Web Development 1
John Rellis

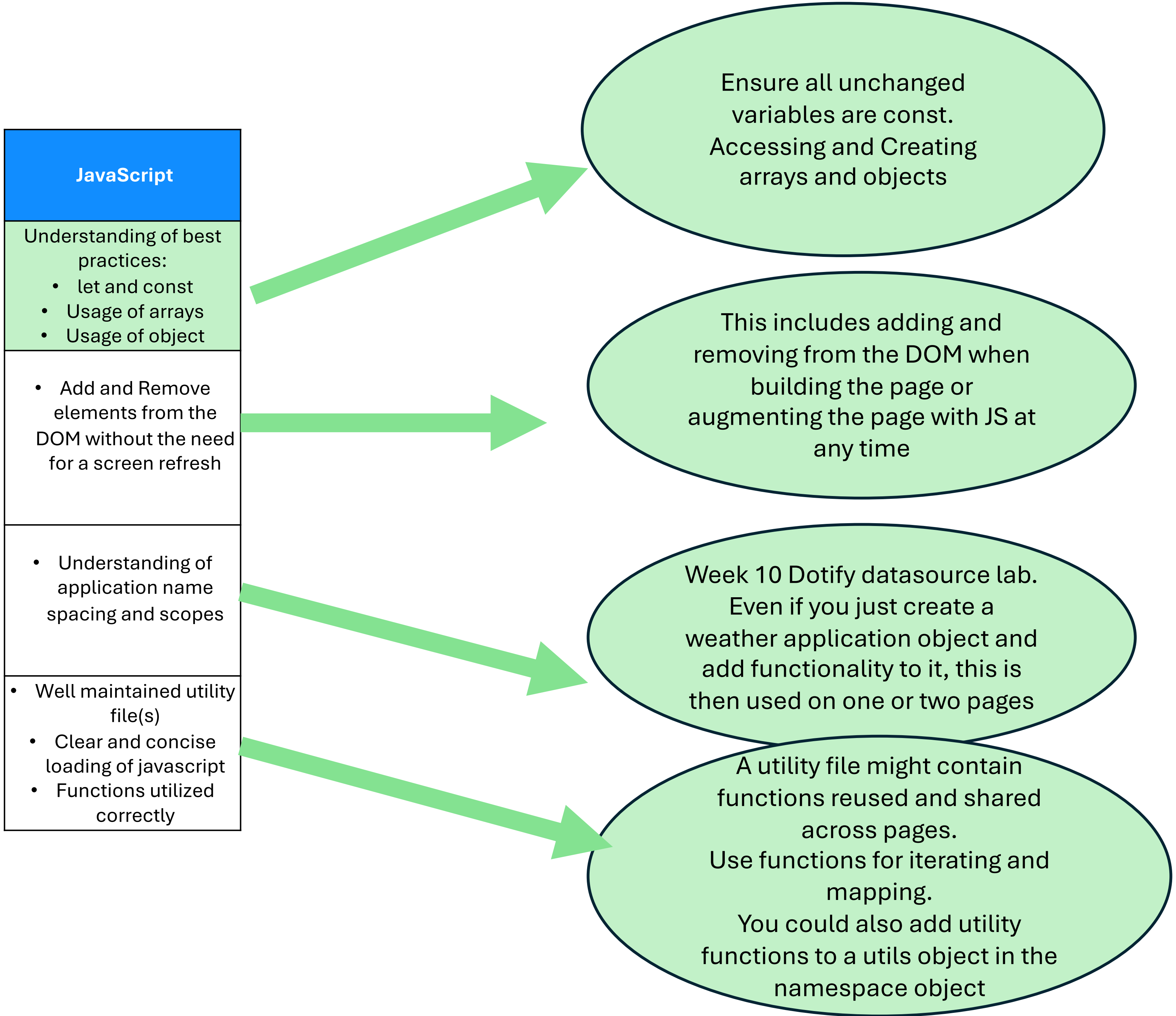
Assignment Clarifications

Updates

Grade Band	JavaScript	Dev Experience	Submission and Deployment	Release	Release Features
starter	Understanding of best practices: • let and const • Usage of arrays • Usage of object	Clearly laid out project structure – HTML, CSS and JavaScript	Zip file to moodle	POC – 1 - City Focus 1 1 City with today's weather	• Weather Code Mapped to Correct Weather • Display City Name • Clear it is today's weather • Max Temp and Wind
baseline	• Add and Remove elements from the DOM without the need for a screen refresh	DRY - Very little Repetition of HTML, JS and Styles	+ github repository link + github commit history OR bulma breadcrumb component	POC – 2 – City Focus 2 + current hour's weather	• Added current hours forecast • Correctly mapped hourly weather code • Correct hour taken from browser
good	• Understanding of application name spacing and scopes	Layered JavaScript Architecture - MVC	+ github tags OR bulma dropdown component to select something (city?)	POC – 3 – City Focus 3 + 7 days summary	• Clear and concise summary for the 7 days • Weather code correctly mapped • Additional weather information
excellent	• Well maintained utility file(s) • Clear and concise loading of javascript • Functions utilized correctly	Excellent README	+ Manual upload to netlify	Release -1 + Dashboard	• Focus on user experience • Ability to navigate to city view • Ability to navigate back to dashboard • Usage of URL paths and parameters
outstanding			+ github push deploys to netlify OR use the bulma modal to show weather info in a popup	Release – 2 + Configure and Persist User Preferences	• User preferences user interface • User preferences saved in localStorage • Clear navigation • User preferences affect how the dashboard render • Reset preferences
amazing			-	Release – 3 + Build out your own features	• Start small and build from there • Do not start this without a lot of the previous releases done, don't feel pressure to get here

Passing Pathway

