week3

Note:

Task1:

A string can also be referred to as a string for short. In the data structure, a string is a linear table with certain constraints on the composition of data elements, that is, all data elements that make up the linear table are required to be characters, so the string is a finite sequence of characters.

Task2:

This talks about how to use array methods and number methods. We can use these methods to better complete JavaScript.

Task3:

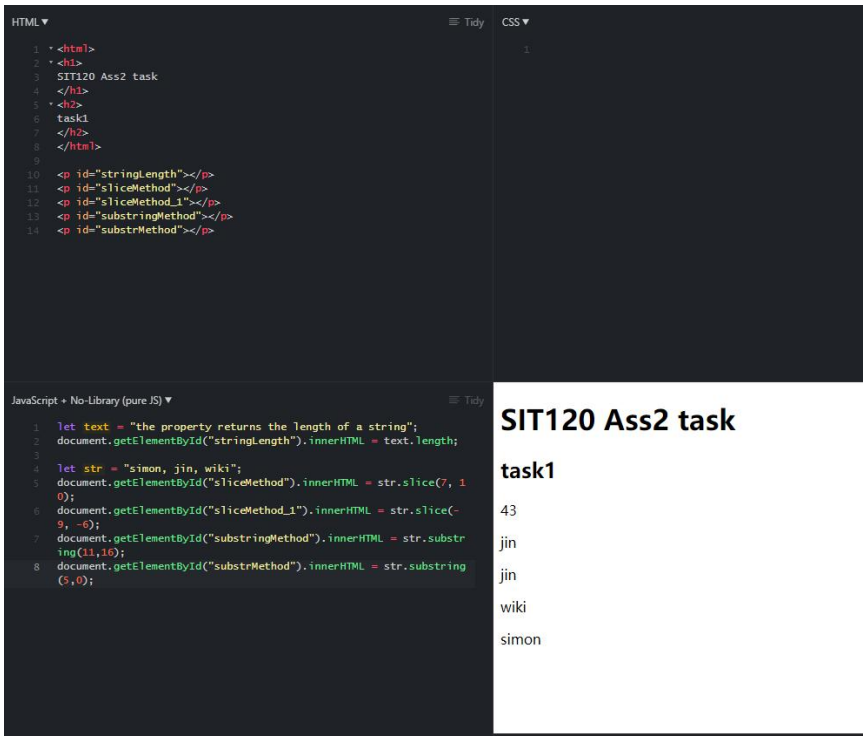
This taught me how to use get and set methods.

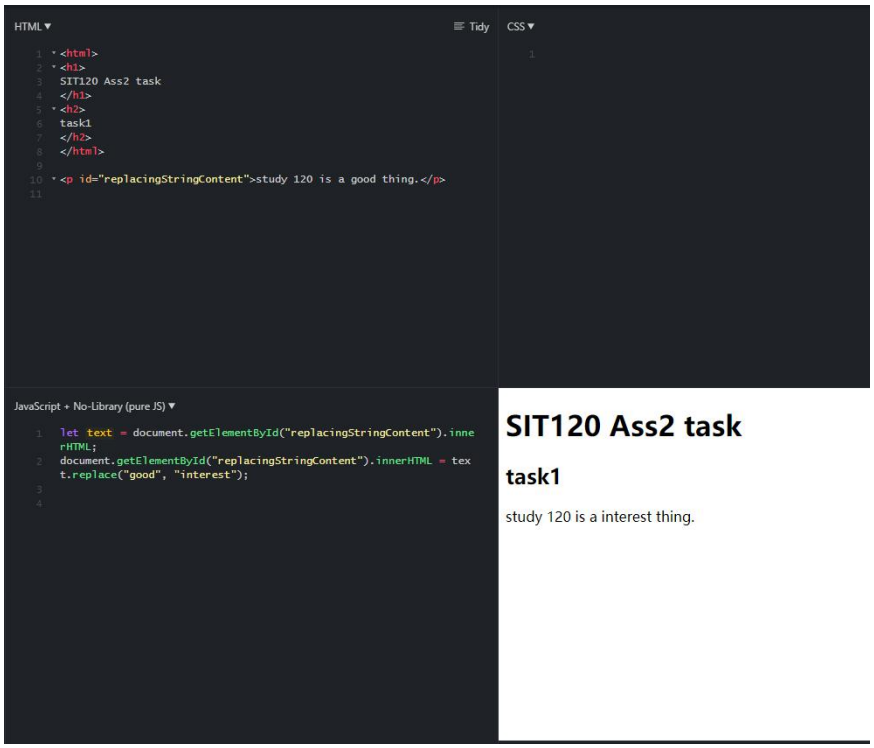
Task4:

