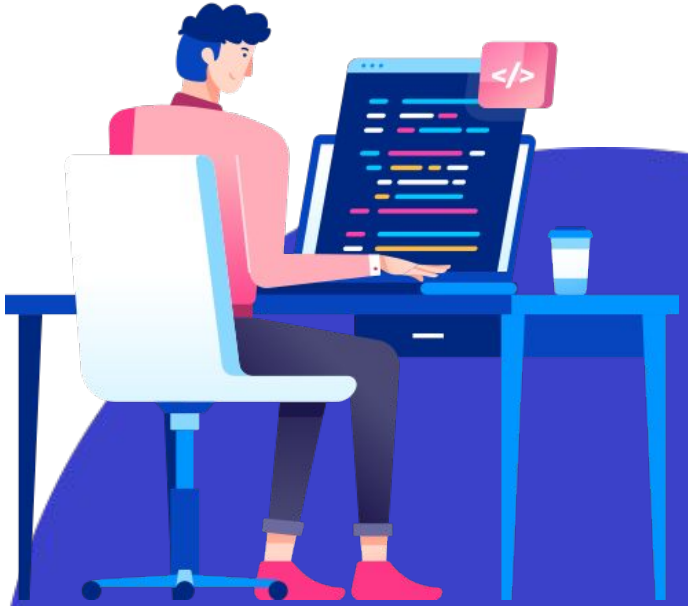
Programming Fundamentals

OBJECTS & LIBS

DIPLOMA IN FULL-STACK DEVELOPMENT

Certificate in **Computing Fundamentals**



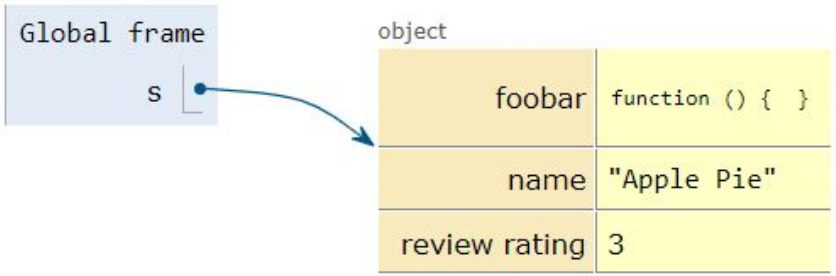
More on Objects

Objects are References

JavaScript ES6
(known limitations)

```
1 let s = {
2   foobar: function() {  },
3   name: "Apple Pie",
4  "review rating": 3
5 }
```

Edit this code



This means that:

- They are passed by reference
- They are **not copied** when assigned to another variable

Traversing Objects (1)

Using ***for (let <key> in <object>)*** allows us to retrieve each key from an object:

```
let book = {
 "title": "The Lord of the Rings",
 "author": "JRR Tolkien",
 "ISBN":"123-123-123-123",
 "pages": 7220
}

for (let k in book) {
 console.log(k);
}
```

title
author
ISBN
pages

Traversing Objects (2)

We can therefore use the **extracted key with the [ ] operator to get each value**:

```
let book = {
 "title": "The Lord of the Rings",
 "author": "JRR Tolkien",
 "ISBN":"123-123-123-123",
 "pages": 7220
}

for (let k in book) {
 console.log(book[k]);
}
```

```
The Lord of the Rings
JRR Tolkien
123-123-123-123
7220
```

Nested Objects

The value of a property can also be an object:

```
let employee = {
 firstName:'Tan',
 lastName:'Ah Kow',
 address: {
   street:'Yishun Ring Road',
   building:'Blk 636',
   unit:'#13-221'
 }
}

console.log(employee.address.street);
```

Yishun Ring Road

Object Methods

Object.values()

Return the values of all properties as an array.

Usage:

```
let book = {
 "title": "The Lord of the Rings",
 "author": "JRR Tolkien",
 "ISBN":"123-123-123-123",
 "pages": 7220
}

console.log("Object.values(book) =>", Object.values(book));
```

```
[ 'The Lord of the Rings', 'JRR
Tolkien', '123-123-123-123',
7220 ];
```

Object Methods

There are a number of methods that you can use on an object. We'll go through the most common of those methods.

Object.entries()

Returns the key/value of one property at a time. Use in a *for* loop:

```
Usage:
for (let [key, value] of Object.entries(book)) {
 console.log(key,"=>",value)
}

Output:
title => The Lord of the Rings
author => JRR Tolkien
ISBN => 123-123-123-123
pages => 7220
```

Object.hasOwnProperty()

Check if a property name exists in the Object:

```
console.log("book.hasOwnProperty() =>",book.hasOwnProperty('pages'));
// Or doing almost the same thing:
console.log('"pages in book =>"', "pages" in book);
```

Object.assign()

"Merges" the properties of two objects together into a new object. The second priorities from the second object has higher priority.

```
let book = {
 "title": "The Lord of the Rings",
 "author": "JRR Tolkien",
 "ISBN":"123-123-123-123",
 "pages": 7220
}
let changes = {
 'pages': 900
}
let book2 = Object.assign(book, changes);
console.log(book2);
```

```
{
  title: 'The Lord of the Rings',
  author: 'JRR Tolkien',
  ISBN: '123-123-123-123',
  pages: 900
}
```

Destructuring Objects

Destructuring Objects

You can copy the values of *properties* in an object into new variables using destructuring.

```
let fruits = {
 "apples": 20,
 "bananas": 35,
 "pears": 45,
 "oranges": 66
}

let {bananas, pears} = fruits;
console.log(bananas, pears);
// will get 35, 45
```

The variable names *must* match the property names that we will to copy to the variable

Destructuring & Rename

You can rename the *copied variables* if you like:

```
let fruits = {
 "apples": 20,
 "bananas": 35,
 "pears": 45,
 "oranges": 66
}

let {bananas:b, pears:p} = fruits;
console.log(b, p);
// will still get 35, 45
```

Destructuring an object as Params

You can destruct an object as parameters in a function too!

```
let product = {
 'price': 35,
 'sku': 'APPLE01',
 'description':'Red apples',
 'salesTax': 0.07
}

function calculateTotalPrice({price, salesTax}) {
 return price * salesTax;  }

console.log(calculateTotalPrice(product));
```

The 5 HTTP methods

Recap for API

The “Verbs”

There are five kind of *actions* we can perform when making a HTTP request. Those are known as *verbs* or *methods*

- GET → retrieve data
- PUT or PATCH → update data
- POST → Create new data on the server
- DELETE → Delete new data from the server

HTTP Get & Params

Endpoint is Dynamic

Unlike a *json* file, the output of a RESTFul API endpoint is generated by a program.

So there's a **program running** behind each point. When we *consume* an endpoint, it's like calling a function over the Internet.

Parameters

Those are the "arguments" we pass to the endpoint.

The endpoint will act differently base on the arguments

Endpoint & Query String

The ***query string*** are the parameters passed to the end point.

It's **not** part of the end point URL itself.

endpoint **Query string**

https://iamasuperhero-api.herokuapp.com/reports?category=Powers%20On%20Empty
GET Method
Category = Powers On Empty
%20 is our spacebar

Understanding Query Strings

The ? is the start of the query string.

Each parameter is separated by a &

Each pair of = is one parameter

?category=food&search_terms=chicken

The *category* parameter has the value of *food*

The *search_terms* parameter has the value of *chicken*

Understanding Query Strings

https://www.food.fake/search?category=food&search_terms=chicken

Can be sent using axios as:

```
axios.get('https://www.food.fake/food, {

  params: {

   category:'food',
   search_terms:'chicken'

  }

}
```

HTTP Post & Body

POST method

The HTTP **POST** method is used for **creating new resources on the server.**

For example, say we can create a new food by sending a request using the **POST** method to *http://ww.food.fake/food* and specify the **body** to have *category* and title.

Can be sent using axios as:

```
axios.post('https://www.food.fake/food, {

   category:'food',
   title:'Chicken Rice'

  }

}
```

HTTP Patch/Put

PUT method

Those two methods are used to update existing data.

- Patch → update a resource by changing certain parts of the existing data
- Put → update a resource by totally replacing it with new data

PUT method

For example, say we can create a update the chicken rice by sending a request using the **PATCH** method to *http://ww.food.fake/food* and specify the **body** to have *category* and title.

Can be sent using axios as:

ID of the 'Chicken Rice' dish that we want to change

```
axios.patch('https://www.food.fake/food/412, {

   category:'chinese-food',
   title:'Hainanese Chicken Rice'

  }

)
```

HTTP Delete

DELETE method

As its name implies, the DELETE method is for removing resources from the server.

