accounting_data_2019.js

```
module.exports = {
  code: 'ACCT',
  name: 'Accounting',
  url:
    'https://www.handbook.unsw.edu.au/Accounting/browse?sa=b4cecfec4fcb5b00eeb3eb4f0310c7eb',
  courses: [
    {
      name: 'Accounting and Financial Management 1A',
      study_level: 'undergraduate',
      code: 'ACCT1501',
      keywords: 'accounting',
      description:
        'The compulsory core accounting unit will have a preparer perspective. It will provide an introduction to
      handbook_url:
        'https://www.handbook.unsw.edu.au/undergraduate/courses/2019/ACCT1501',
      outline_url:
        'https://www.business.unsw.edu.au/degrees-courses/course-outlines/ACCT1501',
      requirements:
        'Only available to the Business School single and double degree students in semester 1. It will be offere
    },
    {
      name: 'Accounting and Financial Management 1B',
      study_level: 'undergraduate',
      code: 'ACCT1511',
      keywords: 'accounting',
      description:
        'During Summer Term, this course is available as General Education to students from faculties outside the
      handbook_url:
        'https://www.handbook.unsw.edu.au/undergraduate/courses/2019/ACCT1511',
      outline_url:
        'https://www.business.unsw.edu.au/degrees-courses/course-outlines/ACCT1511',
      requirements: 'Prerequisite: ACCT1501'
...
```

events.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <link rel="stylesheet" href="/assets/css//basic.css">
      <meta charset="utf-8">
      <title>My first JavaScript</title>
      <style>
          h1 {
              transition: color 0.3s ease-in-out;
          }
      </style>
   </head>
   <body>
      <h1>When you click this header it will change colour</h1>
      <script src="dynamic.js"></script>
   </body>
</html>
```

rest.js

```
// [ ... ]
/*

const name = 'Alex';

'Hello' + ' ' + name;

`Hello ${name}`;
*/

const person = {
    name: 'Zain',
    age: 100,
    speed: 7.5
};

// example of inline renaming, and destructure syntax
const { name: zain_name, age, speed } = person;

console.log(zain_name, age, speed);

// example of inline destructure syntax (in a function)
function personProcessor({ name, age }) {
    console.log(name, age);
}

personProcessor(person);

// example of spread syntax
function printArguments(...args) {
    args.map(arg => console.log(arg));
}

// example of using rest/spread syntax to concatentate two arrays
const girls = ['Sally', 'Lakshi', 'Sophia', 'Tilly'];
const boys  = ['Andrew', 'Barry', 'Tobias', 'Prashant'];

const names = [...girls, ...boys];

console.log(names); // ['Sally', 'Lakshi', 'Sophia', 'Tilly', 'Andrew', 'Barry', 'Tobias', 'Prashant'];

// can give this function any number of args.
```

battery.js

```
const battery = {
    powerLevel: 10,
    capacity: 50
};

export default battery;
```

function.js

```
function getBatteryPower(battery) {
    return battery.powerLevel;
}

export { getBatteryPower };
```

shopping.js

```
const shoppingCart = [
    { item: 'Apple', price: 10 },
    { item: 'Orange', price: 12 },
    { item: 'Pineapple', price: 5 }
 ];

 /* think about what this functions do */
 const multiply = a => b => a * b;
 const pluck = key => object => object[key];

 // let's say tax of 10% for GST and a 5 % first customer discount
 const discount = multiply(0.95);
 const tax = multiply(1.10);

 // the format required for sum
 const sum = (acc, curr) => curr + acc;

 // Now, for some simple readable, easy to reason about code.
 const totalPrice = shoppingCart
     .map(pluck('price'))
     .map(discount)
     .map(tax)
     .reduce(sum, 0);

console.log(totalPrice);
```

dom.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <title>Dom Example</title>
   </head>
   <body>
      <h1>Dom Example</h1>
      <div id="output"></div>
      <!-- <p>Hello Andrew</p> will be inserted here -->
      <script src="dom.js"></script>
   </body>
</html>
```

functional.js

```
const ACCT = require('./accounting_data_2019');

// console.log(ACCT);

const courses = ACCT.courses.map(({ name, code, study_level: level }) => {
    return {
        name,
        code,
        level
    };
});

console.log(courses);
```

simple.js

```
import { getBatteryPower } from './function.js';
import battery from './battery.js';

console.log('Battery Import: ', getBatteryPower(battery));
```

dom.js

```
/* Super basic */

// selecting an element
const output = document.getElementById('output');

// creating an element
const p = document.createElement('p');

// adding it to the dom tree
// need to think about how to add it, ie what parent it needs.
output.appendChild(p);

// manipulating an element
p.innerText = 'Hello Andrew!';
```

import.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <title>Teaching</title>
      <link rel="stylesheet" href="/assets/css/basic.css"/>
   </head>
   <body>
      <h1>Teaching</h1>
      <p>Just some random html for the purposes of showing module things.</p>
      <script src="battery.js"  type="module"></script>
      <script src="function.js" type="module"></script>
      <script src="simple.js"   type="module"></script>
   </body>
</html>
```

dynamic.js

```
// load events. JS is driven by the event loop
// We need to think about how to listen to events
// to react and have some response for the user

// An example of a common event is a click event
// To add an event listerner we need to bind it to an element
// so we can listen for it.
const h1 = document.getElementsByTagName('h1')[0];

h1.addEventListener('click', function() {
    const color = this.style.color;

    // 'this' points the h1 element
    this.style.color = color === 'red' ? 'purple' : 'red';
});


// A billion other types of event handlers exist. Lots of possibilities
```