Vault 101

King's Digital Lab

Jun 15, 2023

CONTENTS:

This documentation is structured following the Divio's documentation system.

CHAPTER

ONE

TUTORIALS

For more information see Tutorials.

CHAPTER

TWO

HOW-TO GUIDES

Here you will find short recipes to solve…

For more information see How-to guides.

2.1 Cookies disclaimer

This document describes how to add a *Cookies disclaimer* box to a project. The source code for this how-to is available in GitHub.

1. Add the *HTML* snippet to every page on the project.

```
<div id="cookies-disclaimer">
  <p>
    We use cookies to track usage and preferences in accordance with our
    <a href="/privacy-policy/">Privacy &amp;amp; Cookie Policy</a>
  </p>
  <p>
    <button aria-label="Dismiss alert" type="button">I understand</button>
  </p>
</div>
```

2. Add the *JavaScript* function to the project.

```
function cookies() {
  const disclaimer = document.querySelector('#cookies-disclaimer')

  if (localStorage.getItem('cookies-accepted')) {
    hideCookiesDisclaimer(disclaimer)
    return
  }

  const button = disclaimer.querySelector('button')

  button.addEventListener('click', (event) => {
    localStorage.setItem('cookies-accepted', true)
    hideCookiesDisclaimer(disclaimer)
  })
}

function hideCookiesDisclaimer(disclaimer) {
  disclaimer.className = ' hidden'
}
```

3. Call the `cookies()` function from the main project script.
4. Add a Privacy and Cookie Policy to the project.

2.2 Wait for Elasticsearch to be ready in Docker/Compose

The Django service dependencies set with depends_on in Docker compose do not wait for the elasticsearch service to be *ready*, they only wait until the service is started.

To guarantee that *elasticsearch* is ready add the following snippet to the *Django* `entrypoint` script:

```
if [ -z "${ELASTICSEARCH_HOST:-}" ]; then
    export ELASTICSEARCH_HOST="elasticsearch:9200"
fi

elasticsearch_ready() {
python << END
import sys

from elasticsearch_dsl import connections
from elasticsearch.exceptions import ConnectionError

try:
    c = connections.create_connection(hosts=["${ELASTICSEARCH_HOST}"], timeout=10)
    c.info()
except (ConnectionError, ConnectionRefusedError) as e:
    sys.exit(-1)

sys.exit(0)
END
}
until elasticsearch_ready; do
  >&2 echo 'Waiting for Elasticsearch to become available...'
  sleep 1
done
>&2 echo 'Elasticsearch is available'
```

CHAPTER

THREE

REFERENCE GUIDES

For more information see Reference guides.

3.1 Hydra/JAMStack

3.1.1 Modules

Projects based on this architecture will have at most three main modules:

- cms (backend)
- etl (data processing module)
- *frontend*

Set up as a monorepo using the npm workspaces setting in *package.json*:

```
"workspaces": [
  "cms",
  "etl",
  "frontend"
]
```

Directory structure

- **.envs/** environment files for each of the modules, *.example* files should be under version control to help setting up new instances
  - .cms.example
  - .etl.example
  - .frontend.example
- **.github/workflows** CI/CD pipelines
- **cms/** the cms module
- **docker/** files for setting up Docker provisioning, as needed
  - **local/module_name/***
  - **production/module_name/***
- **etl/** the data processing module

- *frontend/* the frontend module
- *.eslintrc.json* to specify the linting rules
- **.nvmrc** to specify the node version used to run the project
- **.prettierignore** to specify which files should not be automatically formatted
- CHANGELOG.md to record project changes
- **LICENSE**
- **README.md**
- **docker-compose.override.yaml.example** Docker compose file for local overrides, should be copied and renamed without *.example*
- **docker-compose.yaml** Docker compose file
- **package.json**

Note: Docker is only needed if using the cms, and potentially etl modules. If the project only has the frontend module Docker should not be needed.

The highlevel directory structure for each module/workspace, by default, follows the *standard* for JavaScript packages:

- **public/**
- **src/**
- **tests/**
- **package.json**

Note: This directory structure may not apply to all the workspaces, for example, if using Directus for the cms module the directory structure should follow the structure of the Directus project, which by default doesn't need a *src* directory.