Grade Band	JavaScript	Dev Experience	Submission and Deployment	Release	Release Features
starter	Understanding of best practices: • let and const • Usage of arrays • Usage of object	Clearly laid out project structure – HTML, CSS and JavaScript	Zip file to moodle	POC – 1 - City Focus 1 1 City with today’s weather	• Weather Code Mapped to Correct Weather • Display City Name • Clear it is today’s weather • Max Temp and Wind
baseline	• Add and Remove elements from the DOM without the need for a screen refresh	DRY - Very little Repetition of HTML, JS and Styles	+ github repository link + github commit history OR bulma breadcrumb component	POC – 2 – City Focus 2 + current hour’s weather	• Added current hours forecast • Correctly mapped hourly weather code • Correct hour taken from browser
good	• Understanding of application name spacing and scopes	Layered JavaScript Architecture - MVC	+ github tags OR bulma dropdown component to select something (city?)	POC – 3 – City Focus 3 + 7 days summary	• Clear and concise summary for the 7 days • Weather code correctly mapped • Additional weather information
excellent	• Well maintained utility file(s) • Clear and concise loading of javascript • Functions utilized correctly	Excellent README	+ Manual upload to netlify	Release -1 + Dashboard	• Focus on user experience • Ability to navigate to city view • Ability to navigate back to dashboard • Usage of URL paths and parameters
outstanding			+ github push deploys to netlify OR use the bulma modal to show weather info in a popup	Release – 2 + Configure and Persist User Preferences	• User preferences user interface • User preferences saved in localStorage • Clear navigation • User preferences affect how the dashboard render • Reset preferences
amazing			-	Release – 3 + Build out your own features	• Start small and build from there • Do not start this without a lot of the previous releases done, don’t feel pressure to get here



JavaScript

Understanding of best practices:

- let and const
- Usage of arrays
- Usage of object

Ensure all unchanged variables are const.
Accessing and Creating arrays and objects

