

# Generate pages on demand Incremental Static Generation (ISG) With Nuxt.js and Layer0



# Rishi Raj Jain

**Solutions Engineer** at Layer0 by Limelight  
**Storyblok Ambassador**

# What to expect from this talk.

- **What is ISG?**
- **Benefits of ISG**
- **Drawbacks of ISG**
- **Implementing ISG with Nuxt.js and Layer0**

# What is Layer0?

# Simple developer workflow. Instant-loading sites.

All-in-one Jamstack platform to develop, deploy, preview, split test and monitor your frontend

Deploy Free

Get a Demo

## Powering Sub-Second Websites

REVOLVE

SHOE CARNIVAL

AKIRA

TIE BAR

SHARPER IMAGE

kate spade



296ms loads  
899K pages



337ms loads  
229K pages



275ms loads  
223K pages



282ms loads  
52K pages



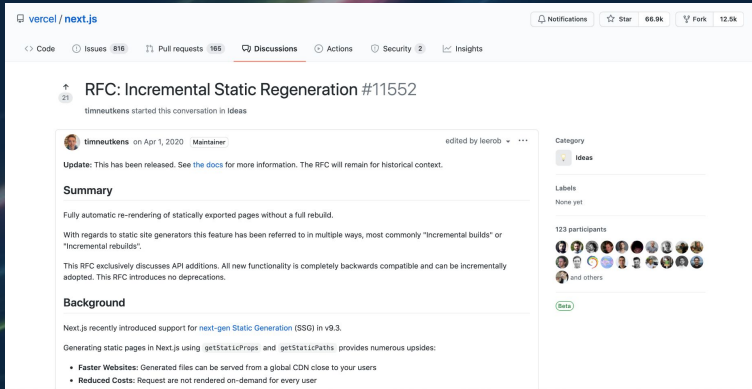
386ms loads  
48K pages



407ms loads  
46K pages

# What is Layer0? All-in-one Jamstack platform.

# 01 – What is ISG?



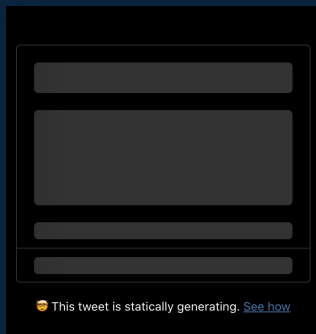
- Generate static pages on demand,
- “Best of both worlds”
  - Builds only the pages you need,
  - Fast “static” page loads after the first visit
- Issues
  - Feature of your infrastructure and your framework,
  - Is limited to Next.js on certain platforms

# 01 – What is ISG?

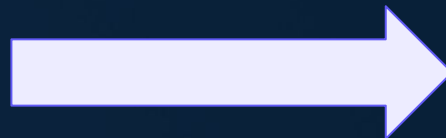
# Incremental Static Generation

- ISG on first visit when set to `fallback: true`

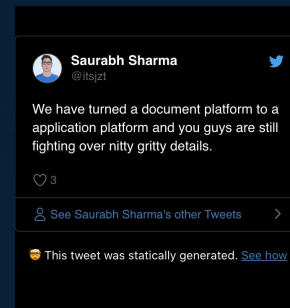
Browser requests a new page that has not yet been visited



CDN quickly returns a universal “fallback” page with placeholder data...



...while fallback is being displayed, page’s static build process is run

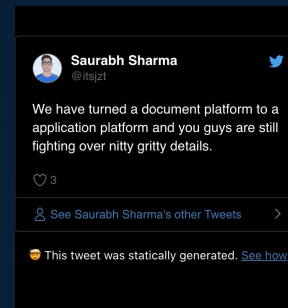


Build completes and fallback loads the static JSON data displaying the final page.



# Incremental Static Generation

- ISG on subsequent visits when set to `fallback: true`



Future visits will get the statically built page HTML (no waiting).

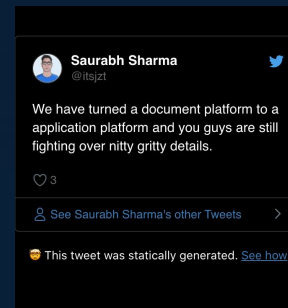
# Incremental Static Generation

- ISG on first visit when set to fallback: blocking

Browser requests  
a new page  
that has not yet  
been visited

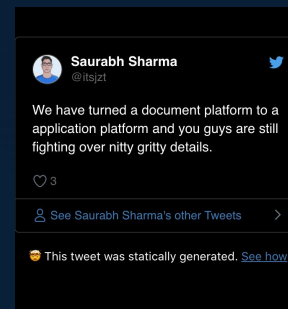


Browser waits for the  
SSR generated response



# Incremental Static Generation

- ISG on subsequent visits when set to fallback: blocking



Future visits will get the statically built page HTML (no waiting).

