short and efficient next-level JavaScript syntax.



Mar 1



1.2K



11



See more recommendations