Static Sites with Vue, Nuxt and Node

Serverless with Vue and Node

Serverless static sites with Vue and Node

What a title!

Roman Kuba

Austria,

Engineering Manager @ GitLab

@codebryo

codebryo.com

What does serverless mean?

Actually it means nothing.

How do we experience serverless today?

Two Consumers of Serverless

1 Applications

2 Static Sites

Static sites can benefit from Serverless!

Static sites can scale massively

Smashing Magazine — For Web

https://www.smashingmagazine.com

Guest

Articles
Design & development

Books
Physical & digital books

Events
Conferences & workshops

Jobs
Find work & employees

Membership
Webinars & early-birds

Topics

Seriously, red?

Don't Miss These Latest Articles

Cosima Mielke *wrote*

29 Days Of February (2020 Wallpapers Edition)

Leave a comment

Wallpapers 126

Anna Prenzel *wrote*

How To Create A Card Matching Game Using Angular And RxJS

Leave a comment

UI 78 # JavaScript 247 # User Interaction 63

Codebryo

The JAM Stack

Let’s build some stuff

Page

Page

Data

Page

Data

Serverless

Page

Data

Serverless

Data

Page

Data

Pipeline

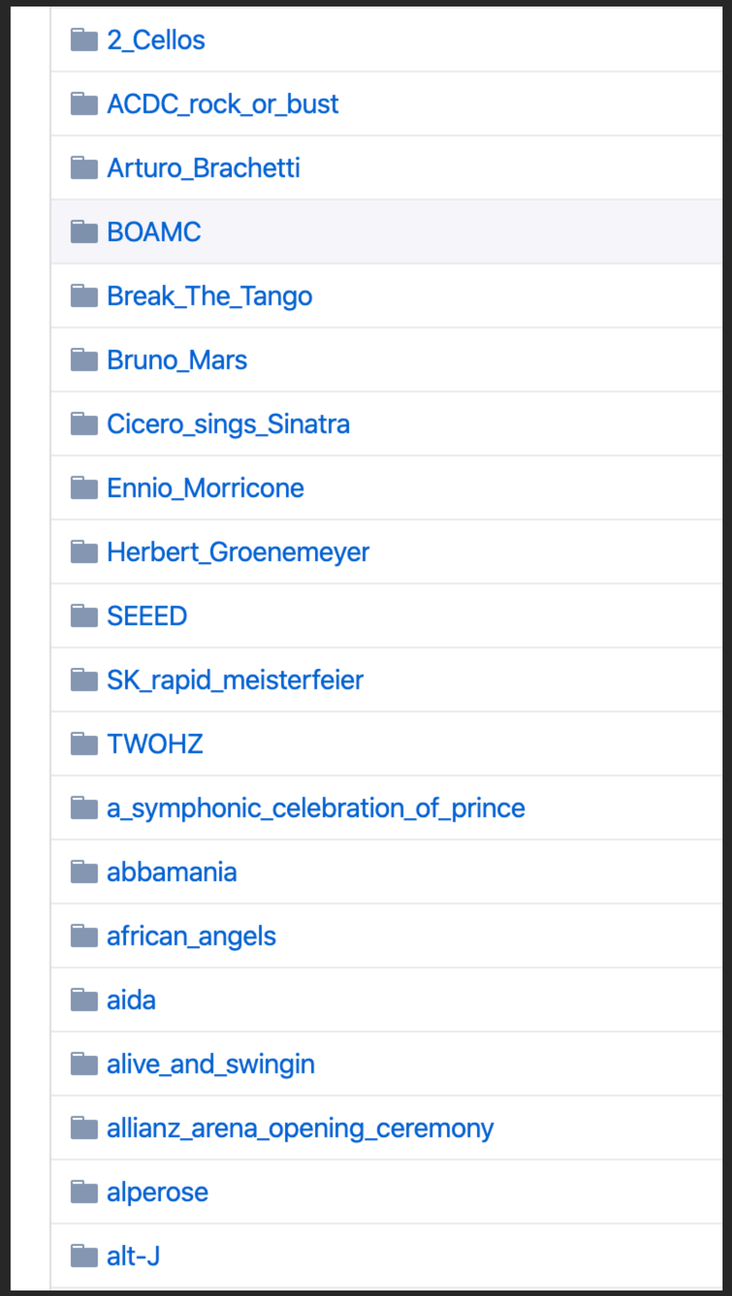
Page

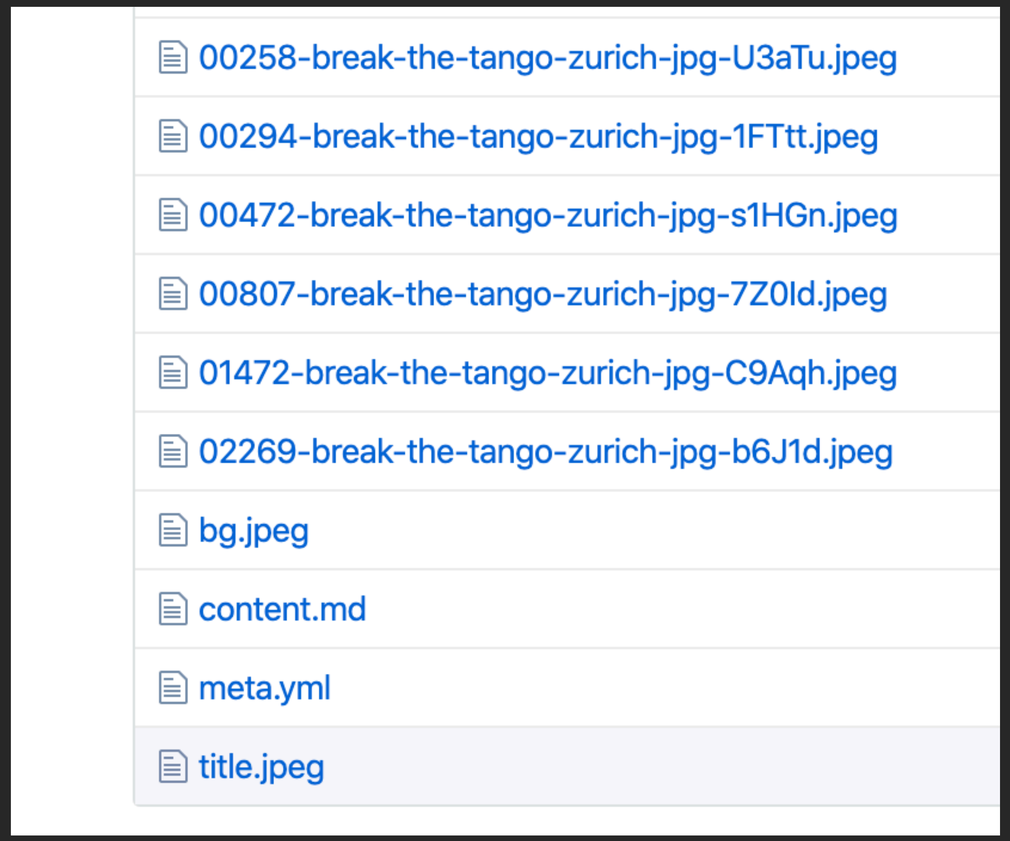
The Pipeline is key!

graph.json

```
const path = require('path')
const fs = require('fs').promises

const PATHS = {
  CONTENT: path.join(__dirname, '..', 'content'),
  OUTPUT: path.join(__dirname, '..', 'static')
}

async function processMarkdownFiles(inputPath) {
  const files = await fs.readdir(inputPath)

  ...

  return await files.reduce( ... )

}
```

Sometimes limitations are great


2_Cellos
ACDC_rock_or_bust
Arturo_Brachetti
BOAMC
Break_The_Tango
Bruno_Mars
Cicero_sings_Sinatra
Ennio_Morricone
Herbert_Groenemeyer
SEEED
SK_rapid_meisterfeier
TWOHZ
a_symphonic_celebration_of_prince
abbamania
african_angels
aida
alive_and_swingin
allianz_arena_opening_ceremony
alperose
alt-J


00258-break-the-tango-zurich-jpg-U3aTu.jpeg
00294-break-the-tango-zurich-jpg-1FTtt.jpeg
00472-break-the-tango-zurich-jpg-s1HGn.jpeg
00807-break-the-tango-zurich-jpg-7Z0Id.jpeg
01472-break-the-tango-zurich-jpg-C9Aqh.jpeg
02269-break-the-tango-zurich-jpg-b6J1d.jpeg
bg.jpeg
content.md
meta.yml
title.jpeg

```
const path = require('path')
const fs = require('fs').promises

const PATHS = {
  CONTENT: path.join(__dirname, '..', 'content'),
  OUTPUT: path.join(__dirname, '..', 'static')
}

async function processMarkdownFiles(inputPath) {
  const files = await fs.readdir(inputPath)

  ...

  return await files.reduce( ... )

}
```

```
