

Offline-first PWA for Github actions

Install main project

1. Install Sveltekit

```
npm create svelte@latest A09
cd A09
npm install
```

2. Install Tailwind

```
npx svelte-add@latest tailwindcss
npm install
```

3. Adjust tailwind.config.cjs

```
/** @type {import('tailwindcss').Config}*/
const config = {
  content: ['./src/**/*.html,js,svelte,ts'],

  theme: {
    extend: {
      zIndex: {
        '100': '100',
        '1000': '1000',
        '2000': '2000',
        '3000': '3000',
        '5000': '5000',
        '10000': '10000',
        '20000': '20000',
      }
    }
  },

  plugins: []
};

module.exports = config;
```

4. Adjust app.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="theme-color" content="#FFFFFF">
    <link rel="icon" href="%sveltekit.assets%/favicon.png" />
    <title>A09</title>
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=0, interactive-widget=resizes-visual"
    >
    %sveltekit.head%
  </head>
  <body data-sveltekit-preload-data="hover" class="overscroll-contain">
    <div style="display: contents">%sveltekit.body%</div>
  </body>
</html>
```

5. Init Git

```
git init
git branch -M main
git add .
git commit -m "initial commit"
```

Set up SPA and Github action

1. Install adapter-static

<https://kit.svelte.dev/docs/adapter-static>

```
npm install -D @sveltejs/adapter-static
```

svelte.config.js

```
import { vitePreprocess } from '@sveltejs/vite-plugin-svelte';
import adapter from '@sveltejs/adapter-static';

/** @type {import('@sveltejs/kit').Config} */
const config = {
  kit: {
    adapter: adapter({
      fallback: '404.html'
    })
  }
};
```

```
    }},
    paths: {
      base: process.argv.includes('dev') ? '' : process.env.BASE_PATH
    },
  },
  preprocess: [vitePreprocess({})]
};

export default config;
```

2. Add +layout.js to routes

```
export const prerender = true;
export const ssr = false;
```

3. Add deploy.yml to A09/.github/workflows

<https://kit.svelte.dev/docs/adaptor-static>

```
name: Deploy to GitHub Pages

on:
  push:
    branches: 'main'

jobs:
  build_site:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Install Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm

      - name: Install dependencies
        run: npm install

      - name: build
        env:
          BASE_PATH: '${{ github.event.repository.name }}'
        run: |
          npm run build

      - name: Upload Artifacts
        uses: actions/upload-pages-artifact@v3
```

```
with:
  # this should match the `pages` option in your adapter-static options
  path: 'build/'

deploy:
  needs: build_site
  runs-on: ubuntu-latest

permissions:
  pages: write
  id-token: write

environment:
  name: github-pages
  url: ${ steps.deployment.outputs.page_url }}

steps:
  - name: Deploy
    id: deployment
    uses: actions/deploy-pages@v4
```

4. Create new Github repo named A09

5. Add to remote repo to Git and upload to Github

```
git remote add origin git@github.com:fg5002/A09.git
git add .
git commit -m "... "
npm run build
git push -u origin main
```

6. Github A09/settings/pages -> Source: Github Actions

Set up PWA

1. add icons folder to static folder
2. add manifest.json to static folder

```
{
  "id": "ffd-A09-pwa",
  "short_name": "A09",
  "start_url": "/A09/",
  "scope": "/A09/",
  "display": "standalone",
  "orientation": "portrait",
  "theme_color": "#A3E635",
  "background_color": "#ffffff",
```

```
"dir": "ltr",
"lang": "en",
"icons": [
  {
    "src": "icons/icon192.png",
    "sizes": "192x192",
    "type": "image/png"
  },
  {
    "src": "icons/icon512.png",
    "sizes": "512x512",
    "type": "image/png"
  }
]
}
```

3. add manifest link to app.html

```
...
<link rel="manifest" href="%sveltekit.assets%/manifest.json" />
<link rel="icon" href="%sveltekit.assets%/favicon.png" />
<title>A09</title>
...
```

4. add service-worker.js to scr folder

<https://kit.svelte.dev/docs/service-workers>