## 02 - Benefits of ISG

- **Pages on-demand,**
- **Minimal static build,**
- **No more re-building the website,**
- **Faster deployments,**
- **Easy cache purging with Layer0 REST API**

## **02 - Benefits of ISG**

## 03 - Drawbacks of ISG

- **Fallback first for the first user,**
- **Some users might run into stale data,**
- **Breaks immutability**
  - **But not with Layer0 as you can choose whether to keep the cache between deployments**

## **03 - Drawbacks of ISG**

# 04 - Implementing ISG with Nuxt.js and Layer0



# 04 - In Action: ISG with Nuxt.js and Layer0

- **Setup API Routes with Nuxt,**
- **Setup dynamic routes with Nuxt,**
  - **Cache dynamic routes on the edge**
- **Configure Layer0 EdgeJS**

## **04 - Steps to implement ISG with Nuxt.js and Layer0**

# api/blogs.js

```
1 const express = require('express')
2 const app = express()
3
4 const hello = require('./blogs')
5 app.use(hello)
6
7 if (require.main === module) {
8   const port = 3001
9   app.listen(port, () => {
10     console.log(`API server listening on port ${port}`)
11   })
12 }
13
14 module.exports = app
```

# api/index.js

## 04.1 - Setup API Routes with Nuxt

```
1 const { parse } = require('rss-to-json')
2 const { Router } = require('express')
3 const router = Router()
4
5 router.use('/blogs/:username.json', async (req, res) => {
6   const slug = req.params.username
7   let errorResponse = (res) => {
8     res.writeHead(404, { 'Content-Type': 'application/json' })
9     res.end(
10       JSON.stringify({
11         code: 0,
12       })
13     )
14   }
15   try {
16     let rss = await parse(`https://medium.com/feed/@${slug}`)
17     let resp = JSON.stringify(rss, null, 3)
18     if (!resp) {
19       return errorResponse(res)
20     }
21     resp = JSON.parse(resp)
22     res.writeHead(200, { 'Content-Type': 'application/json' })
23     res.end(
24       JSON.stringify({
25         resp,
26         code: 1,
27       })
28     )
29   } catch {
30     return errorResponse(res)
31   }
32 })
33
34 module.exports = router
```

pages / blogs / \_slug.vue

```
async asyncData({ params, redirect }) {
  let link = process.env.API_URL
  if (typeof window !== 'undefined') link = window.location.origin
  let blogsData = await fetch(`${link}/api/blogs/${params.slug}.json`).then((res) => res.json())
  if (blogsData['code'] == 0) redirect(404, '/error')
  return {
    resp: blogsData['resp'],
    slug: params.slug,
    link,
  }
},
```

## 04.2 - Setup dynamic routes with Nuxt

```
mounted() {  
  if (typeof window !== 'undefined' && window.__client__ === true) {  
    window.__client__ = false  
    console.log('Client Side Transition, Populating the cache...')  
    fetch(`/blogs/${this.slug}`)  
  }  
},
```

## 04.3 - Cache dynamic routes on the edge

```
15 .get('/blogs/:username', ({ serveStatic, cache, renderWithApp }) => {
16   cache({
17     edge: {
18       maxAgeSeconds: 60 * 60 * 24 * 365, // keep the incrementally generated page for a year
19       staleWhileRevalidateSeconds: 1, // revalidate the data on page every second
20     },
21     browser: false,
22   })
23   serveStatic('dist/blogs/:username.html', {
24     // When the user requests a page that is not already statically rendered, fall back to SSR.
25     onNotFound: () => renderWithApp(),
26   })
27 })
28 .get('/api/blogs/:username.json', ({ serveStatic, cache, renderWithApp }) => {
29   cache({
30     edge: {
31       maxAgeSeconds: 60 * 60 * 24, // cache at the edge for 24 hours
32     },
33   })
34   serveStatic('dist/blogs/:username.json', {
35     // When the user requests data that is not already statically rendered, fall back to SSR.
36     onNotFound: () => renderWithApp(),
37   })
38 })
```

## 04.4 - Configure Layer0 EdgeJS



<https://rishi-raj-jain-nuxt-isg-default.layer0.link/>

Nuxt 3 has been re-architected with a smaller core and optimized for faster performance and better developer experience.



### Lighter

Up to 75x smaller server deployments and smaller client bundle targeting modern browsers.



### Faster

Optimized cold start with dynamic server code-splitting, powered by nitro.



### Hybrid soon

Incremental Static Generation and other advanced modes are now possible.



### Suspense

Fetch data in any component, before or after navigation.



### Composition API

Use the Composition API and Nuxt 3's composables for true code re-usability.



### Nuxt CLI

A new zero-dependency experience for easy scaffolding and module integration.



### Nuxt Devtools soon

Work faster with info and quick fixes right in the browser.



### Nuxt Kit

Brand new module development with Typescript and cross-version compatibility.



### Webpack 5

Faster build time and smaller bundle size, with no configuration required.



### Vite

Experience lightning fast HMR by using Vite as your bundler.



### Vue 3

Vue 3 is a solid foundation for your next web app.



### TypeScript

Built with native TypeScript and ESM - no extra steps required.

[Get started](#)



**Thank You!**