For example, say we can create a delete the chicken rice with the id of **412** by sending a request using the **DELETE** method to *http://ww.food.fake/food*

Can be sent using axios as:

```
axios.delete('https://www.food.fake/food/412')
```

AXIOS

Delightful Library

Reading in JSON files

We can use the *fetch* api to retrieve JSON files

```
fetch('data.json').then((r)=>{
    return r.json()
}).then(function(json){
    console.log(json);
})
```

data.json must be in the same directory as the JS file.

fruits.json

```
{
    "fruits":[
        "apples",
        "oranges",
        "bananas",
        "blueberries"
    ]
}
```

https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

Using Axios

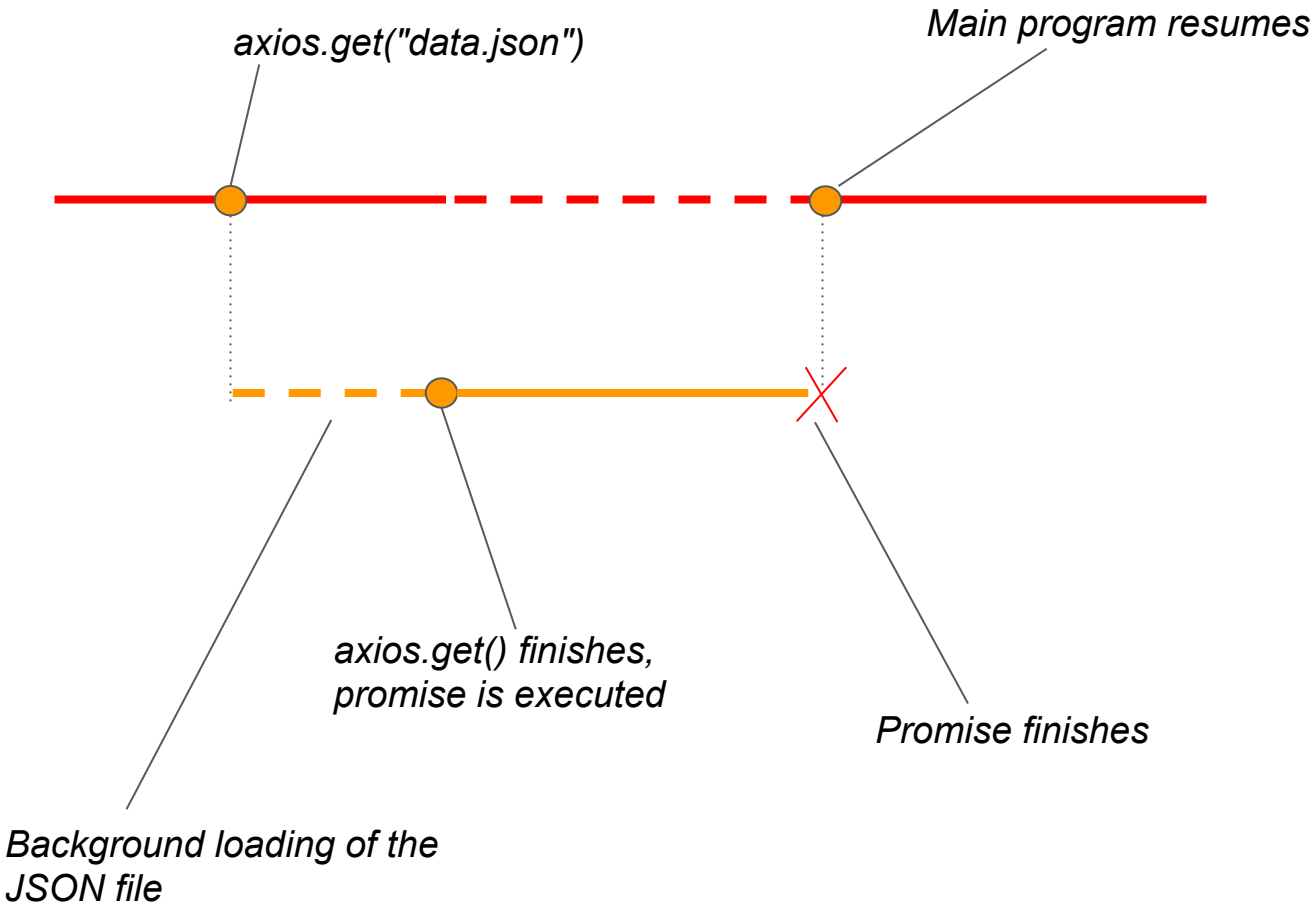
Axios is a library to manage *fetch* calls.

axios.get is async, will return a promise. A *promise* is a function that happens in the background

```
axios.get('data.json').then((r)=>{
    return r.json()
}).then(function(json){
    console.log(json);
})
```

This function is a *called* when the promise *resolves*

https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios



https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

How does AXIOS work

Axios reads file using the HTTP protocol.

This is the same method the browser fetches HTML, CSS, image and such files from a web server.

The files must be available on a web server.

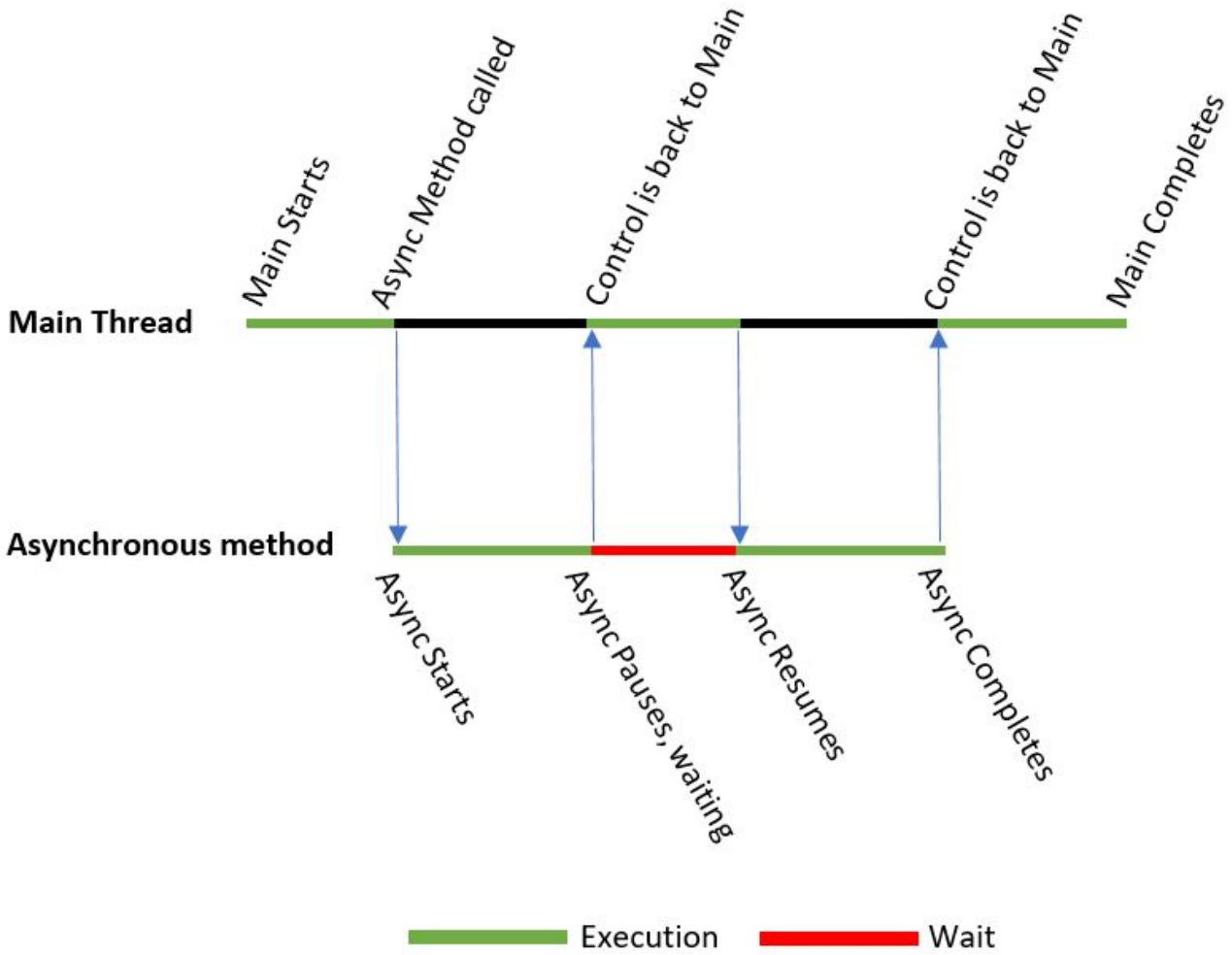
https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