```
/// <reference types="@sveltejs/kit" />
import { build, files, version } from '$service-worker';

// Create a unique cache name for this deployment
const CACHE = `cache-${version}`;

const ASSETS = [
  ...build, // the app itself
  ...files // everything in `static`
];

self.addEventListener('install', (event) => {
  // Create a new cache and add all files to it
  async function addFilesToCache() {
    const cache = await caches.open(CACHE);
    await cache.addAll(ASSETS);
  }

  event.waitUntil(addFilesToCache());
});
```

```
self.addEventListener('activate', (event) => {
  // Remove previous cached data from disk
  async function deleteOldCaches() {
    for (const key of await caches.keys()) {
      if (key !== CACHE) await caches.delete(key);
    }
  }

  event.waitUntil(deleteOldCaches());
});

self.addEventListener('fetch', (event) => {
  // ignore POST requests etc
  if (event.request.method !== 'GET') return;

  async function respond() {
    const url = new URL(event.request.url);
    const cache = await caches.open(CACHE);

    // `build`/`files` can always be served from the cache
    if (ASSETS.includes(url.pathname)) {
      const response = await cache.match(url.pathname);

      if (response) {
        return response;
      }
    }

    // for everything else, try the network first, but
    // fall back to the cache if we're offline
    try {
      const response = await fetch(event.request);

      // if we're offline, fetch can return a value that is not a Response
      // instead of throwing - and we can't pass this non-Response to
      respondWith
      if (!(response instanceof Response)) {
        throw new Error('invalid response from fetch');
      }

      if (response.status === 200) {
        cache.put(event.request, response.clone());
      }

      // cross-origin isolation using COOP and COEP headers
      const newHeaders = new Headers(response.headers);
      newHeaders.set("Cross-Origin-Embedder-Policy", "require-corp");
      newHeaders.set("Cross-Origin-Opener-Policy", "same-origin");

      const moddedResponse = new Response(response.body, {
        status: response.status,
        statusText: response.statusText,
        headers: newHeaders,
      });
    }
  }
});
```

```
    });

    return moddedResponse;
    // headers end

    //return response;

  } catch (err) {
    const response = await cache.match(event.request);

    if (response) {
      return response;
    }

    // if there's no cache, then just error out
    // as there is nothing we can do to respond to this request
    throw err;
  }
}

event.respondWith(respond());
});
```

5. push up modifications

```
git add .
git commit -m "... "
npm run build
git push
```

Calendar and time picker

```
npm install svelte-picker
```

Create map

1. install Leaflet

```
npm install leaflet
```

2. install markercluster

```
npm install leaflet.markercluster
```

3. install featuregroup.subgroup

```
npm install leaflet.featuregroup.subgroup --save
```

4. install necessary turf packages

```
npm install @turf/bearing @turf/destination @turf/distance @turf/midpoint  
@turf/point-on-feature @turf/nearest-point-on-line @turf/explode @turf/helpers
```

5. add map folder to routes

6. add +page.svelte to map folder

7. add +page.js to map folder

```
export const prerender = false;
```

Set up SQLocal

1. Install vite-plugin-cross-origin-isolation

```
npm i -D vite-plugin-cross-origin-isolation
```

2. Install SQLocal

```
npm install sqlocal
```

3. vite.config.js

```
import { sveltekit } from '@sveltejs/kit/vite';  
import { defineConfig } from 'vite';  
import crossOriginIsolation from 'vite-plugin-cross-origin-isolation';  
  
export default defineConfig({  
  plugins: [  

```



```
      sveltekit(),
      crossOriginIsolation()
    ],
    optimizeDeps: {
      exclude: ['sqllocal'],
    },
  });
```

4. Insert crossOrigin : true into every tilelayer

```
<TileLayer
  name={'OSM'}
  url={'https://tile.openstreetmap.org/{z}/{x}/{y}.png'}
  options={{
    minZoom: 7,
    maxZoom: 19,
    attribution: '&copy; OpenstreetMap',
    crossOrigin : true
  }}
  selected
/>
```