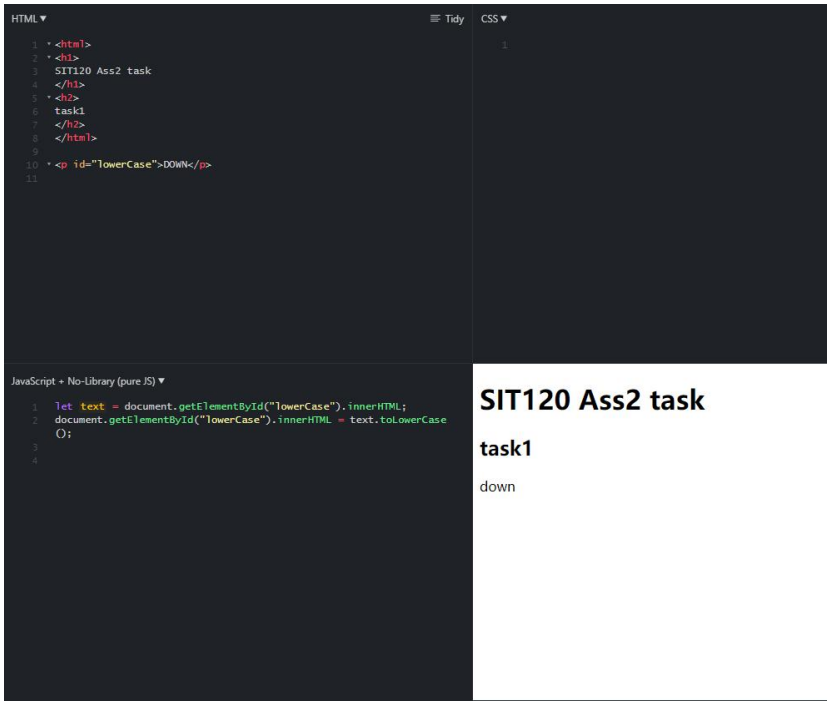
The explanation of the Vue components and element properties in Task 4 made me learn more and different knowledge, which enabled me to have a deeper understanding of VUE components.

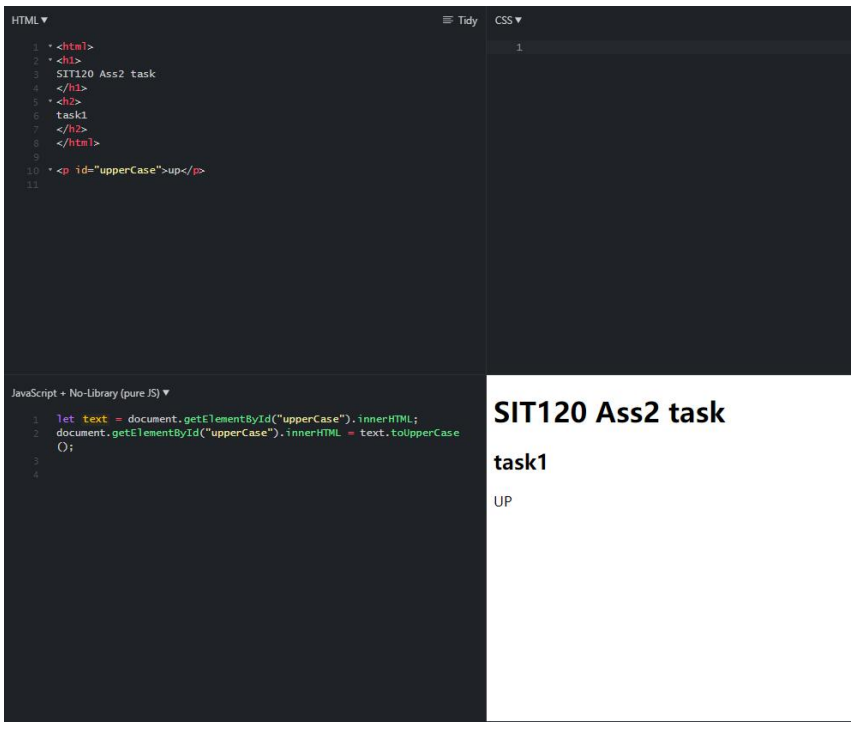
Screenshot:

