Studio: Optimization & CI / CD

Overview: Assessed Lab

SUBMISSION

- Submit the all tasks as a single PDF file on:

  Moodle -> FIT5032 -> Grades -> Assessed Lab 12

  https://learning.monash.edu/mod/assign/view.php?id=4673266

- Please submit the file by Week 8 Wednesday, 23:55 (Melbourne Time). Late submissions will result in a penalty of 10% marks per day.

EFOLIO TASK 12.1 (PASS AND CREDIT LEVEL)

WHAT TO SUBMIT (INDIVIDUAL):

- Link to your deployed project
- **Screenshot**: Screenshot of your project Github action page that indicate the workflow is running

EVALUATION CRITERIA:

- All activities are completed properly. The screenshot resolution is clear. Link is working

EFOLIO TASK 12.2 (DISTINCTION AND HIGH DISTINCTION LEVEL)

WHAT TO SUBMIT (INDIVIDUAL):

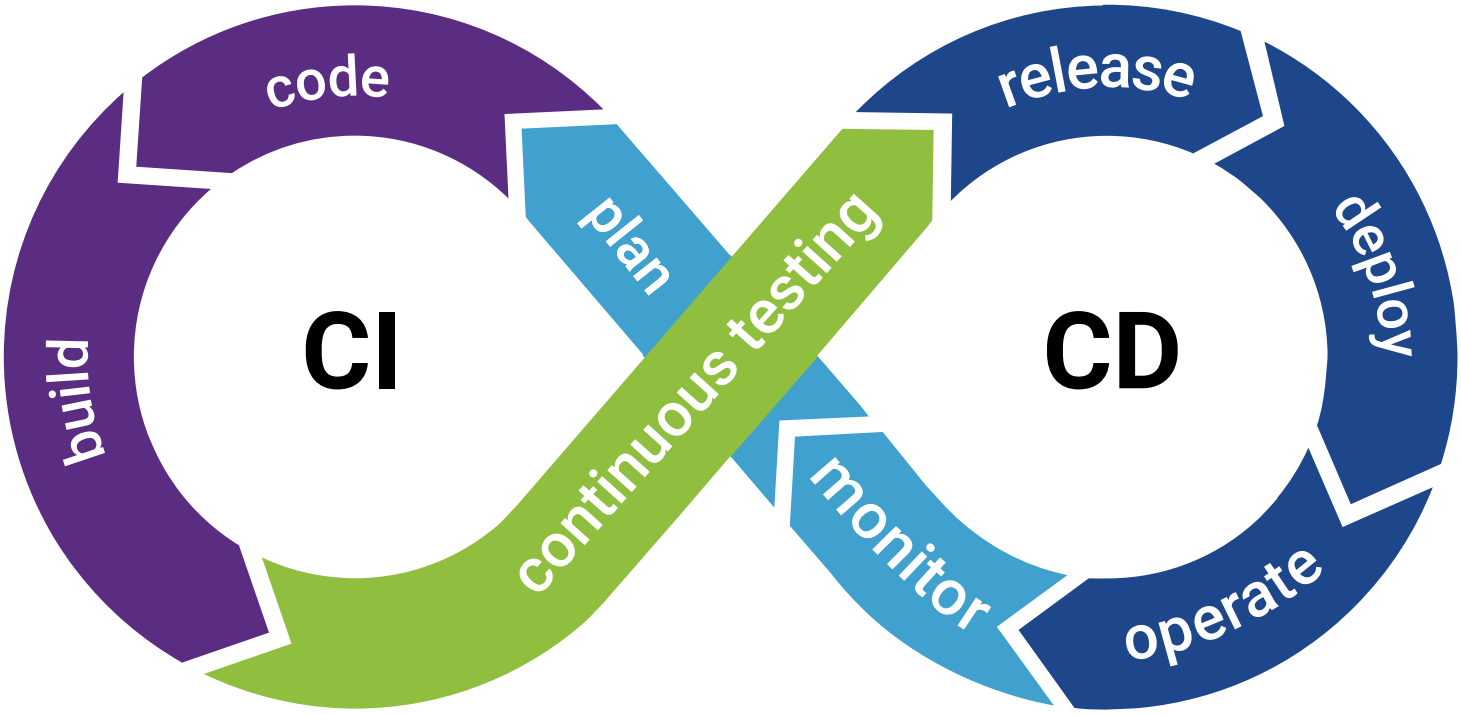
- **Text answer 1:** Perform research on Time To First Byte (TTFB) and share how to reduce TTFB
- **Text answer 2:** Perform research on Hotlink and share how to avoid your site being hotlinked

EVALUATION CRITERIA:

- All activities are completed properly. The screenshot resolution is clear.