ASYNC/AWAIT

This is the most modern method of executing an asynchronous function.

```
async function getData() {
    let response = await axios.get('data.json');
    console.log(response.data);

}

getData();  // execute the async function call
```

https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios



https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

ASYNC/AWAIT

Await and async can only be used in functions marked with <u>async</u>

```
async function getData() {
    let response = await axios.get('data.json');
    console.log(response.data);

}

getData();  // execute the async function call
```

https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

Parallel Async/Await

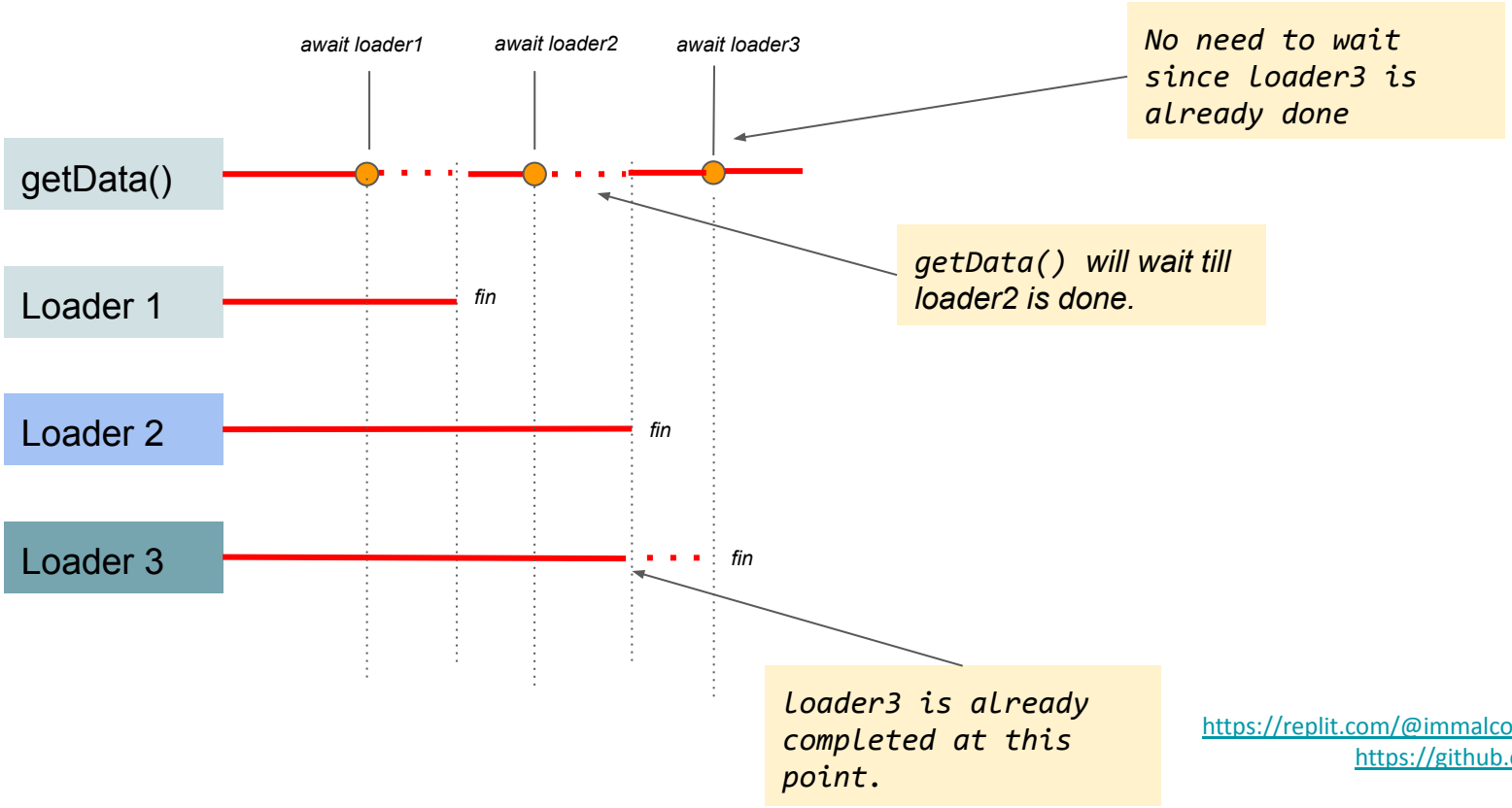
```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>repl.it</title>
   <link href="style.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
   <button id="load">Load</button>
   <textarea id="output" rows="5"></textarea>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.20.0/axios.min.js">
</script>
   <script src="script.js"></script>
 </body>
</html>
```

```js
document.querySelector("#load").addEventListener('click',
async function(){
 let loader1 = axios.get('data.json');
 let loader2 = axios.get('data1.json');
 let loader3 = axios.get('data2.json');
 let response = await loader1;
 document.querySelector('#output').innerHTML += response.data.message + "\n";

 response = await loader2;
 document.querySelector('#output').innerHTML += response.data.message + "\n";

 response = await loader3;
 document.querySelector("#output").innerHTML += response.data.message +"\n";


})
```

Launch three *axios* requests without waiting for the previous one to end

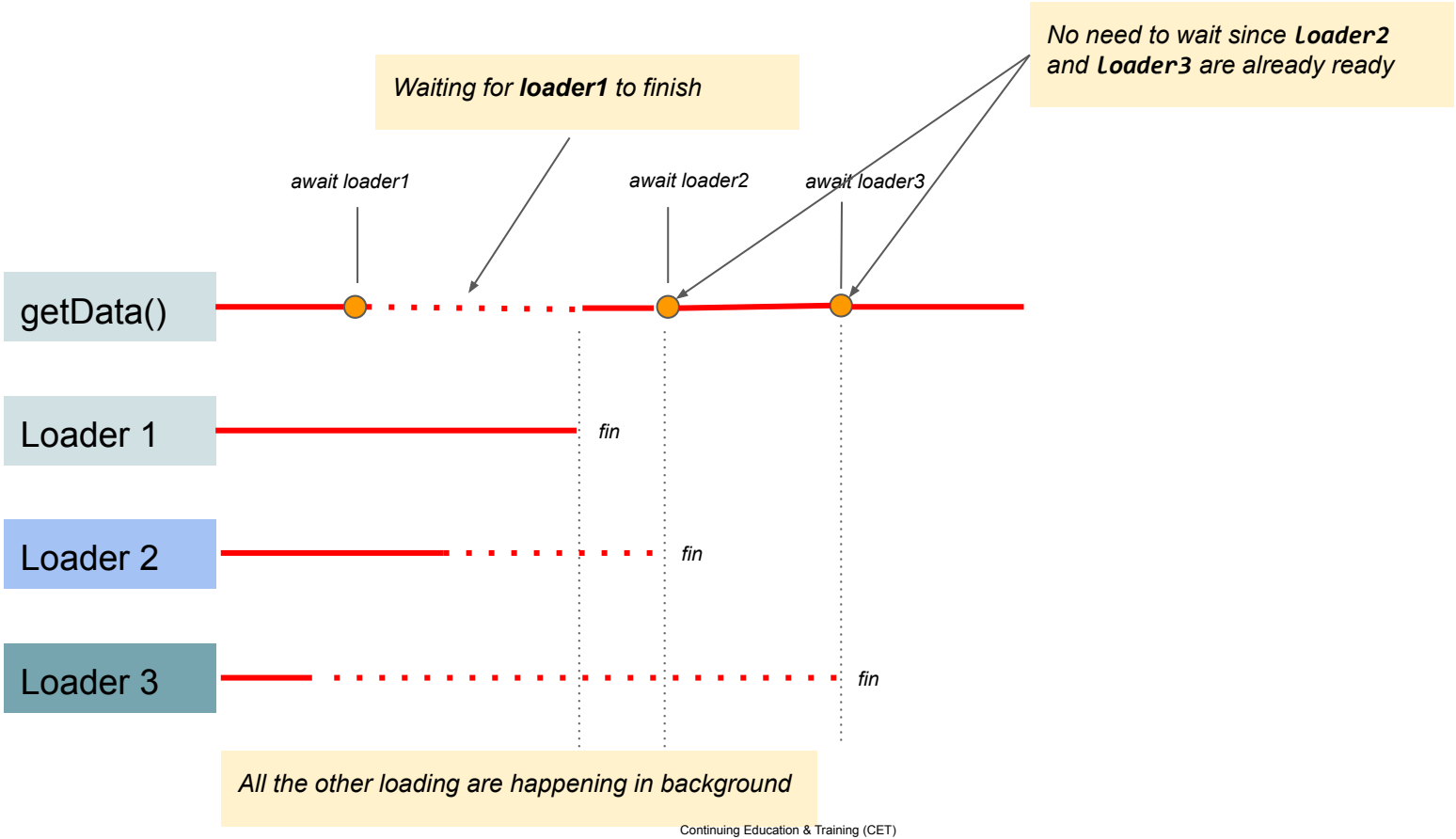
https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