- Add and Remove elements from the DOM without the need for a screen refresh

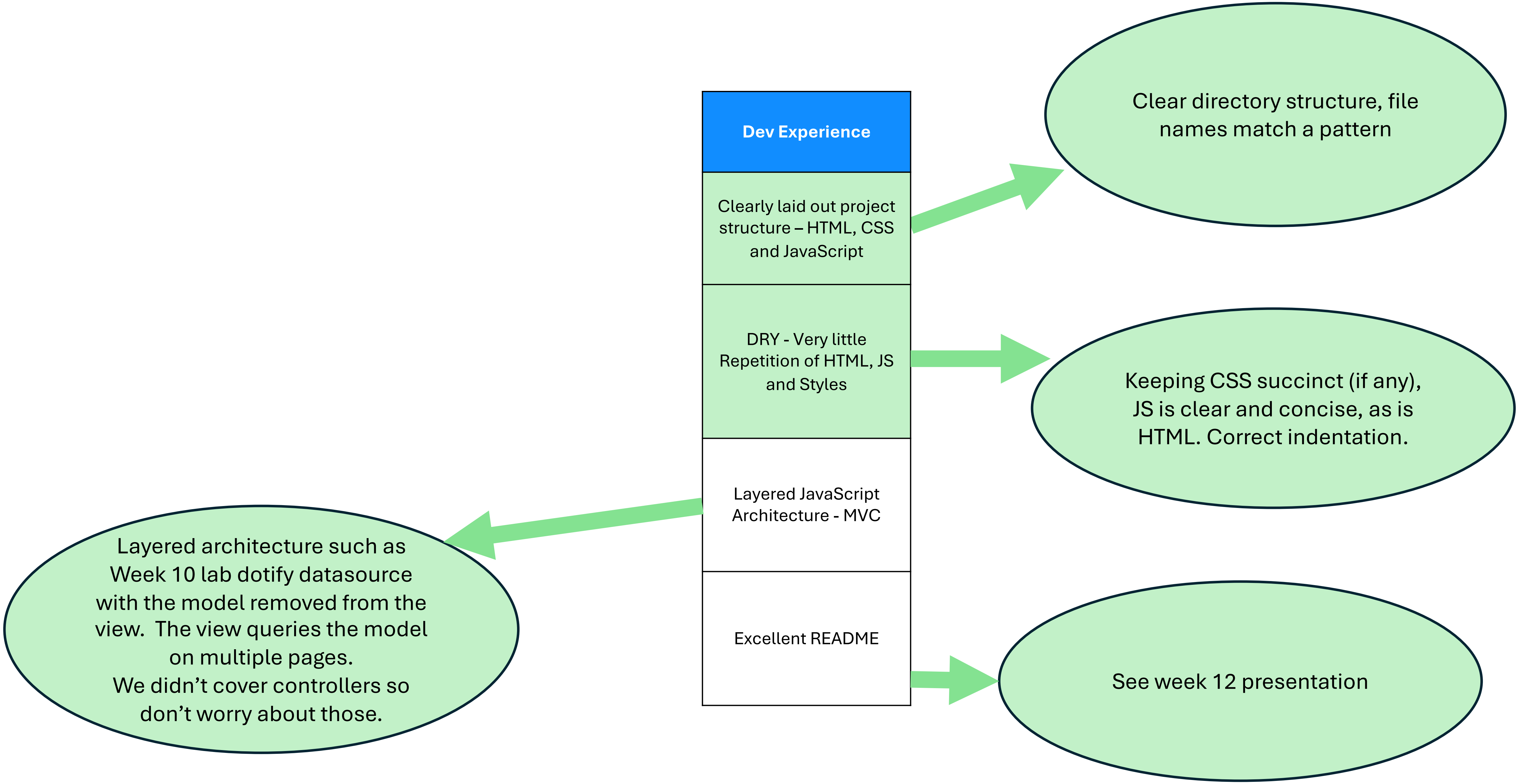
This includes adding and removing from the DOM when building the page or augmenting the page with JS at any time

- Understanding of application name spacing and scopes

Week 10 Dotify datasource lab. Even if you just create a weather application object and add functionality to it, this is then used on one or two pages

- Well maintained utility file(s)
- Clear and concise loading of javascript
- Functions utilized correctly

A utility file might contain functions reused and shared across pages.
Use functions for iterating and mapping.
You could also add utility functions to a utils object in the namespace object