12.1 CI/CD with GitHub Actions concept

12.1 CI/CD with GitHub Actions concept



Source

The workflow we are going to set up in 12.2 is a basic CI/CD pipeline:

- **Continuous Integration (CI)**: Every time you push to the `main` branch, the workflow automatically builds your project, ensuring that the code compiles without errors.
- **Continuous Deployment (CD)**: If the build is successful, the workflow automatically deploys the built files to the `gh-pages` branch, which serves your application on GitHub Pages.

This setup allows you to:

- Automatically test your code with each push (you can add testing steps to the workflow).
- Ensure your main branch always contains deployable code.
- Automate the deployment process, reducing manual errors and saving time.

Best Practices

- **Use Environment Variables**: For sensitive information like API keys, use GitHub Secrets and pass them to your workflow.
- **Add Testing**: Include a testing step in your workflow to ensure code quality.
- **Branch Protection**: Set up branch protection rules for your `main` branch to require pull

request reviews before merging.
- **Versioning**: Consider using semantic versioning and creating releases for significant updates.

In the next step, you will be able to follow instructions to understand how you should setup the Github page for hosting with CI & CD integration

12.2 Deploying Vue.js Apps to GitHub Pages with CI/CD

12.1 Introduction & Prerequisites

This guide will walk you through the process of deploying a Vue.js application to GitHub Pages and setting up a Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub Actions. This approach is particularly relevant as it demonstrates practical application of web development and DevOps concepts.

Prerequisites

- A Vue.js 3 project created with Vue CLI or Vite (Which you have done since beginning of semester)
- A GitHub account
- Git installed on your local machine

12.2 Steps for setting up CI & CD

1. Prepare a Vue.js Project, which is from one of the activities of Studio 1 to 11.

2. Create a `vue.config.js` file in the root of your project (if it doesn't exist) and add:

> The following configuration enables your project to be change according to different model, ,so in normal development model, The base URL will be `/`, so your app will be accessible at `http://localhost:XXX/`. However, when change to production model, then your project will be accessible by visiting `http://example.com/repository-name/`

```
module.exports = {
  publicPath: process.env.NODE_ENV === 'production'
    ? '/<repository-name>/'
    : '/'
}
```

Replace `<repository-name>` with your GitHub repository name.

> Step 3 and step 4 is for new project, if you already have Git repo, you can skip this, this is kept here so that you can still follow the step in the future if you are creating a new project

3. Create a GitHub Repository (This step is optional if you already have Git repo)

- Go to GitHub and create a new repository.

> If your project is not able to be deployed, you might need to try to set your project to public repository

instead. Otherwise, proceed with the following steps

4. Push Your Project to GitHub

- Initialize Git in your project folder (if not already done):

```
git init
```

- Add your files and commit:

```
git add .
git commit -m "Initial commit"
```

- Add your GitHub repository as a remote and push:

```
git remote add origin https://github.com/<username>/<repository-name>.git
git branch -M main
git push -u origin main
```

5. Set Up GitHub Actions for CI/CD

In case you do not know about Github Action (https://github.com/features/actions), it is a platform for continuous integration (CI) and continuous delivery (CD)

.yml file are called YAML files, which is usually a txt file that is commonly used for configurations , file structure is the code provided below

- In your project root, create a `.github/workflows` directory.
- Inside this directory, create a file named `deploy.yml`.
- Add the following content to `deploy.yml`:

The checkout step refers to the process of retrieving the code from your GitHub repository and making it available in the workflow's environment