const path = require('path')
const fs = require('fs').promises

const PATHS = {
  CONTENT: path.join(__dirname, '..', 'content'),
  OUTPUT: path.join(__dirname, '..', 'static')
}

async function processMarkdownFiles(inputPath) {
  const files = await fs.readdir(inputPath)

  ...

  return await files.reduce( ... )

}
```

```
const fm = require('front-matter')

const filesReducer = async (accumulatorPromise, file) => {
  const accumulator = await accumulatorPromise
  const fileNameWithoutExtension = path.basename(
    file.name,
    path.extname(file.name)
  )

  const fileStream = await fs.readFile(
    path.join(inputPath, '/', file.name),
    'utf8'
  )
  // Turn the raw .md content into a JSON object
  const content = fm(fileStream)
  // assign it to the main graph object
  accumulator[fileNameWithoutExtension] = content
  return accumulator
}
```

```
const fm = require('front-matter')

const filesReducer = async (accumulatorPromise, file) => {
  const accumulator = await accumulatorPromise
  const fileNameWithoutExtension = path.basename(
    file.name,
    path.extname(file.name)
  )

  const fileStream = await fs.readFile(
    path.join(inputPath, '/', file.name),
    'utf8'
  )
  // Turn the raw .md content into a JSON object
  const content = fm(fileStream)
  // assign it to the main graph object
  accumulator[fileNameWithoutExtension] = content
  return accumulator
}
```

```
const path = require('path')
const fs = require('fs').promises
const fm = require('front-matter')

const PATHS = {
  CONTENT: path.join(__dirname, '..', 'content'),
  OUTPUT: path.join(__dirname, '..', 'static')
}

async function processMarkdownFiles(inputPath) {
  const files = await fs.readdir(inputPath)
  const filesReducer = async (accumulatorPromise, file) => { ... }

  try {
    return await files.reduce(filesReducer, Promise.resolve({}))
  } catch (err) {
    console.error(err)
  }
}
```

```
import sharp from 'sharp'

async function processImages(imagePath) {
  const files = await fs.readdir(currentPath)

  const images = files.filter(fileIsImageFilter) // Check File ext

  for (const image of images) {
    await resizeAndSave(image)
  }

  return images
}
```

```
async function resizeAndSave(inputFile) {
  const filename = path.basename(inputFile)

  const IMAGE_VALUES = {
    def: 1200,
    small: 400
  }

  for (const [key, val] of Object.entries(IMAGE_VALUES)) {
    const savePath = path.join(PATHS.OUTPUT, '/', `${key}_${filename}`)
    await sharp(inputFile).resize(value).toFile(savePath)
  }
}
```