3.1.2 Tooling

These are used to guarantee consistency across projects and should be set up at the root of the project:

- eslint to find issues in the code set up with the recommended settings and using prettier for formatting.

```
{
  "env": {
    "es6": true,
    "node": true,
    "jest": true
  },
  "extends": [
    "eslint:recommended",
    "prettier"
  ],
  "parser": "@babel/eslint-parser",
  "parserOptions": {
    "requireConfigFile": false
  }
}
```

- *.nvmrc* to specify a node version
- prettier for code formatting

Dependencies

```
"devDependencies": {
  "@babel/eslint-parser": "^7.18.2",
  "@babel/eslint-plugin": "^7.17.7",
  "eslint": "^8.17.0",
  "eslint-config-prettier": "^8.5.0",
  "prettier": "^2.6.2",
  "prettier-plugin-sort-imports": "^1.7.0",
  "simple-git-hooks": "^2.8.0",
  "vscode-langservers-extracted": "^4.2.1"
}
```

Automation

Both linting and formatting should be run as a pre-commit hook set up in *package.json*, by using the simple-git-hooks package.

```
"simple-git-hooks": {
  "pre-commit": "npx lint-staged"
},
"lint-staged": {
  "*.{js}": "npm run format",
  "*.{json,md,yaml}": "npm run prettier:fix"
}
```

Scripts

The top seven sample scripts are mostly useful for when one of the modules in the project uses Docker, they provide a shortcut to interact with the containers.

```
"scripts": {
  "compose": "trap 'echo Stopped; exit 0' SIGINT; docker-compose",
  "up": "npm run compose up -- --build",
  "down": "npm run compose down",
  "exec": "npm run compose exec ${npm_config_service}",
  "pkg": "npm run exec npm",
  "cms:snapshot": "npm run pkg --service=cms run snapshot:create",
  "cms:snapshot:apply": "npm run pkg --service=cms run snapshot:apply ./snapshots/$
↪{npm_config_snapshot}.yaml",
  "lint": "eslint **/src **/tests",
  "lint:fix": "npm run lint -- --fix",
  "prettier": "prettier . --check",
  "prettier:fix": "npm run prettier -- --write",
  "format": "npm run prettier:fix && npm run lint:fix",
  "test": "npm run test --workspaces --if-present"
}
```

3.1.3 Frontend module

By default the frontend module should be built using 11ty.

Directory structure

- **../.github/workflows/frontend.yaml** GitHub pipeline file to build GitHub pages
- .eleventy.js 11ty configuration file
- *.env* environment file with settings to build the frontend
- *_site/* output directory for the built site (not under version control)
- *package.json*
- *public/* contains files that are not used by the build process
  - **assets/*** images, pdfs, etc.
  - **robots.txt**
- **src/** input/src directory for the build process
  - **_data/** global data/collections for the project
    - **eleventyComputed.js** computed fields available to every layout
    - **config.json** settings for the seo plugin and other site metadata
  - **_includes/** contains layouts, include files, extends files, partials, or macros
    - layouts/ contains layout templages
    - **macros/** contains reusable chunks, similar to a function does in programming languages
    - **partials/** contains layouts partials/fragments
  - **assets/**
    - **stylesheets/** SCSS files that will be built by 11ty

Templates

Component-based development for template reuse, one possibility is to use Nunjucks macros:

- https://iainbean.com/posts/2020/flexible-components-in-eleventy-with-nunjucks-macros/
- https://www.trysmudford.com/blog/encapsulated-11ty-components/

> **Warning:** 11ty supports multiple template languages, ~10! I have settled on Nunjucks because it seemed that was the language mostly used in 11ty projects. The other reason I have not used liquid is because the JavaScript implementation is different from the Ruby one used with Jekyll. One issue I have with it is that both the *empty string* and *0* are truthy in *liquid*.