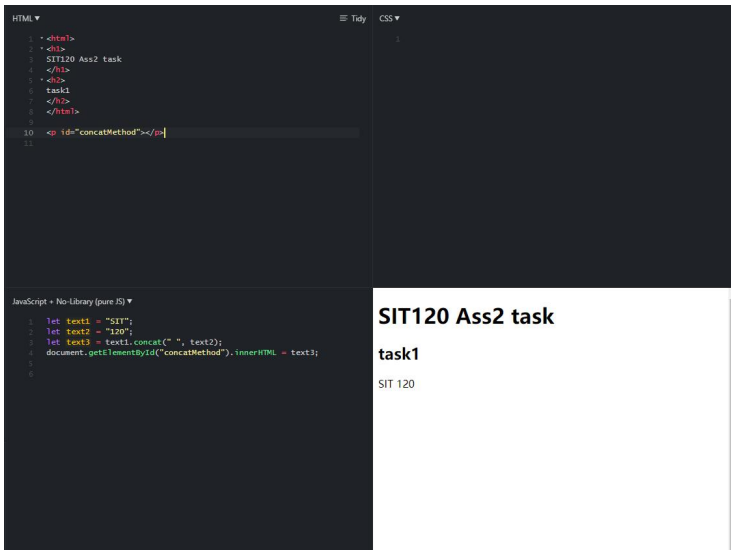
Task1



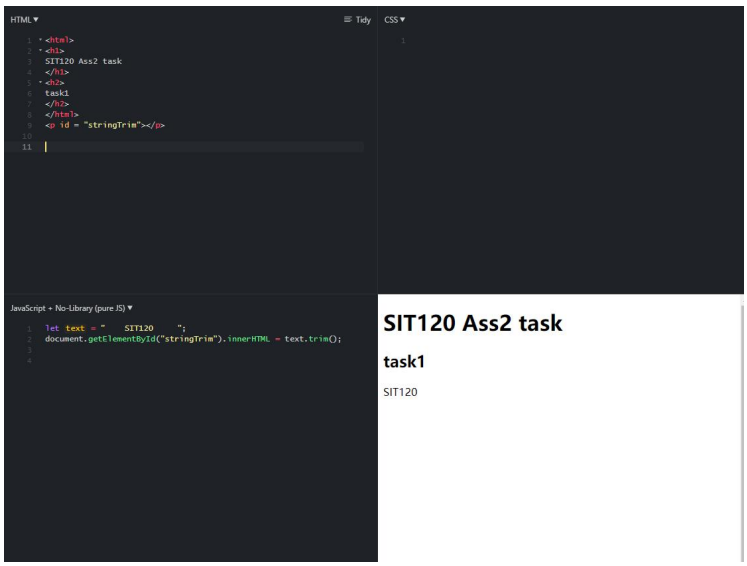



HTML
Tidy
CSS
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>
DOWN
JavaScript + No-Library (pure JS)
let text = document.getElementById("lowerCase").innerHTML;
document.getElementById("lowerCase").innerHTML = text.toLowerCase
();
SIT120 Ass2 task
task1
down