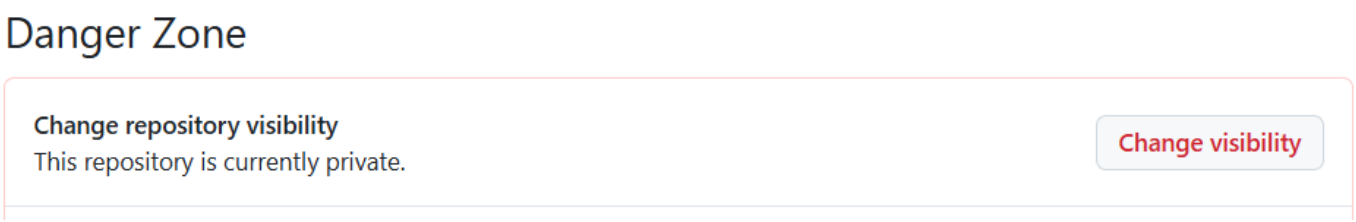
```
name: Deploy to GitHub Pages

on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout
      uses: actions/checkout@v4
```

```
    - name: Setup Node.js
      uses: actions/setup-node@v4
      with:
        node-version: '20'

    - name: Install dependencies
      run: npm ci

    - name: Build
      run: npm run build

    - name: Deploy
      uses: JamesIves/github-pages-deploy-action@4.1.4
      with:
        branch: gh-pages
        folder: dist

permissions:
  contents: write
```

This workflow will automatically build and deploy your Vue.js app to GitHub Pages whenever you push changes to the `main` branch.

6. Enable GitHub Pages

- Go to your repository on GitHub.
- Go to "Settings"
- Change the repository to "public":



-
- Click on "Settings" > "Pages".
- Under "Source", select the `gh-pages` branch.
- Click "Save".

Step 6: Trigger the Deployment

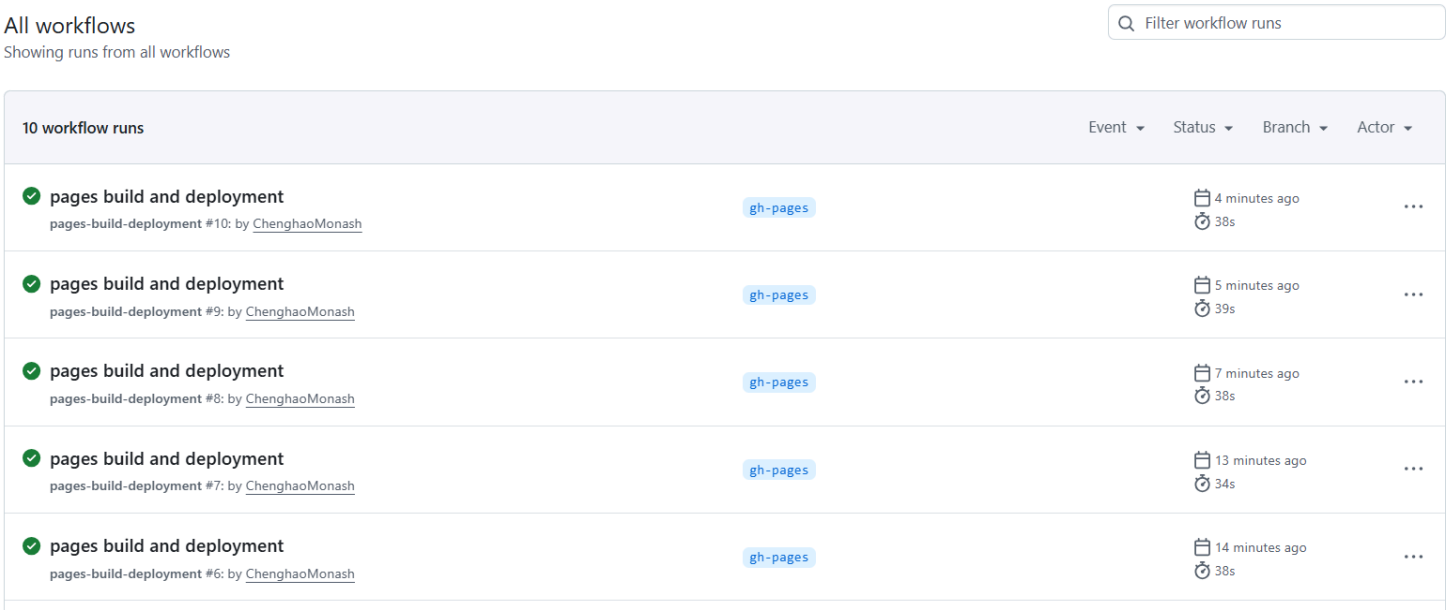
- Make a small change to your project.
- Commit and push the change:

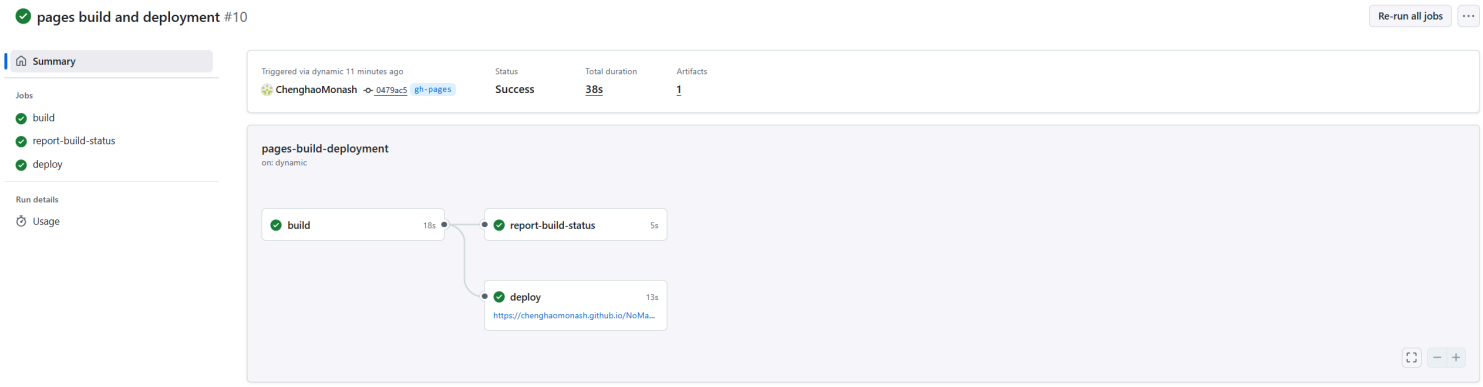
> Please note the following is git command for commit and push, you can use your visual studio code to do it

```
git add .
git commit -m "Trigger deployment"
```

```
git push
```

- Go to your repository on GitHub, click on the "Actions" tab, and you should see your workflow running.





Step 7: Access Your Deployed Application

Usually, once the workflow completes successfully, your app will be available at: `https://<username>.github.io/<repository-name>/`

However, as the latest version of Github and VUE are not incompatible, your app may not be correctly displayed in the browser. This will not decrease your marks.

12.3 Optimisation for web project

12.3 Optimisation for web project

In this section, we will share various ways to optimize your project so that your system can be robust:

We will introduce 5 techniques, you do not need to do anything with your project, but try to remember the theories you can implement

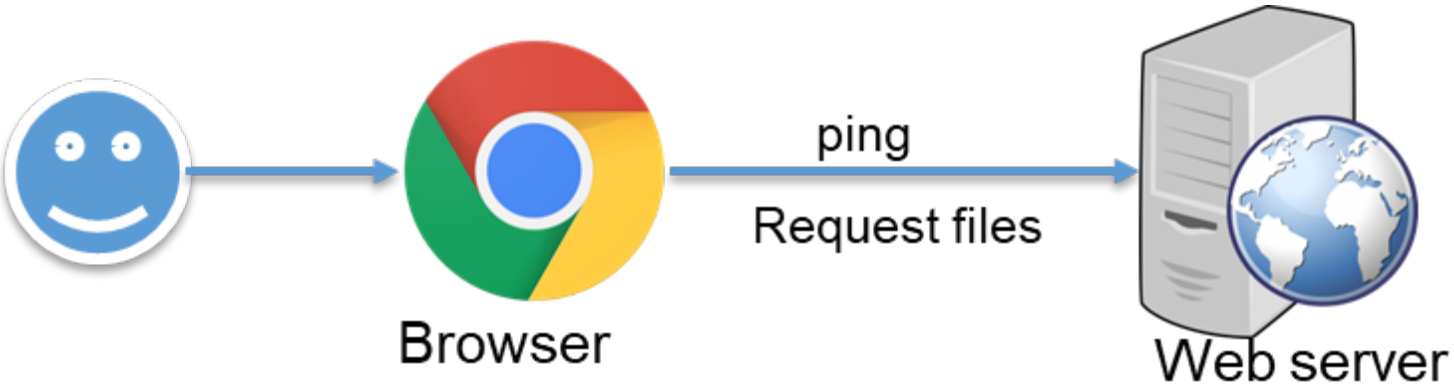
There are a more techniques available, but these are some of the common ones:

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Reduce Latency with a CDN
5. DNS Prefetch and Preconnect

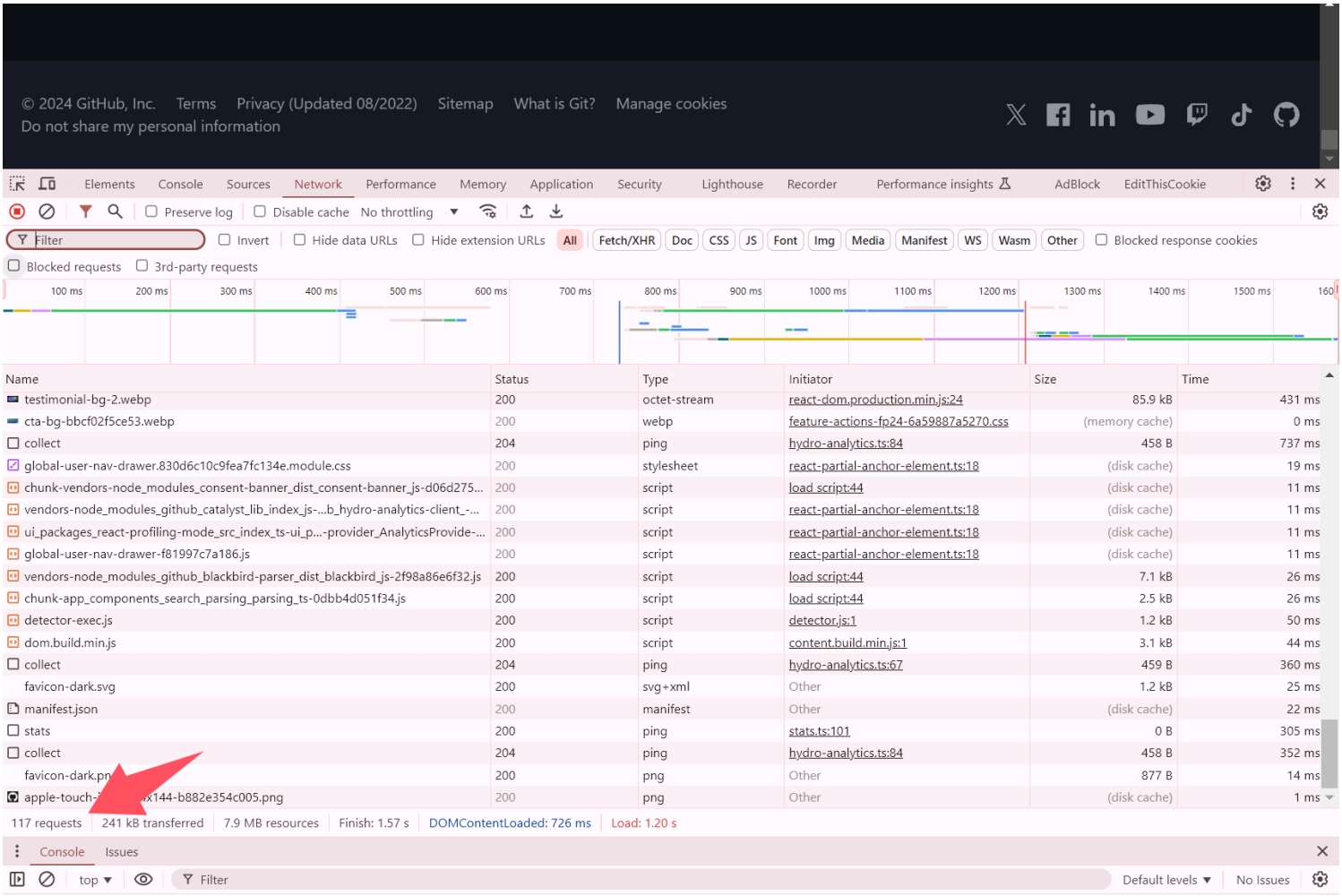
==========

1.Reduce HTTP Requests

Each time someone visits a webpage, here's what typically happens: that person's web browser (Chrome, Firefox, etc.) pings the web server that hosts the webpage they're trying to visit -- in this case, a webpage on your website. It requests that the server send the files containing the content for that site. These files contain any text, images, and multimedia that exist on that webpage, which is presented as picture below