cat.jpeg
> def_cat.jpeg
> small_cat.jpeg

```
async function writeGraphFile() {
  const graph = {
    posts: await processMarkdownFiles(PATHS.CONTENT),
    images: await processImages(PATHS.CONTENT)
  }

  try {
    await fs.writeFile(
      `${PATHS.OUTPUT}/graph.json`,
      JSON.stringify(graph, null, 4)
    )
    console.log('Graph file generated!')
  } catch (err) {
    console.error(err)
  }
}
```

/scripts/graph.js

```
const path = require('path')
const fs = require('fs').promises
const fm = require('front-matter')
const sharp = require('sharp')

const PATHS = { ... }

async function processMarkdownFiles(inputPath) {
  ...
}

async function processImages(imagePath) {
  ...
}

const filesReducer = async (accumulatorPromise, file) => { ... }

async function writeGraphFile() {
  ...
}

writeGraphFile()
```

```
node scripts/graph.js
```

```
<template>
  <Content>
    <Post v-cloak :post="post" />
  </Content>
</template>

<script>
import Content from '~/components/content'
import Post from '~/components/post'
import graph from '~/static/graph.json'

function pickPostBySlug(slug) {
  const match = Object.entries(graph.posts).find(([key, value]) => {
    if (value.attributes.slug === slug) return true
  })
  return match[1] // return just the value
}

export default {
  components: {
    Content,
    Post
  },

  data() {
    return {
      post: pickPostBySlug(this.$route.params.slug)
    }
  }
}
</script>
```

Automation FTW

Automation FTW

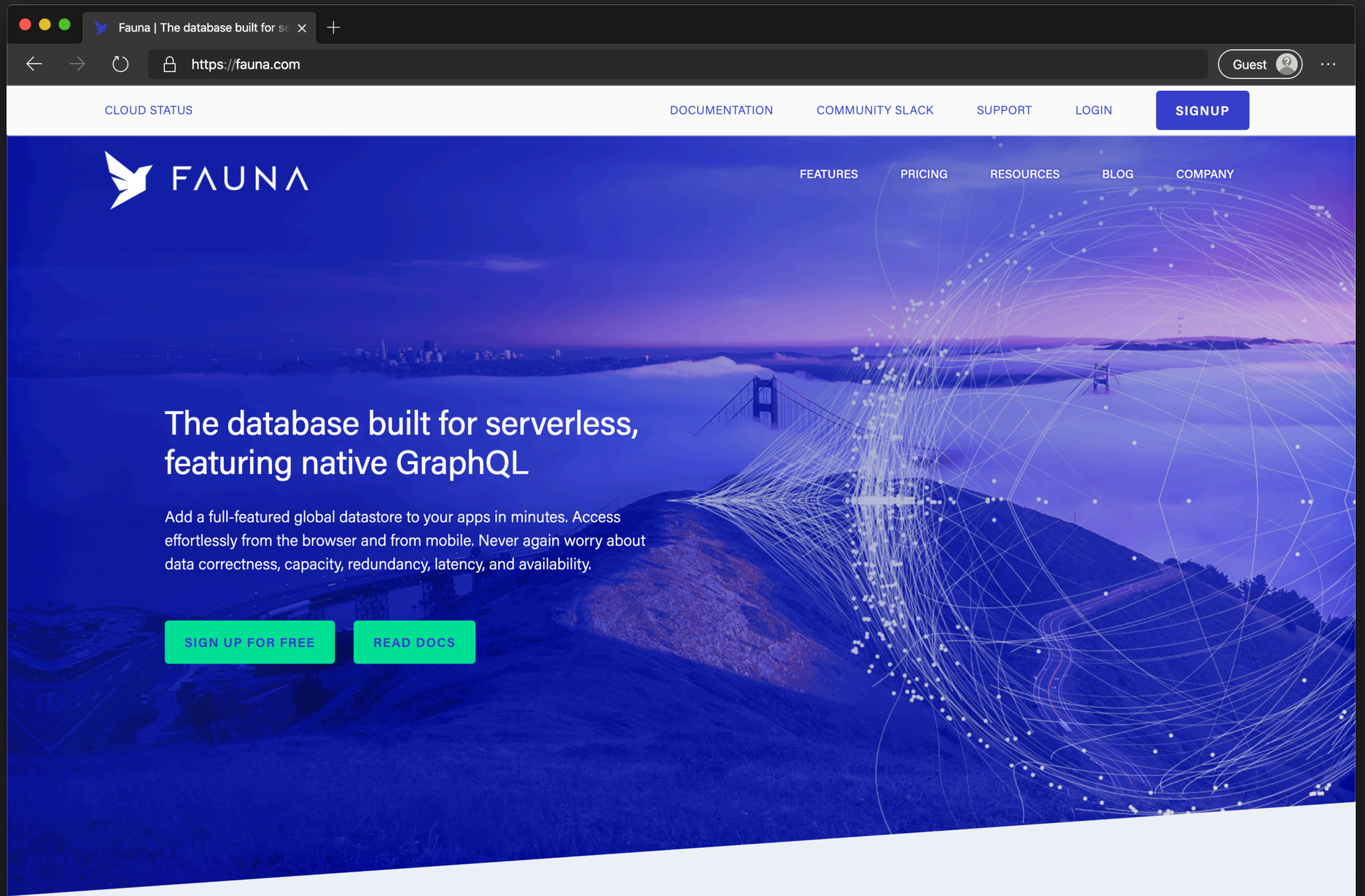
netlify

What about serverless?