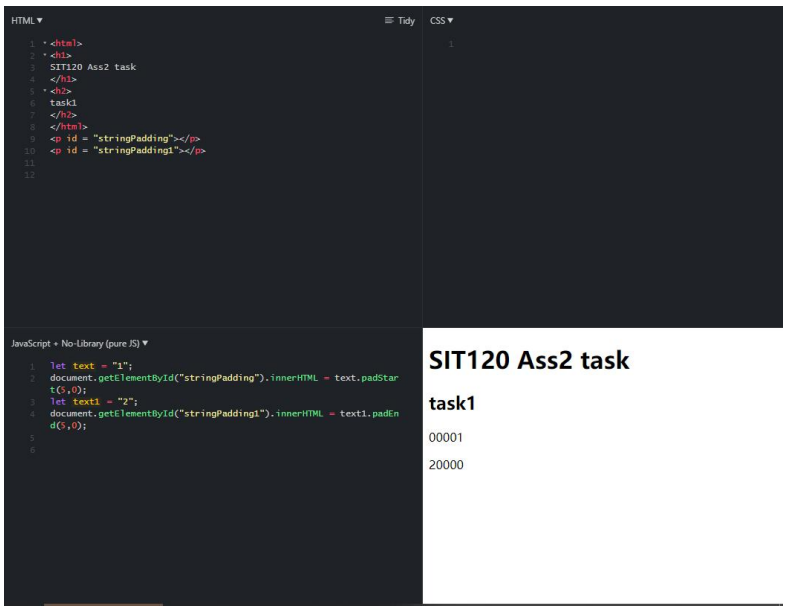

HTML
Tidy
CSS
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>
up
JavaScript + No-Library (pure JS)
let text = document.getElementById("upperCase").innerHTML;
document.getElementById("upperCase").innerHTML = text.toUpperCase
();
SIT120 Ass2 task
task1
UP


HTML
Tidy
CSS
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>

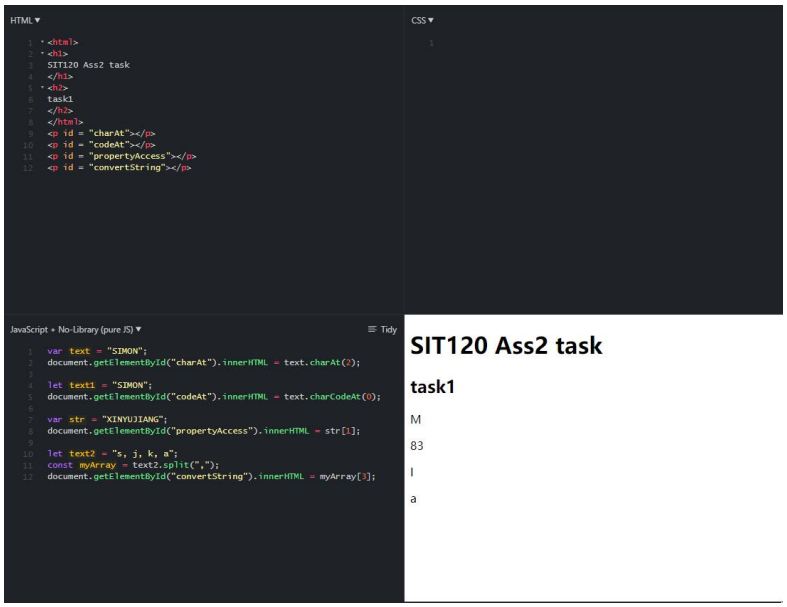
JavaScript + No-Library (pure JS)
let text1 = "SIT";
let text2 = "120";
let text3 = text1.concat(" ", text2);
document.getElementById("concatMethod").innerHTML = text3;
SIT120 Ass2 task
task1
SIT 120


HTML
Tidy
CSS
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>

JavaScript + No-Library (pure JS)
let text = " SIT120 ";
document.getElementById("stringTrim").innerHTML = text.trim();
SIT120 Ass2 task
task1
SIT120