Case 1: Complete in Order



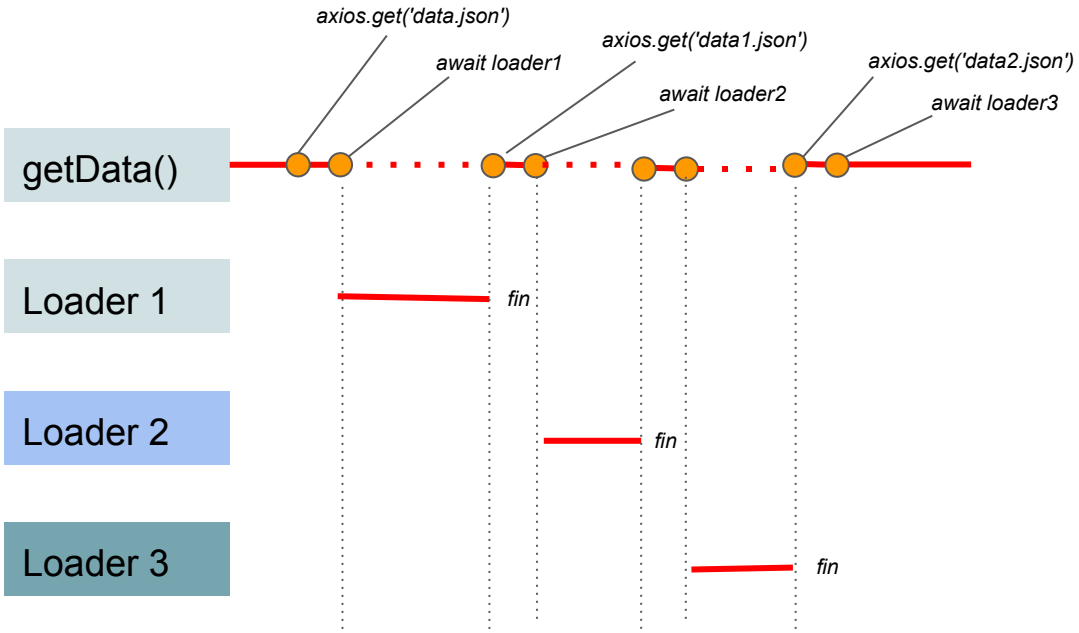
https://replit.com/@immalcolm/simple-axios
https://github.com/axios/axios

Case 2: Reverse Order

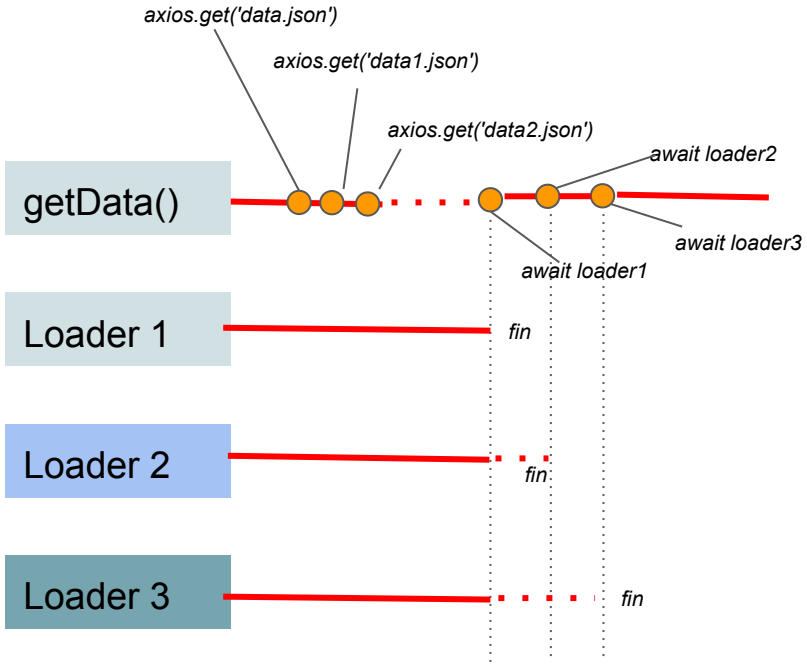


Case 3: What if it's synchronous?

```
document.querySelector("#load").addEventListener('click',
async function(){
 let loader1 = axios.get('data.json');

 let response = await loader1;
 document.querySelector('#output').innerHTML +=
response.data.message + "\n";

 let loader2 = axios.get('data1.json');
 response = await loader2;
 document.querySelector('#output').innerHTML +=
response.data.message + "\n";

 let loader3 = axios.get('data2.json');
 response = await loader3;
 document.querySelector("#output").innerHTML +=
response.data.message +"\n";
})
```



Case 4: Best Case Scenario (async)



JSON FILE

```
[
  {
    "type": "digital",
    "display_name": "Digital"
  },
  {
    "type": "physical",
    "display_name": "Physical"
  },
  {
    "type": "service",
    "display_name": "Online service"
  }
]
```

product-types.json

```
[
    {
        "type": "cash",
        "display_name":"Cash"
    },
    {
        "type":"credit_card",
        "display_name":"Credit Card"
    },
    {
        "type": "paypal",
        "display_name": "PayPal"
    }
]
```

payment-types.json

Data-Driven Form

Load all the product types from the JSON file

```
    <div>
      <div>
        <label>Product Name</label>
        <input type="text" id="product-name" />
      </div>
      ...
    <script
src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"><
/script>
    <script src="list-driven-form.js"></script>
  </body>
```

```
async function loadProductTypes() {
  let response = await axios.get("product-types.json");
  let productTypes = response.data;
  let productTypeDiv = document.querySelector("#product_types");
  for (let p of productTypes) {
    let spanElement = document.createElement("span");
    spanElement.innerHTML = `<input type="radio"
                              name="product-type"
                              class="product-type"
                              value="${p.type}"/>
                              <span>${p.display_name}</span>`;
    productTypeDiv.appendChild(spanElement);
  }
}

loadProductTypes();
```

Make sure to remember to include the *axios* script.

See the full code here: https://replit.com/@immalcolm/axios-data-driven

TOOLS OF THE TRADE

WORK SMART

Designs & Visuals (Where to start)

Use these sites to help you in you visual research

Dribbble - Get inspired by designers
https://dribbble.com/shots/popular/web-design

Behance.net "Linkedin" for creatives
https://www.behance.net/search/projects?field=web%20design

Find inspirations by design patterns
https://pttrns.com/

Website inspiration
https://www.awwwards.com/
https://www.siteinspire.com/
Pinterest Web Design Trends
https://cssnectar.com/

Common Comments

- Cher.. My design skill CMI
- Cher I don't know what colors to use.
- Where do I start?
- How do I layout my site

Colors
https://color.adobe.com/create/color-wheel
Google Material Color Tool
https://material.io/resources/color/#!/

AI Powered Color Generator
http://khroma.co/ (fun, love it)

Instagram Inspiration
https://www.instagram.com/welovewebdesign/
https://www.instagram.com/dailywebdesign/
https://www.instagram.com/uibucket/

Reading
https://99designs.com.sg/blog/tips/10-of-the-best-sources-for-web-design-inspiration/
https://material.io/design/color/the-color-system.html#tools-for-picking-colors

Designs & Visuals (LOGO)

I have no photoshop skills….

It's great when we can see a logo original from your team (esp. IM students) but many a time,…. There's no logo even :(
A logo helps you to stand out. Be remembered.

Here's some quick tools for you to generate a logo

https://hatchful.shopify.com/
https://www.namecheap.com/logo-maker/
https://www.canva.com/create/logos/
https://www.fiverr.com/logo-maker
https://placeit.net/logo-maker

Tools

Once you are done with your logo, generate favicon for multiple platforms
https://www.favicon-generator.org/

Importance of favicon
https://www.hongkiat.com/blog/favicons-importance/

Designs & Visuals (Fonts)

Fonts/Typography helps us to make things readable and stand out more or contrast better

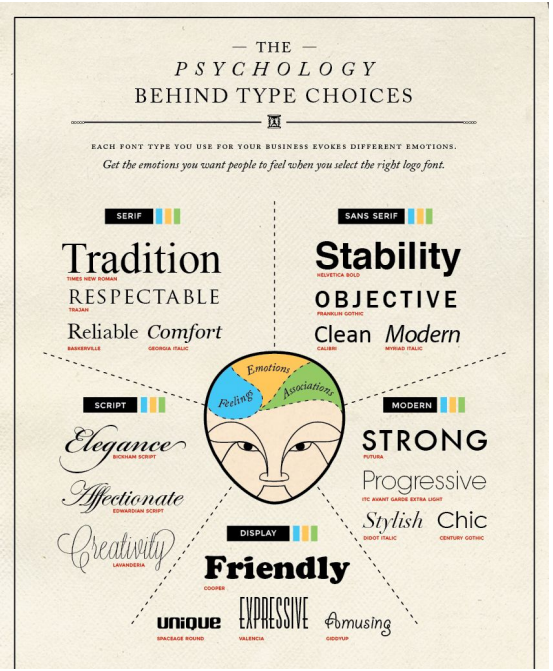
Use better fonts, or better yet use suitable font pairings

https://fonts.google.com/

Fun sites
https://fontpair.co/

Font Pairing Suggestions (Lost? Pick a pair then improvise later)
https://www.pagecloud.com/blog/best-google-fonts-pairings
https://digitalsynopsis.com/design/best-google-font-combinations-typeface-pairings/
https://www.canva.com/learn/best-professional-fonts-use-website/
https://www.awwwards.com/20-best-web-fonts-from-google-web-fonts-and-font-face.html

Students….
I have not been using other fonts..
I think Times New Roman nice lei.. Cher..
I used more than… 3 font faces in my site..