What do we need

1. Store Data
2. Lamda Functions Retrieve and Set Data **(FaaS**)
3. Data preparation

1


Fauna | The database built for se
https://fauna.com
Guest
CLOUD STATUS
DOCUMENTATION
COMMUNITY SLACK
SUPPORT
LOGIN
SIGNUP
FAUNA
FEATURES
PRICING
RESOURCES
BLOG
COMPANY
The database built for serverless, featuring native GraphQL
Add a full-featured global datastore to your apps in minutes. Access effortlessly from the browser and from mobile. Never again worry about data correctness, capacity, redundancy, latency, and availability.
SIGN UP FOR FREE
READ DOCS

Codebryo

FQL

```
client.query(
  q.Get(q.Ref(q.Collection('posts'), '192903209792046592'))
)

// returns
{
  ref:
  Ref(id = 192903209792046592, collection = Ref(id = posts, collection = Ref(id = collections))),
    ts: 1527350638301882,
      data: { title: 'My cat and other marvels' }
}
```

Lambda Functions

1 Time Functions

Firebase Extensions BETA

Deploy functionality to your app quickly using pre-packaged solutions. Firebase Extensions are configurable, and work with Firebase and other Google Cloud Platform products.

Resize Images

Translate Text

Sync with Mailchimp

Trigger Email

What do we need

1. Function that retrieves the data for us
2. Function that stores the data for us

```vue
<template>
  <div class="flex items-center flex-grow">
    <button
      @mousedown="startClap"
      @mouseup="stopClap"
      title="Hold to clap"
      class="border-white border p-2 rounded hover:border-yellow relative"
    >
      <div
        class="flex flex-row items-center pointer-events-none justify-between"
      >
        <div>
          <img
            id="logo"
            src="~/assets/img/clap.svg"
            alt="Clapping Hands"
            class="h-12"
          />
        </div>
        <div class="text-center ml-4 w-10">
          {{ currentCounter }}
        </div>
      </div>
    </button>

    <div class="ml-4">
      <transition
        enter-active-class="transition-all"
        leave-active-class="transition-all"
        enter-class="scale-70"
        enter-to-class="opacity-1 scale-100"
        leave-class="opacity-1 scale-100"
        leave-to-class="scale-130 opacity-0"
      >
        <span
          v-show="newCountVisible"
          class="text-sm text-gray-900 inline-block w-12 rounded-lg bg-yellow text-center"
          >+ {{ newCount }}</span
        >
      </transition>
    </div>
  </div>
</template>

<script>
import debounce from 'lodash/debounce'

const speedMap = {
  slow: 300,
  medium: 100,
  fast: 40
}

export default {
  props: {
    id: {
      type: String,
      required: true
    }
  },

  data() {
    return {
      currentCounter: 0,
      newCount: 0,
      newCountVisible: false,
      newCountBuffer: 0,
      clickMeta: {
        start: undefined,
        timeout: undefined,
        holdTimeout: undefined,
        lastSpeed: undefined
      }
    }
  },

  async mounted() {
    try {
      const postMetaResponse = await fetch(
        `/.netlify/functions/get-post-meta/${this.id}`
      )
      const postMeta = await postMetaResponse.json()
      this.currentCounter = postMeta.data.claps
    } catch (e) {
      console.error(
        'This Error happend when trying to fetch the original post meta.',
        e
      )
    }
  },

  methods: {
    holdCounter() {
      if (this.clickMeta.lastSpeed === 'fast') {
        const now = new Date()
        const diff = now - this.clickMeta.start

        if (diff > 2500) this.clickMeta.lastSpeed = 'fast'
        else if (diff > 1000) this.clickMeta.lastSpeed = 'medium'
        else this.clickMeta.lastSpeed = 'slow'
      }

      this.newCount += speedMap[this.clickMeta.lastSpeed]
      this.clickMeta.holdTimeout = setTimeout(this.holdCounter, speedMap[this.clickMeta.lastSpeed])
    },

    startClap() {
      this.clickMeta.start = new Date()
      this.newCountVisible = true
      this.newCount = 1
      this.clickMeta.timeout = setTimeout(this.holdCounter, 500)
    },

    stopClap() {
      for (const key of ['timeout', 'holdTimeout']) {
        clearTimeout(this.clickMeta[key])
        this.clickMeta[key] = undefined
      }
      for (const key of ['start', 'lastSpeed']) {
        this.clickMeta[key] = undefined
      }
      this.newCountVisible = false
      this.currentCounter = this.currentCounter + this.newCount
      this.newCountBuffer = this.newCountBuffer + this.newCount

      this.syncClaps()
    },

    syncClaps: debounce(
      async function() {
        const claps = this.newCountBuffer
        this.newCountBuffer = 0
        const postMetaResponse = await fetch(
          `/.netlify/functions/update-post-meta/${this.id}`,
          {
            body: JSON.stringify({ claps }),
            method: 'POST'
          }
        )
        const postMeta = await postMetaResponse.json()
        this.currentCounter = postMeta.data.claps
      },
      500,
      { maxWait: 2000 }
    )
  }
}
</script>
```

```vue
<template>
  <div class="flex items-center flex-grow">
    <button
      @mousedown="startClap"
      @mouseup="stopClap"
      title="Hold to clap"
      class="border-white border p-2 rounded hover:border-yellow relative"
    >
      <div
        class="flex flex-row items-center pointer-events-none justify-between"
      >
        <div>
          <img
            id="logo"
            src="~/assets/img/clap.svg"
            alt="Clapping Hands"
            class="h-12"
          />
        </div>
        <div class="text-center ml-4 w-10">
          {{ currentCounter }}
        </div>
      </div>
    </button>

    <div class="ml-4">
      <transition