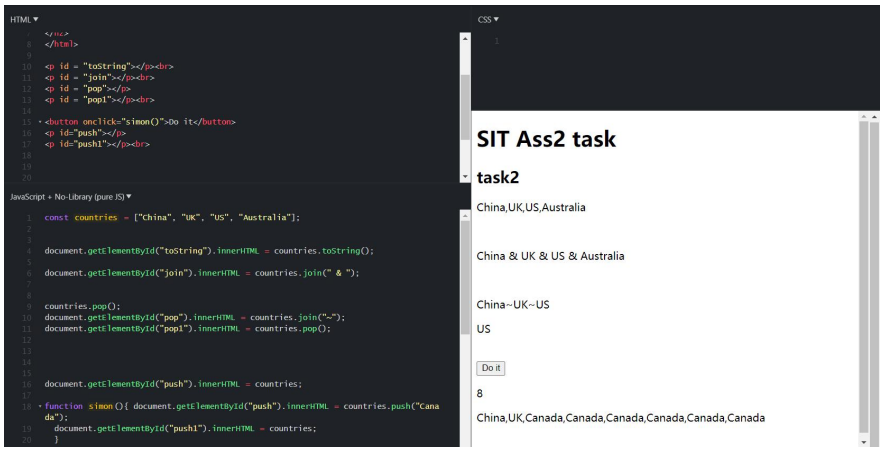

HTML
Tidy
CSS
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>

JavaScript + No-Library (pure JS)
let text = "1";
document.getElementById("stringPadding").innerHTML = text.padStart(5,0);
let text1 = "2";
document.getElementById("stringPadding1").innerHTML = text1.padEnd(5,0);
SIT120 Ass2 task
task1
00001
20000


HTML
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task1
</h2>
</html>

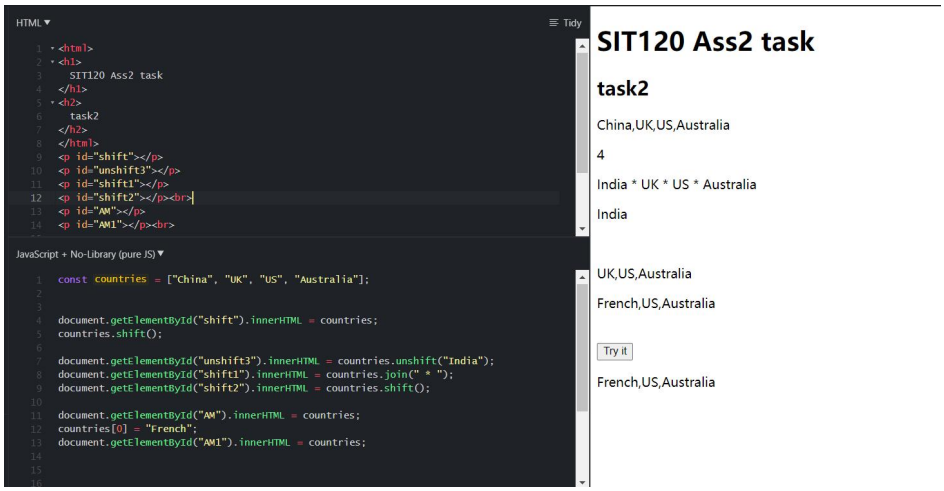
CSS
JavaScript + No-Library (pure JS)
Tidy
var text = "SIMON";
document.getElementById("charAt").innerHTML = text.charAt(2);
let text1 = "SIMON";
document.getElementById("codeAt").innerHTML = text.charCodeAt(0);
var str = "XINYUJIANG";
document.getElementById("propertyAccess").innerHTML = str[1];
let text2 = "s, j, k, a";
const myArray = text2.split(",");
document.getElementById("convertString").innerHTML = myArray[3];
SIT120 Ass2 task
task1
M
83
I
a