Reading
https://99designs.com.sg/blog/tips/the-best-and-worst-typefaces-and-why/
https://www.crazyegg.com/blog/psychology-of-fonts-infographic/

Displaying your Work Better

Adding in a snapshot or an animated gif of your final product into your README.md

https://www.screentogif.com/
Allows you to capture your screen into an animated gif

How to add screenshot to README.md?

Method 1: HTML

```
<img src="images/screenshot.png"
style="margin: 0;">
```

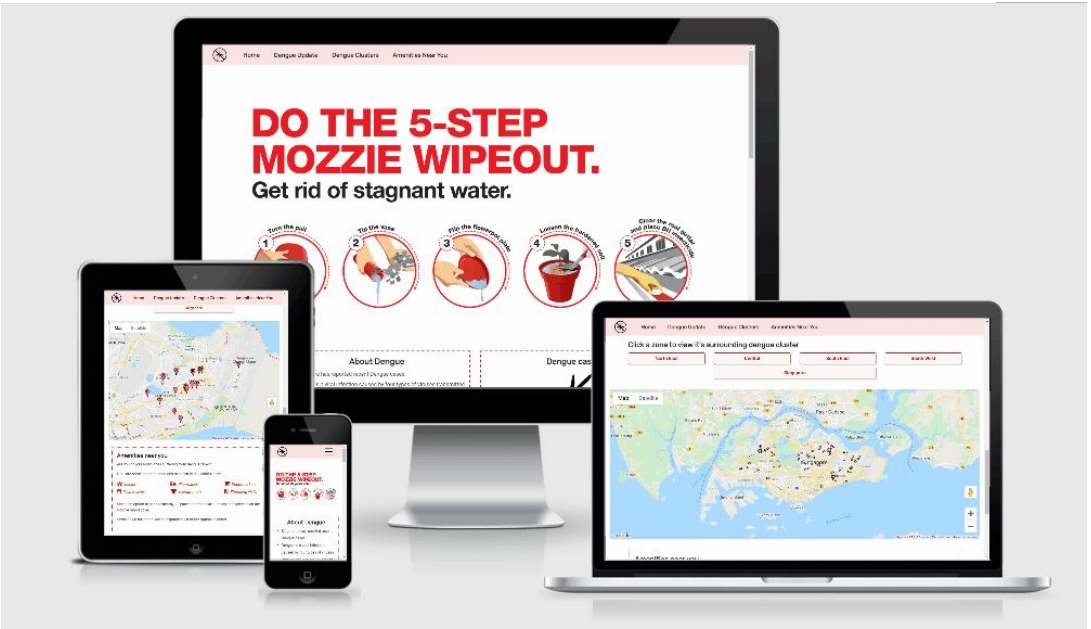
Method 2: Using Markdown

```
![Alt text](relative/path/to/img.jpg)
```

https://guides.github.com/features/mastering-markdown/#example-images

Demo

A live demo of the finished project can be found here.



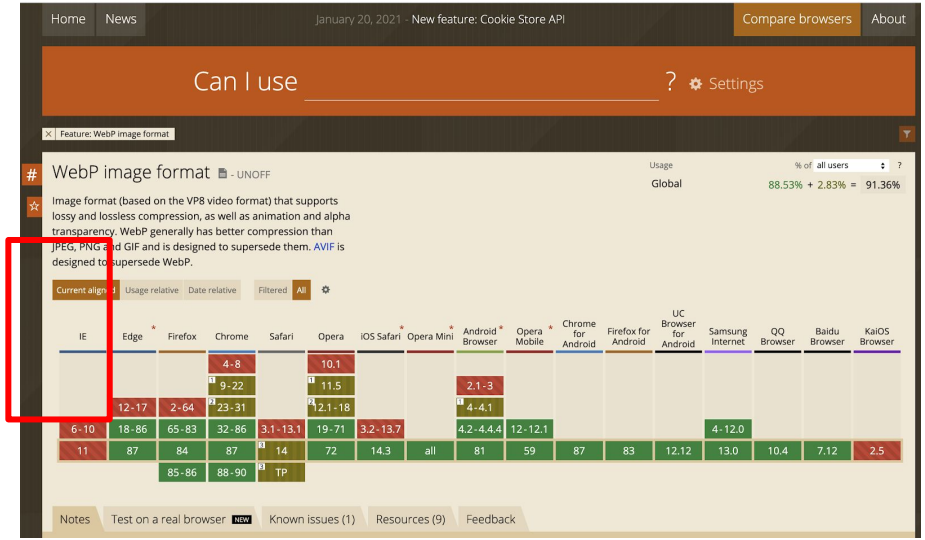
Example of how your README.md can be visual and descriptive just by using a screencapture gif

Checking the latest features

HTML5 has been actively progressing forward, however not all browsers support the latest features.

Check here first >>> https://caniuse.com/

Before using a new HTML API or feature, do check on caniuse.com to ensure that it is properly supported across the browsers you are developing for.



Example query of webp support https://caniuse.com/webp
Support for full webp is almost there!

Cross platform/browser Testing

Mac user but need to test on Microsoft Edge? How are we going to handle this?

Other than manually installing multiple browsers into our machine, we can automate those tests across browsers and platforms.

Test on different devices, different platforms

Test for responsive design

UI inconsistencies

If you are still testing in Internet Explorer… you are in trouble..

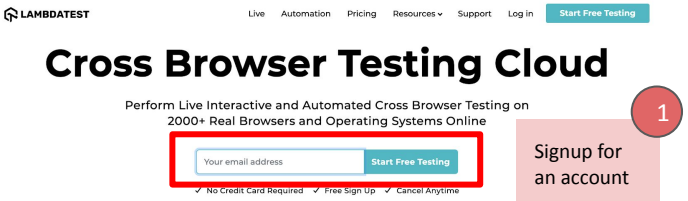
Tools & Resources

https://www.browserstack.com/cross-browser-testing
https://wptdashboard.appspot.com/
https://crossbrowsertesting.com/
https://www.lambdatest.com/ (Awesome)
http://browsershots.org/ (Great)

Reading

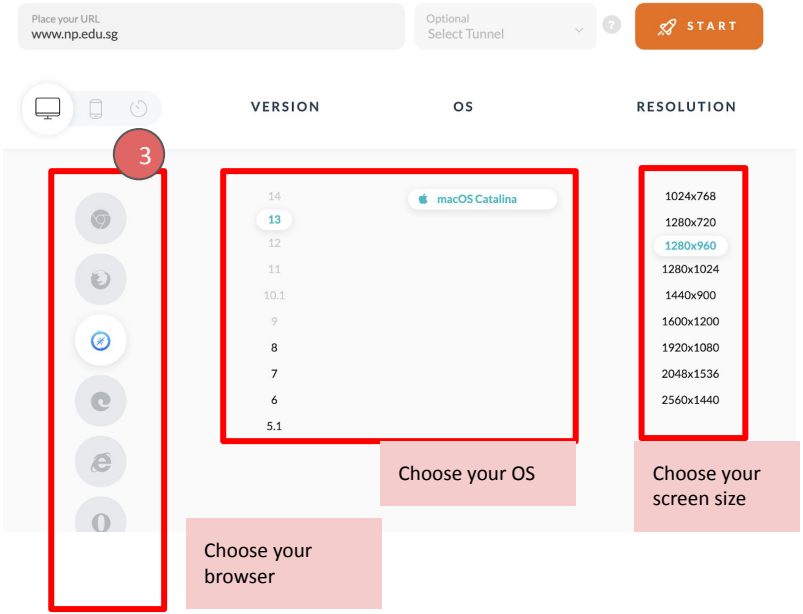
https://www.sitepoint.com/cross-browser-testing-tools/
https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Introduction