        enter-active-class="transition-all"
        leave-active-class="transition-all"
        enter-class="scale-70"
        enter-to-class="opacity-1 scale-100"
        leave-class="opacity-1 scale-100"
        leave-to-class="scale-130 opacity-0"
      >
        <span
          v-show="newCountVisible"
          class="text-sm text-gray-900 inline-block w-12 rounded-lg bg-yellow text-center"
          >+ {{ newCount }}</span
        >
      </transition>
    </div>
  </div>
</template>

<script>
import debounce from 'lodash/debounce'

const speedMap = {
  slow: 300,
  medium: 100,
  fast: 40
}

export default {
  props: {
    id: {
      type: String,
      required: true
    }
  },

  data() {
    return {
      currentCounter: 0,
      newCount: 0,
      newCountVisible: false,
      newCountBuffer: 0,
      clickMeta: {
        start: undefined,
        timeout: undefined,
        holdTimeout: undefined,
        lastSpeed: undefined
      }
    }
  },

  async mounted() {
    try {
      const postMetaResponse = await fetch(
        `/.netlify/functions/get-post-meta/${this.id}`
      )
      const postMeta = await postMetaResponse.json()
      this.currentCounter = postMeta.data.claps
    } catch (e) {
      console.error(
        'This Error happend when trying to fetch the original post meta.',
        e
      )
    }
  },

  methods: {
    holdCounter() {
      if (this.clickMeta.lastSpeed === 'fast') {
        const now = new Date()
        const diff = now - this.clickMeta.start

        if (diff > 2500) this.clickMeta.lastSpeed = 'fast'
        else if (diff > 1000) this.clickMeta.lastSpeed = 'medium'
        else this.clickMeta.lastSpeed = 'slow'
      }

      this.newCount += speedMap[this.clickMeta.lastSpeed]
      this.clickMeta.holdTimeout = setTimeout(this.holdCounter, speedMap[this.clickMeta.lastSpeed])
    },

    startClap() {
      this.clickMeta.start = new Date()
      this.newCountVisible = true
      this.newCount = 1
      this.clickMeta.timeout = setTimeout(this.holdCounter, 500)
    },

    stopClap() {
      for (const key of ['timeout', 'holdTimeout']) {
        clearTimeout(this.clickMeta[key])
        this.clickMeta[key] = undefined
      }
      for (const key of ['start', 'lastSpeed']) {
        this.clickMeta[key] = undefined
      }
      this.newCountVisible = false
      this.currentCounter = this.currentCounter + this.newCount
      this.newCountBuffer = this.newCountBuffer + this.newCount

      this.syncClaps()
    },

    syncClaps: debounce(
      async function() {
        const claps = this.newCountBuffer
        this.newCountBuffer = 0
        const postMetaResponse = await fetch(
          `/.netlify/functions/update-post-meta/${this.id}`,
          {
            body: JSON.stringify({ claps }),
            method: 'POST'
          }
        )
        const postMeta = await postMetaResponse.json()
        this.currentCounter = postMeta.data.claps
      },
      500,
      { maxWait: 2000 }
    )
  }
}
</script>
```

```
async mounted() {
   try {
     const postMetaResponse = await fetch(
       `/.netlify/functions/get-post-meta/${this.id}`
     )
     const postMeta = await postMetaResponse.json()
     this.currentCounter = postMeta.data.claps
   } catch (e) {
     console.error(
       'This Error happened when trying to fetch the original post meta.',
       e
     )
   }
 },
```

```vue
<template>
  <div class="flex items-center flex-grow">
    <button
      @mousedown="startClap"
      @mouseup="stopClap"
      title="Hold to clap"
      class="border-white border p-2 rounded hover:border-yellow relative"
    >
      <div
        class="flex flex-row items-center pointer-events-none justify-between"
      >
        <div>
          <img
            id="logo"
            src="~/assets/img/clap.svg"
            alt="Clapping Hands"
            class="h-12"
          />
        </div>
        <div class="text-center ml-4 w-10">
          {{ currentCounter }}
        </div>
      </div>
    </button>

    <div class="ml-4">
      <transition
        enter-active-class="transition-all"
        leave-active-class="transition-all"
        enter-class="scale-70"
        enter-to-class="opacity-1 scale-100"
        leave-class="opacity-1 scale-100"
        leave-to-class="scale-130 opacity-0"
      >
        <span
          v-show="newCountVisible"
          class="text-sm text-gray-900 inline-block w-12 rounded-lg bg-yellow text-center"
          >+ {{ newCount }}</span
        >
      </transition>
    </div>
  </div>
</template>

<script>
import debounce from 'lodash/debounce'

const speedMap = {
  slow: 300,
  medium: 100,
  fast: 40
}

export default {
  props: {
    id: {
      type: String,
      required: true
    }
  },

  data() {
    return {
      currentCounter: 0,
      newCount: 0,
      newCountVisible: false,
      newCountBuffer: 0,
      clickMeta: {
        start: undefined,
        timeout: undefined,
        holdTimeout: undefined,
        lastSpeed: undefined
      }
    }
  },

  async mounted() {
    try {
      const postMetaResponse = await fetch(
        `/.netlify/functions/get-post-meta/${this.id}`
      )
      const postMeta = await postMetaResponse.json()
      this.currentCounter = postMeta.data.claps
    } catch (e) {
      console.error(
        'This Error happend when trying to fetch the original post meta.',
        e
      )
    }
  },

  methods: {
    holdCounter() {
      if (this.clickMeta.lastSpeed === 'fast') {
        const now = new Date()
        const diff = now - this.clickMeta.start

        if (diff > 2500) this.clickMeta.lastSpeed = 'fast'
