#### > Menu



# **Route Segment Config**

The Route Segment options allows you to configure the behavior of a Page, Layout, or Route Handler by directly exporting the following variables:

Option	Туре	Default
experimental_ppr	'true'   'false'	
dynamic	'auto'   'force-dynamic'   'error'   'force-static'	'auto'
dynamicParams	boolean	true
revalidate	false   0   number	false
fetchCache	<pre>'auto'   'default-cache'   'only-cache'   'force-cache'   'force-no-store'   'default-no-store'   'only-no-store'</pre>	'auto'
runtime	'nodejs'   'edge'	'nodejs'
preferredRegion	'auto'   'global'   'home'   string   string[]	'auto'
maxDuration	number	Set by deployment platform

## **Options**

### experimental\_ppr

Enable Partial Prerendering (PPR) for a layout or page.

### dynamic

Change the dynamic behavior of a layout or page to fully static or fully dynamic.

```
1 export const dynamic = 'auto'
2 // 'auto' | 'force-dynamic' | 'error' | 'force-static'
```

Good to know: The new model in the app directory favors granular caching control at the fetch request level over the binary all-or-nothing model of getServerSideProps and getStaticProps at the page-level in the pages directory. The dynamic option is a way to opt back in to the previous model as a convenience and provides a simpler migration path.

- ('auto') (default): The default option to cache as much as possible without preventing any components from opting into dynamic behavior.
- 'force-dynamic': Force dynamic rendering, which will result in routes being rendered for each user at request time. This option is equivalent to:
  - getServerSideProps() in the pages directory.
  - Setting the option of every fetch() request in a layout or page to { cache: 'no-store', next: { revalidate: 0 } }.
  - Setting the segment config to export const fetchCache = 'force-no-store'
- 'error': Force static rendering and cache the data of a layout or page by causing an error if any components use dynamic functions or uncached data. This option is equivalent to:

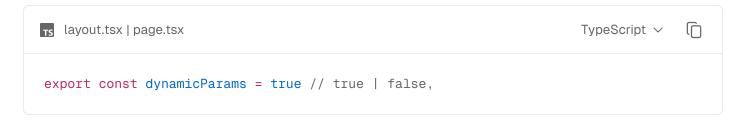
- getStaticProps() in the pages directory.
- Setting the option of every fetch() request in a layout or page to { cache: 'force-cache' }.
- Setting the segment config to fetchCache = 'only-cache', dynamicParams = false.
- dynamic = 'error' changes the default of dynamicParams from true to false. You can opt back into dynamically rendering pages for dynamic params not generated by generateStaticParams by manually setting dynamicParams = true.
- 'force-static': Force static rendering and cache the data of a layout or page by forcing cookies(), headers() and useSearchParams() to return empty values.

#### Good to know:

Instructions on how to migrate from <code>getServerSideProps</code> and <code>getStaticProps</code> to dynamic: <code>'force-dynamic'</code> and <code>dynamic: 'error'</code> can be found in the upgrade guide.

#### dynamicParams

Control what happens when a dynamic segment is visited that was not generated with generateStaticParams.



- true (default): Dynamic segments not included in generateStaticParams are generated on demand.
- false: Dynamic segments not included in generateStaticParams will return a 404.

#### Good to know:

- This option replaces the fallback: true | false | blocking option of getStaticPaths in the pages directory.
- To statically render all paths the first time they're visited, you'll need to return an empty array in generateStaticParams or utilize export const dynamic = 'force-static'.
- When dynamicParams = true, the segment uses Streaming Server Rendering.

- If the dynamic = 'error' and dynamic = 'force-static' are used, it'll change the default of dynamicParams to false.

#### revalidate

Set the default revalidation time for a layout or page. This option does not override the revalidate value set by individual fetch requests.

```
1 export const revalidate = false
2 // false | 0 | number
TypeScript \( \tag{\text{TypeScript}} \)
```

- option to 'force-cache' or are discovered before a dynamic function is used.

  Semantically equivalent to revalidate: Infinity which effectively means the resource should be cached indefinitely. It is still possible for individual fetch requests to use cache: 'no-store' or revalidate: 0 to avoid being cached and make the route dynamically rendered. Or set revalidate to a positive number lower than the route default to increase the revalidation frequency of a route.
- 0: Ensure a layout or page is always dynamically rendered even if no dynamic functions or uncached data fetches are discovered. This option changes the default of fetch requests that do not set a cache option to 'no-store' but leaves fetch requests that opt into 'force-cache' or use a positive revalidate as is.
- number: (in seconds) Set the default revalidation frequency of a layout or page to n seconds.

#### Good to know:

- The revalidate value needs to be statically analyzable. For example revalidate = 600 is valid, but revalidate = 60 \* 10 is not.
- The revalidate value is not available when using runtime = 'edge'.

#### **Revalidation Frequency**

- The lowest revalidate across each layout and page of a single route will determine the revalidation frequency of the *entire* route. This ensures that child pages are revalidated as frequently as their parent layouts.
- Individual fetch requests can set a lower revalidate than the route's default revalidate to increase the revalidation frequency of the entire route. This allows you to dynamically opt-in to more frequent revalidation for certain routes based on some criteria.

#### fetchCache

► This is an advanced option that should only be used if you specifically need to override the default behavior.

#### runtime

We recommend using the Node.js runtime for rendering your application, and the Edge runtime for Middleware (only supported option).

```
1 export const runtime = 'nodejs'
2 // 'nodejs' | 'edge'
TypeScript \( \tau_{\text{log}} \)
```

- 'nodejs' (default)
- 'edge'

Learn more about the different runtimes.

### preferredRegion

```
1 export const preferredRegion = 'auto'
2 // 'auto' | 'global' | 'home' | ['iad1', 'sfo1']
```

Support for preferredRegion, and regions supported, is dependent on your deployment platform.

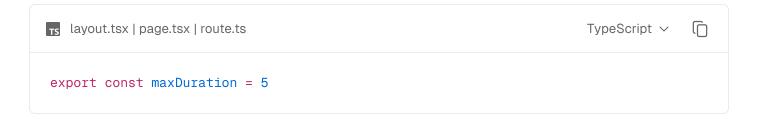
#### Good to know:

- If a preferredRegion is not specified, it will inherit the option of the nearest parent layout.
- The root layout defaults to all regions.

#### maxDuration

By default, Next.js does not limit the execution of server-side logic (rendering a page or handling an API). Deployment platforms can use maxDuration from the Next.js build output to add specific execution limits. For example, on Vercel 7.

**Note**: This settings requires Next.js 13.4.10 or higher.



#### Good to know:

- If using Server Actions, set the maxDuration at the page level to change the default timeout of all Server Actions used on the page.

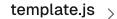
#### generateStaticParams

The generateStaticParams function can be used in combination with dynamic route segments to define the list of route segment parameters that will be statically generated at build time instead of on-demand at request time.

See the API reference for more details.

Previous







<b>▲</b> Vercel	Resources	More	About Vercel	Legal		
	Docs	Next.js Commerce	Next.js + Vercel	Privacy Policy		
	Learn	Contact Sales	Open Source Software			
	Showcase	GitHub	GitHub			
	Blog	Releases	Χ			
	Analytics	Telemetry				
	Next.js Conf	Governance				
	Previews					
Subscribe to our newsletter						
Stay updated on new releases and						

features, guides, and case studies.

you@domain.com

Subscribe

© 2024 Vercel, Inc.