Task2

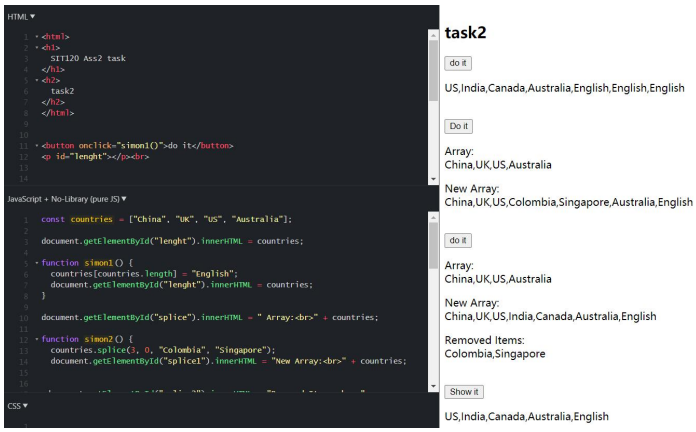

HTML
</html>

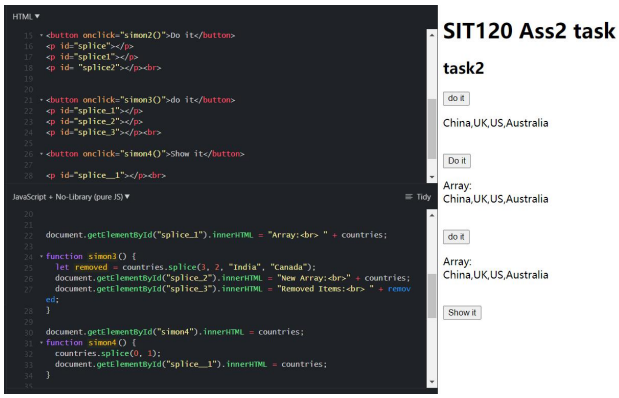
<button onclick="simon()">Do it</button>

CSS
JavaScript + No-Library (pure JS)
const countries = ["China", "UK", "US", "Australia"];
document.getElementById("toString").innerHTML = countries.toString();
document.getElementById("join").innerHTML = countries.join(" & ");
countries.pop();
document.getElementById("pop").innerHTML = countries.join("~");
document.getElementById("pop1").innerHTML = countries.pop();
document.getElementById("push").innerHTML = countries;
function simon(){ document.getElementById("push").innerHTML = countries.push("Canada");
document.getElementById("push1").innerHTML = countries;
}
SIT Ass2 task
task2
China,UK,US,Australia
China & UK & US & Australia
China~UK~US
US
Do it
8
China,UK,Canada,Canada,Canada,Canada,Canada,Canada


HTML
Tidy
<html>
<h1>
SIT120 Ass2 task
</h1>
<h2>
task2
</h2>
</html>

JavaScript + No-Library (pure JS)
const countries = ["China", "UK", "US", "Australia"];
document.getElementById("shift").innerHTML = countries;
countries.shift();
document.getElementById("unshift3").innerHTML = countries.unshift("India");
document.getElementById("shift1").innerHTML = countries.join(" * ");
document.getElementById("shift2").innerHTML = countries.shift();
document.getElementById("AM").innerHTML = countries;
countries[0] = "French";
document.getElementById("AM1").innerHTML = countries;
SIT120 Ass2 task
task2
China,UK,US,Australia
4
India * UK * US * Australia
India
UK,US,Australia
French,US,Australia
Try it
French,US,Australia





SIT120 Ass2 task

task2

do it

Australia,India,Canada,Colombia,Singapore,Colombia,Singapore,Colombia,Singa

Do it

Array:
China,UK,US,Australia

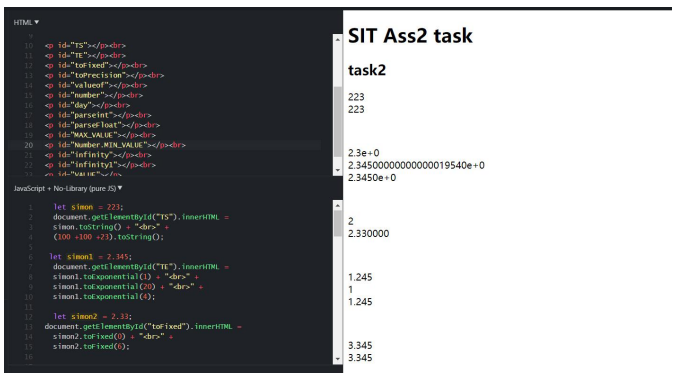
New Array:
Australia,India,Canada,Colombia,Singapore,Colombia,Singapore,Colombia,Singa