        else if (diff > 1000) this.clickMeta.lastSpeed = 'medium'
        else this.clickMeta.lastSpeed = 'slow'
      }

      this.newCount += speedMap[this.clickMeta.lastSpeed]
      this.clickMeta.holdTimeout = setTimeout(this.holdCounter, speedMap[this.clickMeta.lastSpeed])
    },

    startClap() {
      this.clickMeta.start = new Date()
      this.newCountVisible = true
      this.newCount = 1
      this.clickMeta.timeout = setTimeout(this.holdCounter, 500)
    },

    stopClap() {
      for (const key of ['timeout', 'holdTimeout']) {
        clearTimeout(this.clickMeta[key])
        this.clickMeta[key] = undefined
      }
      for (const key of ['start', 'lastSpeed']) {
        this.clickMeta[key] = undefined
      }
      this.newCountVisible = false
      this.currentCounter = this.currentCounter + this.newCount
      this.newCountBuffer = this.newCountBuffer + this.newCount

      this.syncClaps()
    },

    syncClaps: debounce(
      async function() {
        const claps = this.newCountBuffer
        this.newCountBuffer = 0
        const postMetaResponse = await fetch(
          `/.netlify/functions/update-post-meta/${this.id}`,
          {
            body: JSON.stringify({ claps }),
            method: 'POST'
          }
        )
        const postMeta = await postMetaResponse.json()
        this.currentCounter = postMeta.data.claps
      },
      500,
      { maxWait: 2000 }
    )
  }
}
</script>
```

```vue
<template>
  <div class="flex items-center flex-grow">
    <button
      @mousedown="startClap"
      @mouseup="stopClap"
      title="Hold to clap"
      class="border-white border p-2 rounded hover:border-yellow relative"
    >
      <div
        class="flex flex-row items-center pointer-events-none justify-between"
      >
        <div>
          <img
            id="logo"
            src="~/assets/img/clap.svg"
            alt="Clapping Hands"
            class="h-12"
          />
        </div>
        <div class="text-center ml-4 w-10">
          {{ currentCounter }}
        </div>
      </div>
    </button>

    <div class="ml-4">
      <transition
        enter-active-class="transition-all"
        leave-active-class="transition-all"
        enter-class="scale-70"
        enter-to-class="opacity-1 scale-100"
        leave-class="opacity-1 scale-100"
        leave-to-class="scale-130 opacity-0"
      >
        <span
          v-show="newCountVisible"
          class="text-sm text-gray-900 inline-block w-12 rounded-lg bg-yellow text-center"
          >+ {{ newCount }}</span
        >
      </transition>
    </div>
  </div>
</template>

<script>
import debounce from 'lodash/debounce'

const speedMap = {
  slow: 300,
  medium: 100,
  fast: 40
}

export default {
  props: {
    id: {
      type: String,
      required: true
    }
  },

  data() {
    return {
      currentCounter: 0,
      newCount: 0,
      newCountVisible: false,
      newCountBuffer: 0,
      clickMeta: {
        start: undefined,
        timeout: undefined,
        holdTimeout: undefined,
        lastSpeed: undefined
      }
    }
  },

  async mounted() {
    try {
      const postMetaResponse = await fetch(
        `/.netlify/functions/get-post-meta/${this.id}`
      )
      const postMeta = await postMetaResponse.json()
      this.currentCounter = postMeta.data.claps
    } catch (e) {
      console.error(
        'This Error happend when trying to fetch the original post meta.',
        e
      )
    }
  },

  methods: {
    holdCounter() {
      if (this.clickMeta.lastSpeed === 'fast') {
        const now = new Date()
        const diff = now - this.clickMeta.start

        if (diff > 2500) this.clickMeta.lastSpeed = 'fast'
        else if (diff > 1000) this.clickMeta.lastSpeed = 'medium'
        else this.clickMeta.lastSpeed = 'slow'
      }

      this.newCount = this.newCount + speedMap[this.clickMeta.lastSpeed]
      this.clickMeta.timeout = setTimeout(this.holdCounter, 500)
    },

    startClap() {
      this.clickMeta.start = new Date()
      this.newCountVisible = true
      this.newCount = 1
      this.clickMeta.timeout = setTimeout(this.holdCounter, 500)
    },

    stopClap() {
      for (const key of ['timeout', 'holdTimeout']) {
        clearTimeout(this.clickMeta[key])
        this.clickMeta[key] = undefined
      }
      for (const key of ['start', 'lastSpeed']) {
        this.clickMeta[key] = undefined
      }
      this.newCountVisible = false
      this.currentCounter = this.currentCounter + this.newCount
      this.newCountBuffer = this.newCountBuffer + this.newCount

      this.syncClaps()
    },

    syncClaps: debounce(
      async function() {
        const claps = this.newCountBuffer
        this.newCountBuffer = 0
        const postMetaResponse = await fetch(
          `/.netlify/functions/update-post-meta/${this.id}`,
          {
            body: JSON.stringify({ claps }),
            method: 'POST'
          }
        )
        const postMeta = await postMetaResponse.json()
        this.currentCounter = postMeta.data.claps
      },
      500,
      { maxWait: 2000 }
    )
  }
}
</script>
```

```
syncClaps: debounce(
      async function() {
        const claps = this.newCountBuffer
        this.newCountBuffer = 0
        const postMetaResponse = await fetch(
          `/.netlify/functions/update-post-meta/${this.id}`,
          {
            body: JSON.stringify({ claps }),
            method: 'POST'
          }
        )
        const postMeta = await postMetaResponse.json()
        this.currentCounter = postMeta.data.claps
      },
      500,
      { maxWait: 2000 }
    )
```