Cross platform/browser Testing w/ LambdaTest



1

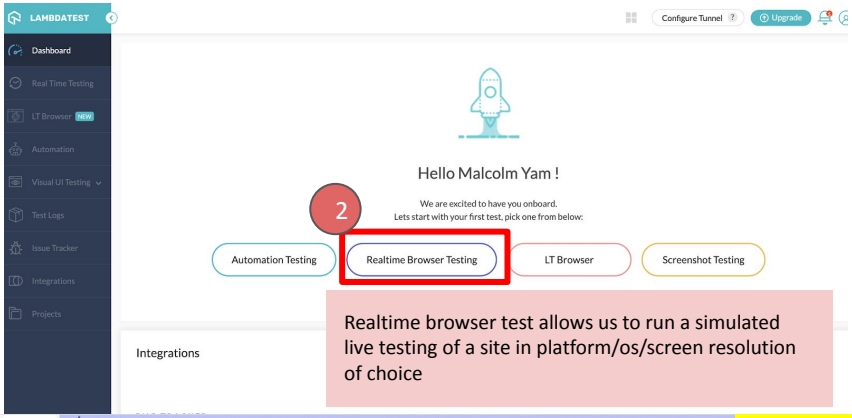
Signup for an account



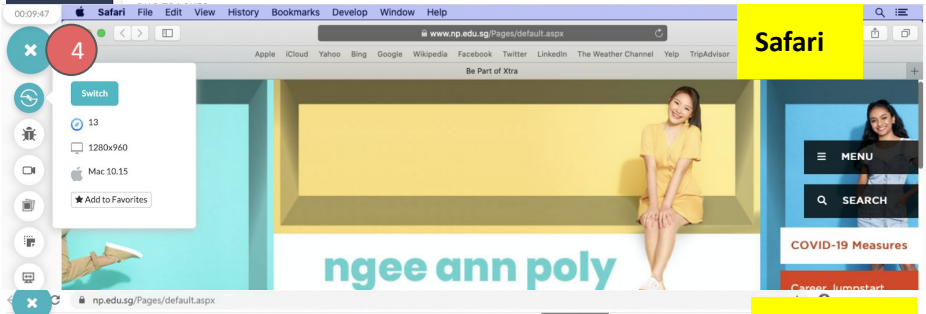
Choose your OS

Choose your screen size

Choose your browser



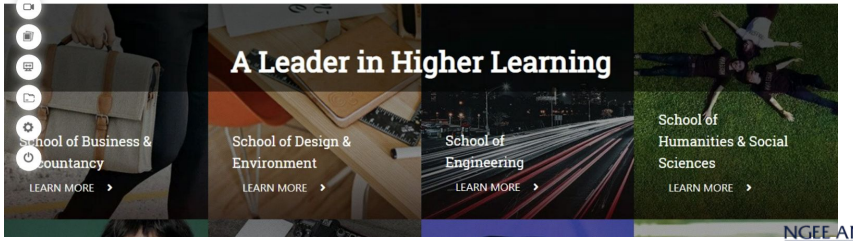
Realtime browser test allows us to run a simulated live testing of a site in platform/os/screen resolution of choice



Safari

Chrome

The free version is pretty awesome. 10 mins of testing :)



Decent Code Editor

If you are still using repl.it for assignments, it's time to move on.

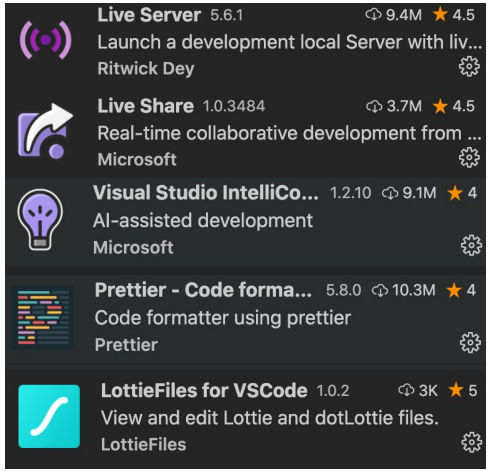
If you are using Dreamweaver still, it's fine if you good in it but it can be a hassle to modify and move your way around code and terrible for collaboration

Extensions for VS Code. Use with Caution
https://scotch.io/bar-talk/22-best-visual-studio-code-extensions-for-web-development

https://www.syncfusion.com/blogs/post/15-best-visual-studio-code-extensions-for-web-developers.aspx

Tools
https://code.visualstudio.com/ (recommended)
https://github.com/features/codespaces
http://brackets.io/
https://atom.io/

Great Extensions



Code Minifiers

Code minifies helps you to compress code so that it's lighter and loads faster.

The basic minifiers removes all unnecessary code, removes whitespace, link breaks, comments and so on.

Be sure to backup your original files before minifying. You should **ALWAYS have two files**, one minified and one unminified.

Use the minified one for production. Most of the tools simply require you to copy your code into their platform. They will generate a minified version for you.

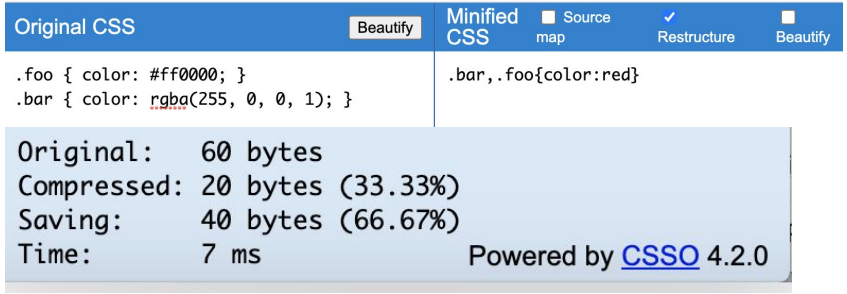
For JS codes, it helps in code obfuscation by making code a little more difficult to read.

CSS Minifiers

https://cssnano.co/playground/
http://css.github.io/csso/csso.html

JS Minifiers

https://closure-compiler.appspot.com/home
https://www.minifier.org/
https://javascript-minifier.com/



Example of CSS minification using CSSO
https://css.github.io/csso/csso.html

BOOTSTRAP

Bootstrap is a wonderful CSS framework for doing responsive designs and layouts. Fast and extremely well-documented, the basic of bootstrap can be easily understood.
Designing responsive forms and tables can also be easily accomplished with bootstrap.
https://getbootstrap.com/

Email address

We'll never share your email with anyone else.

Password

Check me out

Submit

Copy

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="ema
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Step 1: Install
https://getbootstrap.com/docs/5.0/getting-started/introduction/

Step 2: Do responsive layout
Understand their layout system using classes of containers, rows, columns
https://getbootstrap.com/docs/5.0/layout/grid/

Step 3: Doing responsive forms
https://getbootstrap.com/docs/5.0/forms/overview/

It's best to start with their documentation samples in their documentation, then move on from there. It is much faster :)
Their examples in the site can be downloaded and modified to your liking.

Use their search tool to your advantage when finding sample

Search docs... Ctrl + /

- Getting started
- Customize
- Layout
- Content
- Forms
  - Overview

Sample responsive form design and code

TRANSITING BETWEEN PAGES (CODE)

You designed multiple pages in your site, but realise your code doesn't translate or "saved" across the pages. Then you change your design into a single page application..
Or you might have submitted a form but the data is not stored or saved…

Sounds Familiar?

It's time to use LocalStorage to your advantage.

https://repl.it/@malcolmyam/simple-transition-pages-localstorage

Look at script.js that's being used in index.html
Then look at another.html's Javascript code whereby we retrieve the localStorage item.

VALIDATION

Code checking

VALIDATION

Why we validate our code? We want to check that the code is aligning towards the latest standards or "what is right", help you catch mistakes that you might missed.

Every programming language out there, there's a certain way to do things proper, certain formatting, certain way of calling things that is standardised.

One of the main reasons for standardisation is….. Anyone can takeover and at least know what is going on.

Reading
https://vanseodesign.com/web-design/validating-code/

Checking out HTML/Website is A-OK
Checks for HTML standards
Learn from best practises

https://validator.nu/
https://observatory.mozilla.org/
http://watson.addy.com/

Checking our CSS is A-OK
https://jigsaw.w3.org/css-validator/

Form Validation
https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation
How to in Validate forms in JS:
https://www.freecodecamp.org/news/form-validation-with-html5-and-javascript/
https://www.w3schools.com/js/js_validation.asp

Using Library to help
https://formvalidation.io/

Key Takeaway?

Work on it

Takes quite a sum of practise to get the hang of using promises in combination with async/await and the different ways of interfacing with an API

ALWAYS read the API documentation.