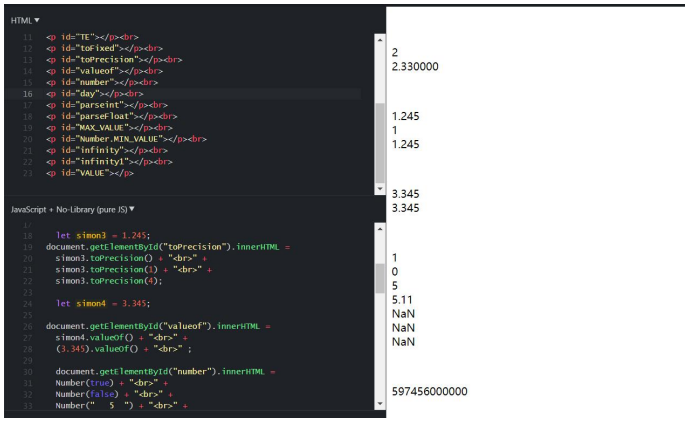
do it

Array:
China,UK,US,Australia

New Array:
Australia,India,Canada,India,Canada

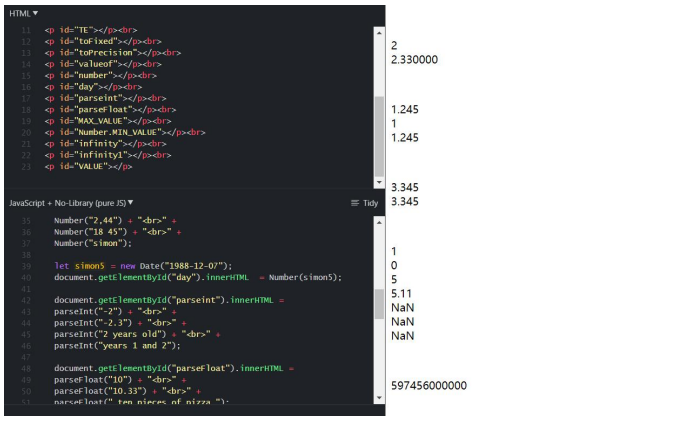
Removed Items:
India,Canada





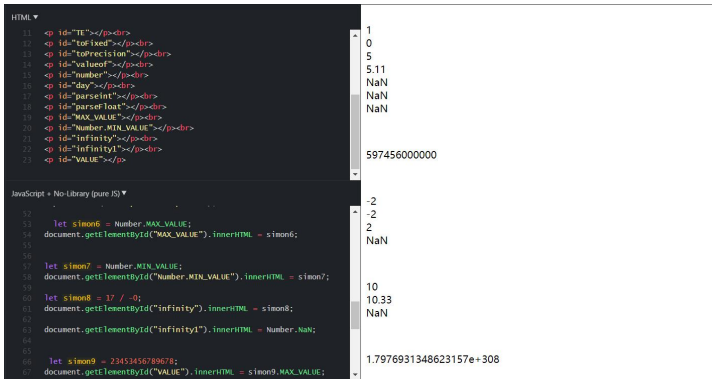
```
HTML
<p id="TE"></p><br>
<p id="toFixed"></p><br>
<p id="toPrecision"></p><br>
<p id="valueOf"></p><br>
<p id="number"></p><br>
<p id="day"></p><br>
<p id="parseint"></p><br>
<p id="parseFloat"></p><br>
<p id="MAX_VALUE"></p><br>
<p id="Number.MIN_VALUE"></p><br>
<p id="infinity"></p><br>
<p id="infinity1"></p><br>
<p id="VALUE"></p>

JavaScript + No-Library (pure JS)

let simon3 = 1.245;
document.getElementById("toPrecision").innerHTML =
simon3.toPrecision() + "<br>" +
simon3.toPrecision(1) + "<br>" +
simon3.toPrecision(4);

let simon4 = 3.345;

document.getElementById("valueOf").innerHTML =
simon4.valueOf() + "<br>" +
(3.345).valueOf() + "<br>" ;

document.getElementById("number").innerHTML =
Number(true) + "<br>" +
Number(false) + "<br>" +
Number("   5  ") + "<br>" +
```

```
2
2.330000


1.245
1
1.245


3.345
3.345


1
0
5
5.11
NaN
NaN
NaN


597456000000
```