/.netlify/functions/get-post-meta.js

```
client.query(q.Get(q.Match(q.Index('posts_meta_by_id'), postID)))
```

/.netlify/functions/update-post-meta.js

```
client
    .query(q.Get(q.Match(q.Index('posts_meta_by_id'), postID)))
    .then(response => {
      const ref = response.ref
      const mergedData = mergePostMeta(response.data, clientData)

      client
        .query(q.Update(ref, { data: mergedData }))
        .then(response => {
          callback(null, {
            statusCode: 200,
            body: JSON.stringify(response)
          })
        })
        .catch(error => {
          requestFailed(error)
        })
    })
    .catch(error => {
      requestFailed(error)
    })
```

/.netlify/functions/update-post-meta.js

```
client
    .query(q.Get(q.Match(q.Index('posts_meta_by_id'), postID)))
    .then(response => {
      const ref = response.ref
      const mergedData = mergePostMeta(response.data, clientData)

      client
        .query(q.Update(ref, { data: mergedData }))
        .then(response => {
          callback(null, {
            statusCode: 200,
            body: JSON.stringify(response)
          })
        })
        .catch(error => {
          requestFailed(error)
        })
    })
    .catch(error => {
      requestFailed(error)
    })
```

/.netlify/functions/update-post-meta.js

```
client
    .query(q.Get(q.Match(q.Index('posts_meta_by_id'), postID)))
    .then(response => {
      const ref = response.ref
      const mergedData = mergePostMeta(response.data, clientData)

      client
        .query(q.Update(ref, { data: mergedData }))
        .then(response => {
          callback(null, {
            statusCode: 200,
            body: JSON.stringify(response)
          })
        })
        .catch(error => {
          requestFailed(error)
        })
    })
    .catch(error => {
      requestFailed(error)
    })
```

Data Preparations

Data

Heads Up

Serverless

```
node scripts/createPost.js
```

/scripts/createPost.js

```
// Get process.stdin as the standard input object.
const standardInput = process.stdin
const path = require('path')
const fs = require('fs').promises
const pathForNewPost = path.join(__dirname, '..', 'content')

// Set input character encoding.
standardInput.setEncoding('utf-8')

// Prompt user to input data in console.
console.log('New Post Title:')

// When user input data and click enter key.
standardInput.on('data', async title => {
  if (title.trim().length > 0) {
    await createNewPost(title)
    console.log('New Post Created!')
    process.exit()
  }
})

async function createNewPost(title) {
// Create Files, Folders, …
}
```

/content/posts/vuejs-amsterdam.md

```
---
title: Vuejs-Amsterdam

id: 45ab1ab2-0bd9-4e31-9a39-7f14f9ba07fa
slug: 45ab1ab2-0bd9-4e31-9a39-7f14f9ba07fa-vuejs-amsterdam
published: 20.01.2020
updated: 20.01.2020
tags:
  - tag
---
```

```
ntl dev:exec node scripts/faunaCreatePostMeta.js {postId}
```

/scripts/faunaCreatePostMeta.js

```
const faunadb = require('faunadb')

const postId = process.argv[2]

if (!postId) return console.log('Please pass the id of the post')

const q = faunadb.query

const client = new faunadb.Client({
  secret: process.env.FAUNADB_SERVER_SECRET
})

async function createPostsMeta(id) {
  const data = {
    id,
    claps: 0
  }

  try {
    const response = await client.query(
      q.Create(q.Collection('posts_meta'), { data })
    )
    console.log(JSON.stringify(response, null, 2))
  } catch (e) {
    console.log('Post Creation on Fauna Failed')
  }
}

createPostsMeta(postId)
```

/scripts/faunaCreatePostMeta.js

```
const faunadb = require('faunadb')

const postId = process.argv[2]

if (!postId) return console.log('Please pass the id of the post')

const q = faunadb.query

const client = new faunadb.Client({
  secret: process.env.FAUNADB_SERVER_SECRET
})

async function createPostsMeta(id) {
  const data = {
    id,
    claps: 0
  }

  try {
    const response = await client.query(
      q.Create(q.Collection('posts_meta'), { data })
    )
    console.log(JSON.stringify(response, null, 2))
  } catch (e) {
    console.log('Post Creation on Fauna Failed')
  }
}

createPostsMeta(postId)
```