Dev Experience

Clearly laid out project structure – HTML, CSS and JavaScript

Clear directory structure, file names match a pattern

DRY - Very little Repetition of HTML, JS and Styles

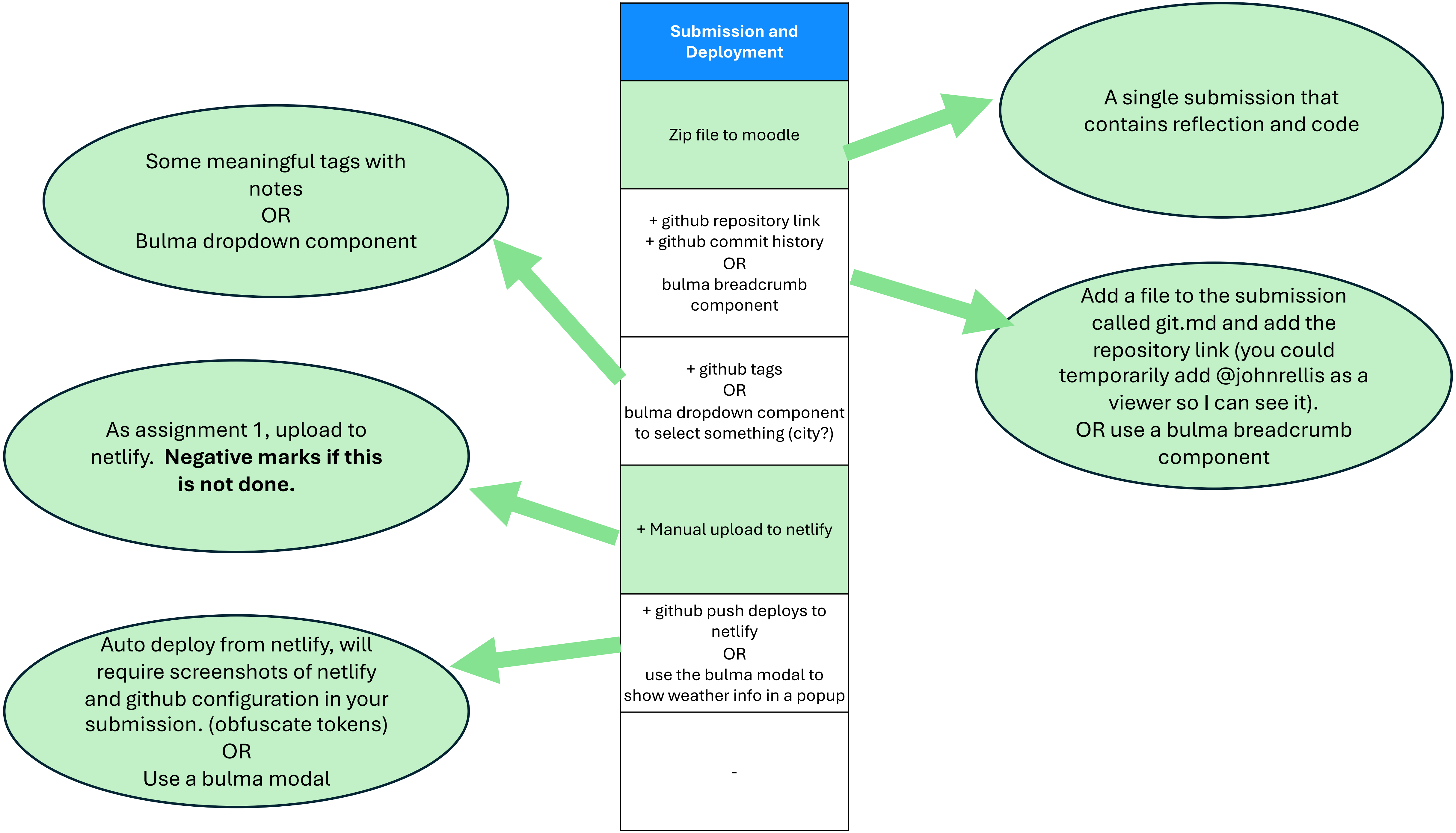
Keeping CSS succinct (if any), JS is clear and concise, as is HTML. Correct indentation.

Layered JavaScript Architecture - MVC

Layered architecture such as Week 10 lab dotify datasource with the model removed from the view. The view queries the model on multiple pages.
We didn't cover controllers so don't worry about those.