If you look at Network tab from your browser, and at bottom left, you will know how many requests it is sending:



To optimise these area, you can think of reducing the number request, you can optimize HTTP request by:

- Combing images (A single image is more efficient than multiple small images)
- Reduce the size of files (File minimalization)
- Making JavaScript asynchronous (Some of the page can be delayed to send and load later)

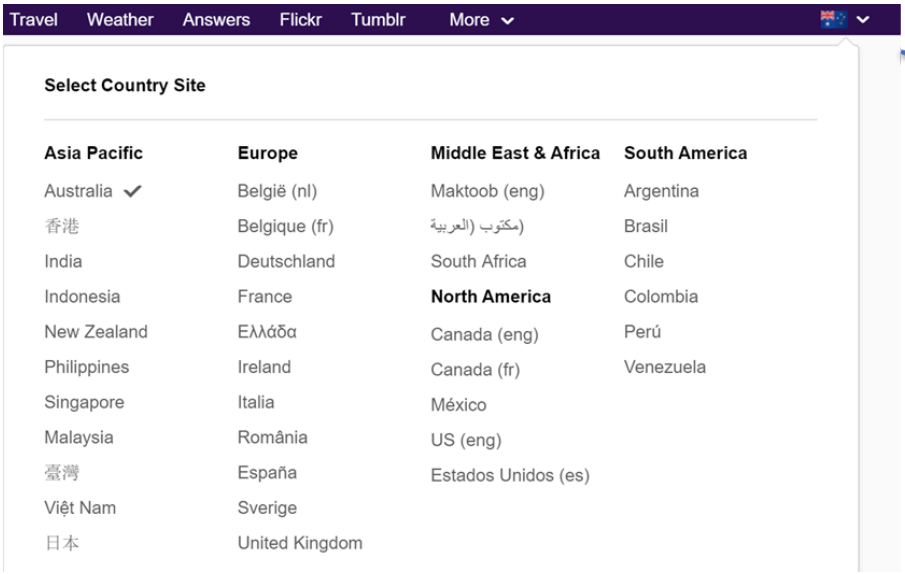
2.Image Optimization

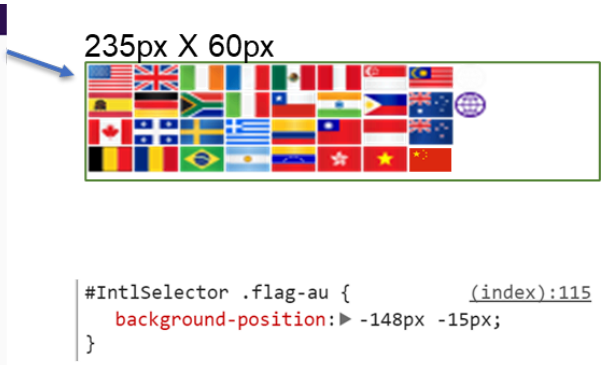
2.1 Replace image with css or reduce number of images

Make use of CSS to replace icons instead of small sized images (for example, you can use css to draw circles)

If small icons is unavoidable, merge the icons in one picture, access each picture by limiting the display boundary of the picture through CSS, this is due to retrieving each picture needs to initialize a HTTP request each time which is not efficient

Picture obtained Yahoo representing each country





2.2 Image compression - image file types

There are various of image types, most common images compression types are:

PNG – **Lossless compression** (The original data is perfectly reconstructed from the compressed data)

Suitable for icons, pictures with no background

JPEG – **Lossy compression** (Ideals for balance between the storages size and image quality)

Right picture shows the image after multiple compression algorithm applied ↑

2.3 Image compression – Responsive Images

Pre-process your image to save multiple resolution of it, and only retrieve the necessary ones according to your screen resolution.

This is how you load the image normally:

```
<img src=“monash.jpg" alt="responsive images car" width=“800" height=“600">
```

However, if you use *srcset* attribute, (Browser will decide which image it choose, w means the width

of the viewport - aka your browser window width)

```
<img src="monash.jpg" alt="responsive images car" srcset="monash-160.jpg 160w, monash-320.jpg 320w,
```

3.Minify CSS and JavaScript

Majority of your time, the css and JavaScript contains a lot of white spaces to make your coding human readable, however, this is wasting a lot of storage space.