/scripts/faunaCreatePostMeta.js

```
const faunadb = require('faunadb')

const postId = process.argv[2]

if (!postId) return console.log('Please pass the id of the post')

const q = faunadb.query

const client = new faunadb.Client({
  secret: process.env.FAUNADB_SERVER_SECRET
})

async function createPostsMeta(id) {
  const data = {
    id,
    claps: 0
  }

  try {
    const response = await client.query(
      q.Create(q.Collection('posts_meta'), { data })
    )
    console.log(JSON.stringify(response, null, 2))
  } catch (e) {
    console.log('Post Creation on Fauna Failed')
  }
}

createPostsMeta(postId)
```

/scripts/faunaCreatePostMeta.js

```
const faunadb = require('faunadb')

const postId = process.argv[2]

if (!postId) return console.log('Please pass the id of the post')

const q = faunadb.query

const client = new faunadb.Client({
  secret: process.env.FAUNADB_SERVER_SECRET
})

async function createPostsMeta(id) {
  const data = {
    id,
    claps: 0
  }

  try {
    const response = await client.query(
      q.Create(q.Collection('posts_meta'), { data })
    )
    console.log(JSON.stringify(response, null, 2))
  } catch (e) {
    console.log('Post Creation on Fauna Failed')
  }
}

createPostsMeta(postId)
```

Static Sites are Awesome

…with serverless they get even more awesome!

Firebase

https://firebase.google.com

Guest

Firebase Products Use Cases Pricing Docs Support Search Language Go to console Sign in

Firebase helps mobile and web app teams succeed

Get started

Watch the video

Build apps fast, without managing infrastructure

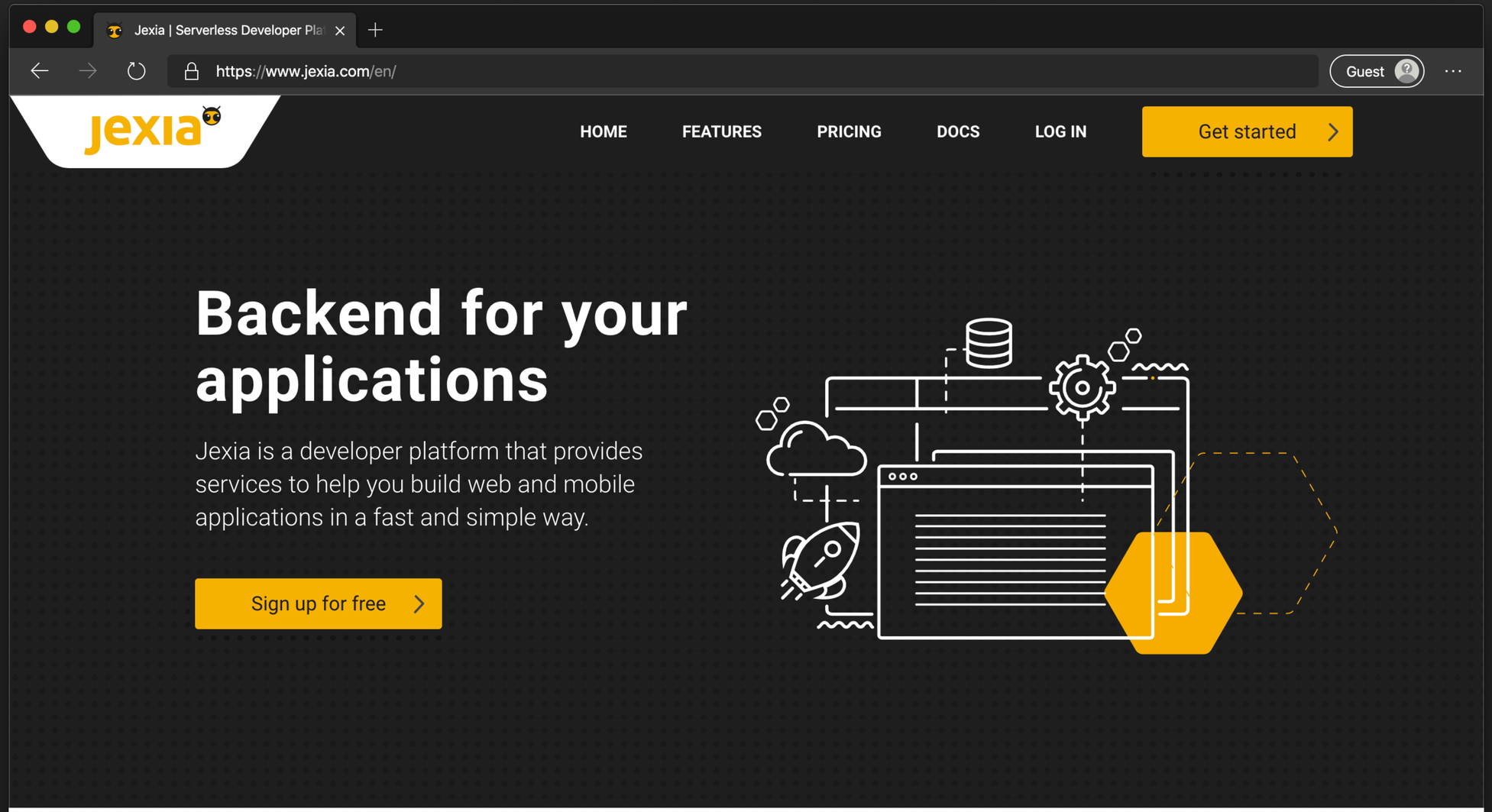
Firebase gives you functionality like

Backed by Google, trusted by top apps

Firebase is built on Google infrastructure

One platform, with products that work better together

Codebryo



Code snippets to use Jexia

Paste the codes into your IDE, tweak with your project info

More personal pages again.

Thank You

 @codebryo

Codebryo