Excellent README

See week 12 presentation



Submission and Deployment
Zip file to moodle
+ github repository link + github commit history OR bulma breadcrumb component
+ github tags OR bulma dropdown component to select something (city?)
+ Manual upload to netlify
+ github push deploys to netlify OR use the bulma modal to show weather info in a popup
-

A single submission that contains reflection and code

Add a file to the submission called git.md and add the repository link (you could temporarily add @johnrellis as a viewer so I can see it).
OR use a bulma breadcrumb component

Some meaningful tags with notes
OR
Bulma dropdown component

As assignment 1, upload to netlify. **Negative marks if this is not done.**

Auto deploy from netlify, will require screenshots of netlify and github configuration in your submission. (obfuscate tokens)
OR
Use a bulma modal

The following pages give suggestions on working with data for each release

These do not represent the only way to access the data of course

There are more advanced ways, such as datastore

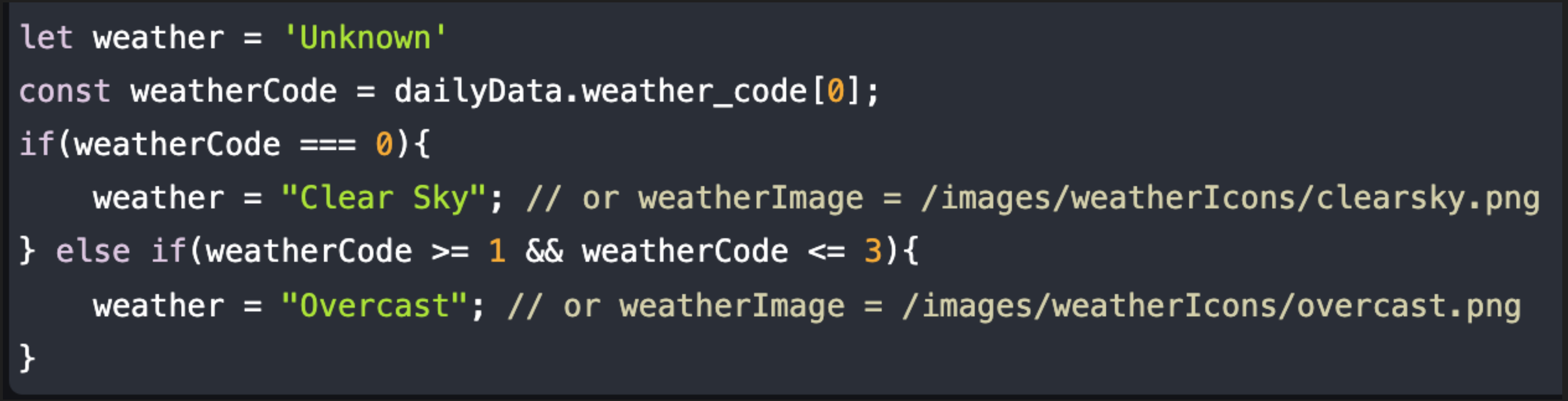
There are also ways to get the index of the the day and hour and work from there

BUT the following will show you how to access data with several other tips

These do no indicate the only way, just “a” way, if you are not doing these, do not worry

Release	Release Features
POC – 1 - City Focus 1 1 City with today’s weather	• Weather Code Mapped to Correct Weather • Display City Name • Clear it is today’s weather • Max Temp and Wind

- Accessing data
  - `const dailyData = weatherData["berlin_daily"].daily;`
    `const weatherCodeToday = dailyData.weather_code[0];`
  - `const maxTempToday = dailyData.temperature_2m_max[0]`
  - `const maxTempTomorrow = dailyData.temperature_2m_max[1]`
  - https://tutors.dev/lab/wit-hdip-comp-sci-2024-web-dev-1/topic-11-dates-query-params-tables/unit-0/book-b-dates/02
- Weather Code Mapping
  - This could potentially be a handy util function – to map the weather code to the words/path to image
  - https://tutors.dev/lab/wit-hdip-comp-sci-2024-web-dev-1/topic-11-dates-query-params-tables/unit-0/book-b-dates/02



```
let weather = 'Unknown'
const weatherCode = dailyData.weather_code[0];
if(weatherCode === 0){
    weather = "Clear Sky"; // or weatherImage = /images/weatherIcons/clearsky.png
} else if(weatherCode >= 1 && weatherCode <= 3){
    weather = "Overcast"; // or weatherImage = /images/weatherIcons/overcast.png
}
```