It's going to save you a great deal of time

Understanding JSON

1. **JSON** starts with curly brace **{ }**
2. There's a key/property in each value

```
let simpleJSON = {
  "key1" : "value1",
}
```

how we access?
simpleJSON.**<key1>** --> simpleJSON.key1 (retrieves value1)

3. Sometimes, arrays are embedded inside

```
{
"key": [
  object1,
  object2
]
}
```

In our object we can have our object key/value pairs
Sample Object

```
{
"studentid" : "T93",
"studentname": "Uncle Roger"
}
```

KEY

```
{
"studentList": [
  {
    "studentid" : "T93",
    "studentname": "Uncle Roger"
  },
  {
    "studentid" : "U491",
    "studentname": "Uncle Phil"
  }
]
}
```

To access object 1 studentid??
studentList[0].studentid --> because it's the first item in the array. Take note of the square brackets

Fetch

Additional

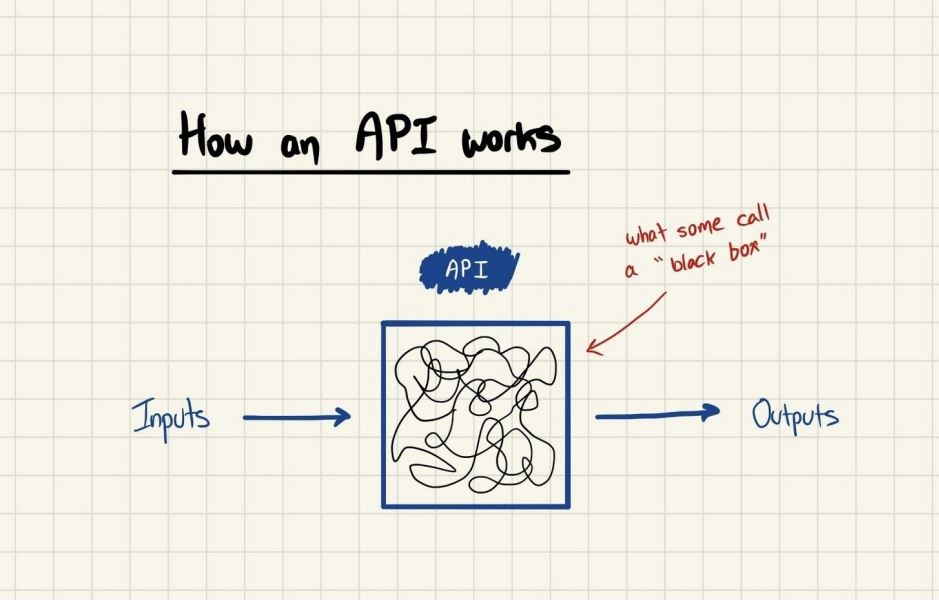
Learn Fetch API in 6 mins
https://www.youtube.com/watch?v=cuEtnrL9-H0

Fetch API Introduction
https://www.youtube.com/watch?v=Oive66jrwBs

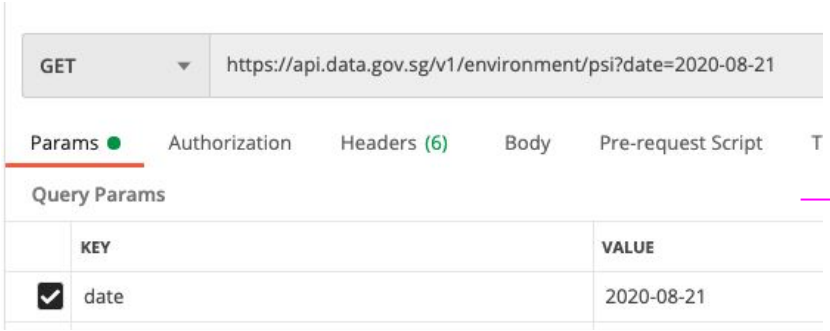
Fetch API: https://www.youtube.com/watch?v=g6-ZwZmRncs

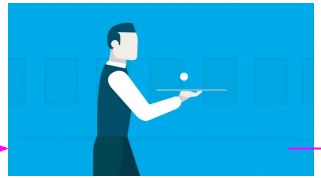
What is an API

An API is a group of logic that takes a specific input and gives you a specific output.



- If you give the **Google Maps API** an address as an *input*, it gives you back that addresses lat / long coordinates as an *output*
- If you give the **Grab Driver API** a start and finish address as an *input*, it finds the best driver as an *output*
- If you give the **PSI API a** date and time as an *input*, it finds the PSI levels for the date time specified





API


```
{
  "region_metadata": […
  ],
  "items": [
    {
      "timestamp": "2020-08-21T01:00:00+08:00",
      "update_timestamp": "2020-08-21T01:08:52+08:00",
      "readings": {
        "o3_sub_index": {
          "west": 3,
          "national": 6,
          "east": 5,
          "central": 6,
          "south": 5,
          "north": 6
        },
        "pm10_twenty_four_hourly": {
```


Reference: https://technically.substack.com/p/whats-an-api

What is an API

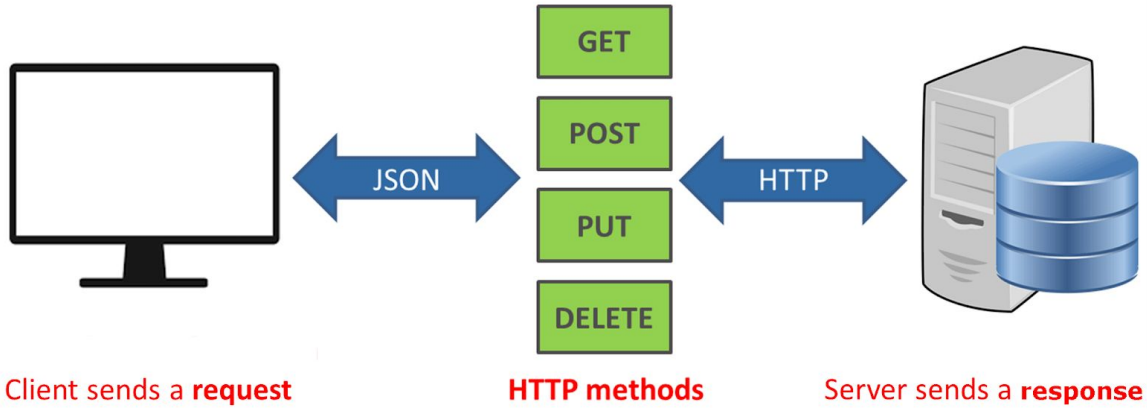
Applications are just a bunch of functions that get things done: **APIs wrap those functions** in easy to use interfaces so you can work with them without being an expert. Let's start with an example. If you're an e-commerce company, there are a bunch of things you need to get done internally that power your site:

- Show available items and sizes
- Create orders
- Update an email address

When you give an API a bunch of inputs to get the outputs you want, it's called **calling the API**.

What is an API

REST APIs work via HTTP, which means you need to use a request-response model: you make a request with a bunch of details, and the server sends back the data you asked for (or a message telling you that something went wrong). Here's how it breaks down:



https://phpenthusiast.com/blog/what-is-rest-api

What is a REST API

Video: https://www.youtube.com/watch?v=7YcW25PHnAA

More on JSON

Learn JSON in 10 minutes: https://www.youtube.com/watch?v=iiADhChRriM

How a request is formed.

1. Endpoint: It is the URL where the REST Server is listening.
https://ipapi.co/json List IP details
https://api.data.gov.sg/v1/environment/psi?date=2020-08-21 - List PSI data on 2020-08-21

2. Method: REST implements multiple 'methods' for different types of request, the following are most popular:

- **GET**: Get resource from the server.
- **POST**: Create resource to the server.
- **PATCH** or **PUT**: Update existing resource on the server.
- **DELETE**: Delete existing resource from the server.

3. Headers: The additional details provided for communication between client and server (remember, REST is stateless). Some of the common headers are:
Request:
- *host*: the IP of client (or from where request originated)
- *accept-language*: language understandable by the client
- *user-agent*: data about client, operating system and vendor

Response:
- *status*: the status of request or HTTP code.
- *content-type*: type of resource sent by server.
- *set-cookie*: sets cookies by server

4. **Data**: (also called body or message) contains info you want to send to the server.

QNS: Why my API works in POSTMAN but cannot work in my code?

Sometimes, APIs companies restrict their API access and also prevent based on authentication, URLs, restrictions, whitelisting etc. Many different ways.