One of the way to reduce the issues is by removing unecesary characters from your HTML

This can include:

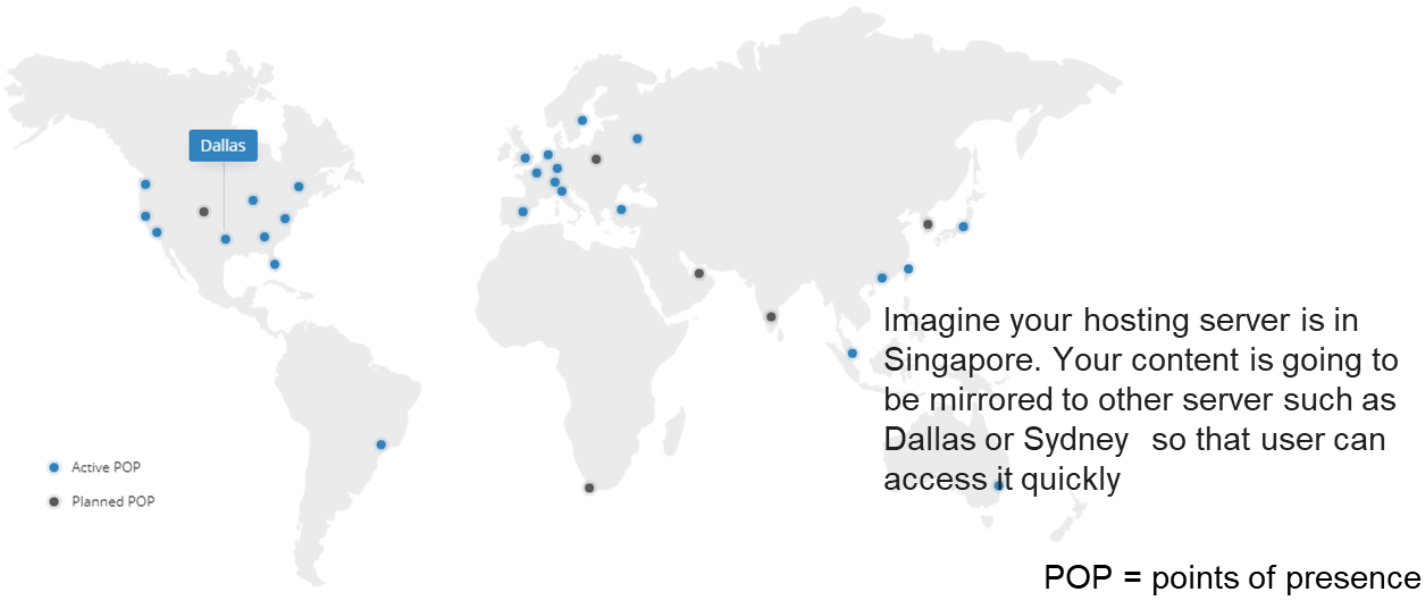
- White space characters
- New line characters
- Comments
- Block delimiters

For example, if you use https://javascript-minifier.com/ then you can make your JavaScript one line code like below:

4.Reduce Latency with a CDN

CDN such as cloud flare, is a **content delivery network** (**CDN**) that distributed servers (**network**) that deliver pages and other Web **content** to a user, based on the geographic locations of the user, the origin of the webpage's content and the **content delivery** server.

In this way, the latency of your server is reduced a lot



5. DNS Prefetch and Preconnect

5.1 DNS prefetching is an attempt to resolve domain names before a user tries to follow a link

> When we talk about “resolving” a domain name, we’re referring to the process of translating a human-readable domain name (like `example.com`) into an IP address (like `192.0.2.1`) that computers use to identify each other on the network. This process is handled by the Domain Name System (DNS).

Example:

```
<link rel="dns-prefetch" href="//www.example.com">
```

5.2 Preconnect is a way for browser to get a head start on connecting to a website, there are usually some tasks the browser needs to do such as, DNS Look up (Check 5.1), the TCP handshake (Establish a connection between your laptop and server) and TLS Negotiation (Transport layer security) to ensure secured connection is established.

Preconnect will do all these steps in advance so you don't need to wait it to happen.

Example:

<link href='https://example.com' rel='preconnect' crossorigin>