Release	Release Features
POC – 2 – City Focus 2 + current hour’s weather	• Added current hours forecast • Correctly mapped hourly weather code • Correct hour taken from browser

- Accessing data

```
const hourlyData = weatherData["berlin_hourly"].hourly;
const weatherCodeMidnight = hourlyData.weather_code[0];
const weatherCode1am = hourlyData.weather_code[1];
const tempAt1am= hourlyData.temperature_2m[1]
```

- Current Hour from Browser - dayjs().hour();
  - https://tutors.dev/lab/wit-hdip-comp-sci-2024-web-dev-1/topic-11-dates-query-params-tables/unit-0/book-b-dates/02

Release	Release Features
POC – 3 – City Focus 3 + 7 days summary	• Clear and concise summary for the 7 days • Weather code correctly mapped • Additional weather information

- Accessing data
  - `const dailyData = weatherData["berlin_daily"].daily;`
    `const weatherCodeToday = dailyData.weather_code[0];`
  - `const weatherCodeTomorrow = dailyData.weather_code[1];`
  - `const maxTempToday = dailyData.temperature_2m_max[0]`
  - `const maxTempTomorrow = dailyData.temperature_2m_max[1]`
- To find all elements that have ids that have the same prefix (remember ids have to be unique per HTML document)

```
document.querySelectorAll("[id^=forecast-]").forEach(element => {
// will find all elements that ids that start with forecast- such as forecast-Today,
forecast-Today+1, forecast-Today+2 etc
});
```

- See week 12 localstorage lab for usage in dotify

Release	Release Features
Release -1 + Dashboard	• Focus on user experience • Ability to navigate to city view • Ability to navigate back to dashboard • Usage of URL paths and parameters

- You could hardcode what cities you want to display, either in an array or otherwise
- You could also use Object.keys(weatherData) to get all the city names
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/keys
- You must link to a single cityFocus page with the query param that has the city name
  - /cityFocus?city=berlin
- You might now switch your index to be the dashboard and your cityFocus page should be separate
- You should not have multiple cityFocus pages, e.g. cityFocusParis, cityFocusBerlin
- You should have one city focus page with a query parameter of city or name e.g. /cityFocus?city=berlin, /cityFocus?city=paris
  - https://tutors.dev/lab/wit-hdip-comp-sci-2024-web-dev-1/topic-11-dates-query-params-tables/unit-0/book-a-dotify-tables-templating/04

Release – 2 + Configure and Persist User Preferences	• User preferences user interface • User preferences saved in localStorage • Clear navigation • User preferences affect how the dashboard render • Reset preferences
Release – 3 + Build out your own features	• Start small and build from there • Do not start this without a lot of the previous releases done, don’t feel pressure to get here

- For localStorage, check out week 12 dotify lab with favourites
- Examples of settings or user preferences
    - You may want to only show favourites on your dashboard, or highlight them
    - Might want a separate favourites page
    - More examples in the original assignment spec

Marks for Technology & Data

- There will be a section in the marking scheme for technogies used and the proficiency with which you used them
- It is a requirement that you use JavaScript, Bulma, HTML
- CSS (if you want to deviate from bulma, there's no need to though but up to you)
- There will also be marks for usage of
  - dayjs
  - Components as per dotify labs
  - Templating and partials (nunjucks)
  - Eleventy
  - Eleventy layouts
  - Different design patterns/techniques
  - More at my discretion, will be pointed out in your grade
- You don't need to aim for all of the above as you can get full marks without using all of the above.
- I included JQuery in the spec to give people with experience a "quality of life" improvement, if you haven't used it, do not worry at all.

Cutoff

- Get your questions in now as a cutoff has already been communicated to you