The most common issue is CORS.
Cross origin Resource Sharing mechanism that restricts APIs to be used.
https://www.codecademy.com/articles/what-is-cors

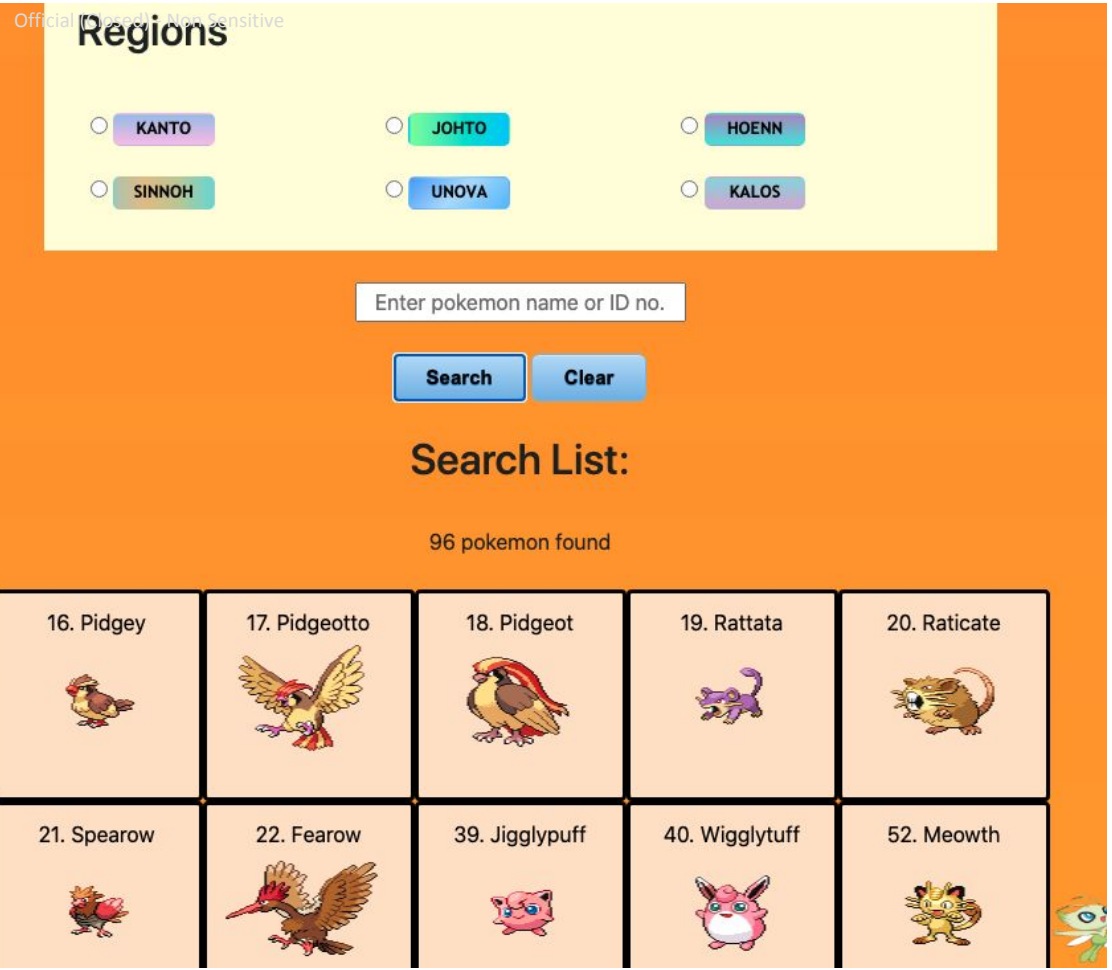
We can bypass this issue to a limited extend by implementing cors-anywhere (this is an external call made by a developer)
https://github.com/Rob--W/cors-anywhere

How we implement it?
We simply add on the cors-anywhere URL to our own API endpoint

```
Origin endpoint: http://datamall2.mytransport.sg/ltaodataservice/BusArrivalv2?BusStopCode=83139
New Endpoint:
https://cors-anywhere.herokuapp.com/http://datamall2.mytransport.sg/ltaodataservice/BusArrivalv2?BusStopCode=83139&=",
```

https://replit.com/@immalcolm/cors-issue-fix-ltabusapi-test

Do use your own account keys
Signup for LTA API access: https://www.mytransport.sg/content/mytransport/home/dataMall.html

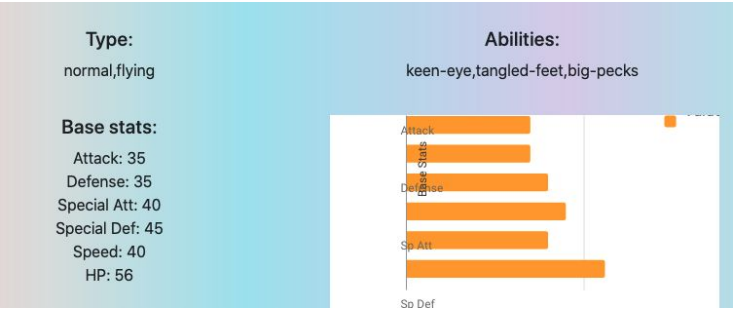


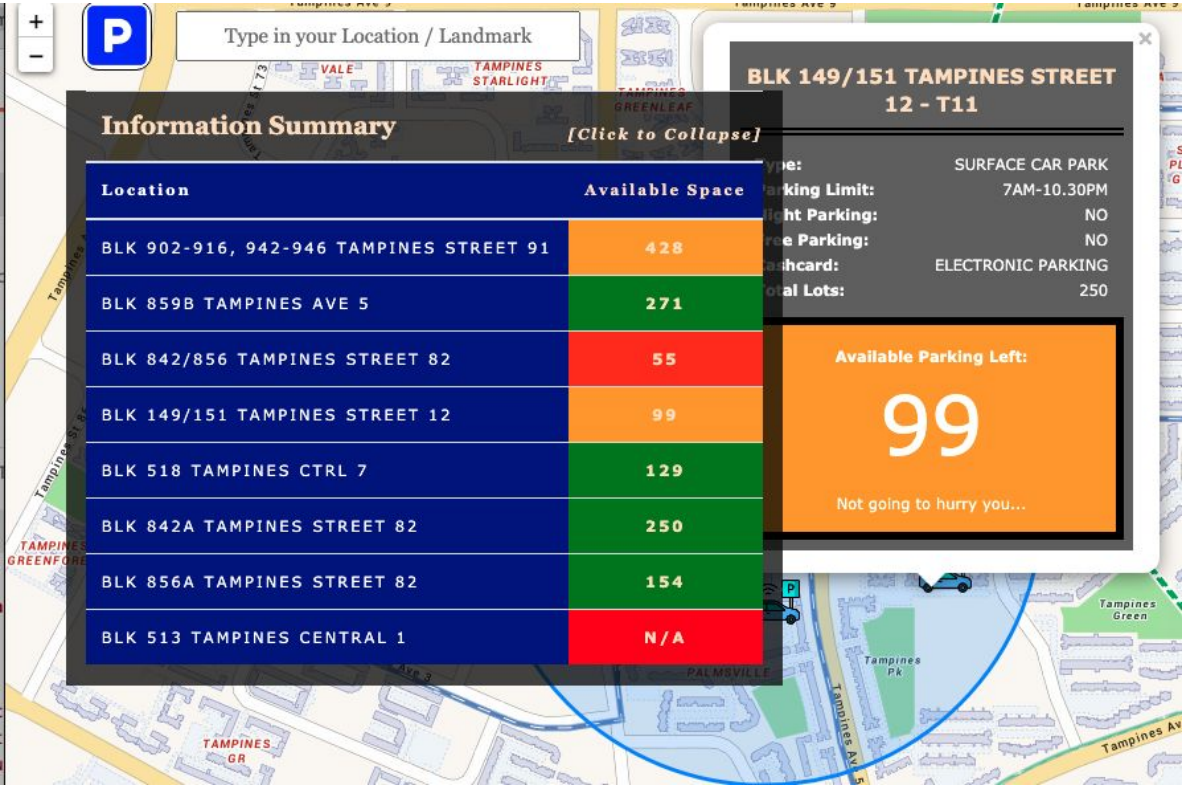
"Pokedex"

Bootstrap
jQuery
API Used: Pokemon API, localStorage, Google Charts
https://pokeapi.co/

Problem Statement?

- Many generations of Pokemon, All in one handler for it





Parking App Desktop/Mobile

Bootstrap
jQuery
Leaflet JS
API Used: OneMapSG, Data.gov.sg (Carpark Availablity)

https://docs.onemap.sg/
https://data.gov.sg/dataset/carpark-availability
https://www.mapbox.com/

Some Sample APIS

https://any-api.com
https://dev.twitter.com/
http://www.dex.sg/collections/
https://developers.data.gov.sg
https://www.nea.gov.sg/api/
https://github.com/public-apis/public-apis
https://rapidapi.com/collection/list-of-free-apis
https://rapidapi.com/
https://apilist.fun/
https://public-apis.xyz/
https://www.mapbox.com/
https://www.mas.gov.sg/development/fintech/financial-industry-api-register
https://docs.onemap.sg/#onemap-rest-apis
https://github.com/TonnyL/Awesome_APIs
https://dev.to/camerenisonfire/10-intriguing-public-rest-apis-for-your-next-project-2gbd
https://public-apis.io/
http://www.omdbapi.com/
https://the-one-api.dev/
https://www.reddit.com/dev/api/
https://spoonacular.com/food-api
https://nominatim.org/release-docs/develop/api/Search/
https://wger.de/

To create a NoSQL database with RESTful APIs?
https://restdb.io/

ICONS?

https://www.flaticon.com/
https://thenounproject.com/
https://icons8.com/
https://freeicons.io/
https://www.canva.com/learn/free-icons-download/

Charts?

https://www.chartjs.org/
https://repl.it/@malcolmyam/wk0x-simplecharts#index.html