Scripts

Execute them with *npm run*.

```
"scripts": {
  "build": "eleventy",
  "dev": "eleventy --serve",
  "debug": "DEBUG=Eleventy* npm run dev",
  "index": "npx pagefind --source _site"
}
```

Metadata

Post

```
---
title: Title of the post
# the subtitle field is optional and depends on the project
subtitle: A fancy subtitle
# posts can have multiple tags
tags:
# post is a default tag
  - post
# one tag per line
  - one
# spaces are allowed in tag names
  - multiple words
# posts can have multiple authors
authors:
# id possible use user ids
  - auser
# otherwise full names, don't mix both approaches in the same project
  - Aa User
# if the post filename starts with the date 2022-11-01_post-title.md, then the date is
# optional; the date in the frontmatter has higher precendence
date: 2022-09-07
# an execerpt, optional depending on the project
excerpt: Lorem ipsum dolor sit amet, qui minim labore adipisicing minim sint cillum
  sint consectetur cupidatat.
# featured image
feature:
# image URL
  image: images/IDH_banner.original.jpg
# image description
  description: Indigenous Digital Humanities Banner
---

Lorem ipsum dolor sit amet, qui minim labore adipisicing minim sint cillum sint
consectetur cupidatat.
```

Plugins

The 11ty website has a list of both core and community plugins.

Currently used

- eleventy-navigation plugin for creating hierarchical navigation and breadcrumbs
- seo plugin to generate meta tags for improved SEO
- toc plugin to generate table of contents from page headers

Potentially useful

- https://www.11ty.dev/docs/plugins/image/
- https://www.11ty.dev/docs/plugins/i18n/
- https://www.npmjs.com/package/eleventy-plugin-html-validate
- https://www.npmjs.com/package/@quasibit/eleventy-plugin-sitemap

Search

Search can easily be added to an 11ty project by using the pagefind package.

3.2 Useful web map service providers for Leaflet.js and OpenLayers

Base maps are essential for contextualising spatial data in a web map. However, layer providers often change their terms or their APIs and this can result in a map layer becoming unavailable and leaving a blank base layer.

This page provides 2 options for default map providers that can be used according to their T&Cs allowing that our sites don't normally drive a huge volume of traffic.

3.2.1 Option 1 - OpenStreetMap

OpenStreetMap provides a tile service with a usage policy detailed here: https://operations.osmfoundation.org/policies/tiles/

NB. This option should not be used if we anticipate heavy traffic, i.e through an app or very busy website.

```
// By default subdomains denoted by {s} are [a,b,c], and are used to speed up tile␣
↪loading

newTileLayer = L.TileLayer("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
↪{attribution:"© OpenStreetMap contributors"})
```

3.2.2 Option 2 - ArcServer

There are several backdrops that can be selected from the list here: https://server.arcgisonline.com/ArcGIS/rest/services/

The *World Streets* layer can be used in web maps according to the terms found here https://www.arcgis.com/home/item.html?id=3b93337983e9436f8db950e38a8629af and should be attributed as follows:

Sources: Esri, HERE, Garmin, USGS, Intermap, INCREMENT P, NRCAN, Esri Japan, METI, Esri China (Hong Kong), NOSTRA, © OpenStreetMap contributors, and the GIS User Community

Tile URL

https://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/tile/{z}/{y}/{x}

```
newTileLayer = L.TileLayer("http://server.arcgisonline.com/ArcGIS/rest/services/World_
↪Street_Map/MapServer/tile/{z}/{y}/{x}",{attribution:'<a href="https://services.
↪arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer">World Street Map</
↪a> Esri, HERE, Garmin, USGS, Intermap, INCREMENT P, NRCAN, Esri Japan, METI, Esri␣
↪China (Hong Kong), NOSTRA, © OpenStreetMap contributors, and the GIS User Community
↪'})
```

CHAPTER

FOUR

EXPLANATION

For more information see Explanation.

CHAPTER

FIVE

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

5.1 [Unreleased] - yyyy-mm-dd

5.1.1 Added

- Cookies component.
- Initial documentation structure.
- How-to: wait for Elasticsearch to be ready in Docker.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search