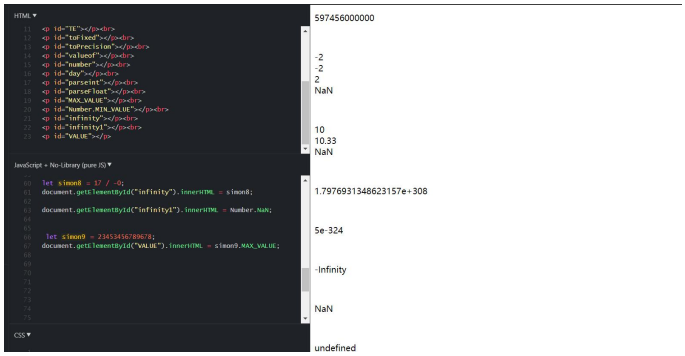
```
HTML
<p id="TE"></p><br>
<p id="toFixed"></p><br>
<p id="toPrecision"></p><br>
<p id="valueOf"></p><br>
<p id="number"></p><br>
<p id="day"></p><br>
<p id="parseint"></p><br>
<p id="parseFloat"></p><br>
<p id="MAX_VALUE"></p><br>
<p id="Number.MIN_VALUE"></p><br>
<p id="infinity"></p><br>
<p id="infinity1"></p><br>
<p id="VALUE"></p>

JavaScript + No-Library (pure JS)   Tidy

Number("2,44") + "<br>" +
Number("18 45") + "<br>" +
Number("simon");

let simon5 = new Date("1988-12-07");
document.getElementById("day").innerHTML  = Number(simon5);

document.getElementById("parseint").innerHTML =
parseInt("-2") + "<br>" +
parseInt("-2.3") + "<br>" +
parseInt("2 years old") + "<br>" +
parseInt("years 1 and 2");

document.getElementById("parseFloat").innerHTML =
parseFloat("10") + "<br>" +
parseFloat("10.33") + "<br>" +
parseFloat("  ten pieces of pizza  ");
```

```
2
2.330000


1.245
1
1.245


3.345
3.345


1
0
5
5.11
NaN
NaN
NaN


597456000000
```



```
HTML
<p id="TE"></p><br>
<p id="toFixed"></p><br>
<p id="toPrecision"></p><br>
<p id="valueOf"></p><br>
<p id="number"></p><br>
<p id="day"></p><br>
<p id="parseint"></p><br>
<p id="parseFloat"></p><br>
<p id="MAX_VALUE"></p><br>
<p id="Number.MIN_VALUE"></p><br>
<p id="infinity"></p><br>
<p id="infinity1"></p><br>
<p id="VALUE"></p>

JavaScript + No-Library (pure JS)

let simon6 = Number.MAX_VALUE;
document.getElementById("MAX_VALUE").innerHTML = simon6;

let simon7 = Number.MIN_VALUE;
document.getElementById("Number.MIN_VALUE").innerHTML = simon7;

let simon8 = 17 / -0;
document.getElementById("infinity").innerHTML = simon8;

document.getElementById("infinity1").innerHTML = Number.NaN;

let simon9 = 2345345678967 8;
document.getElementById("VALUE").innerHTML = simon9.MAX_VALUE;
```

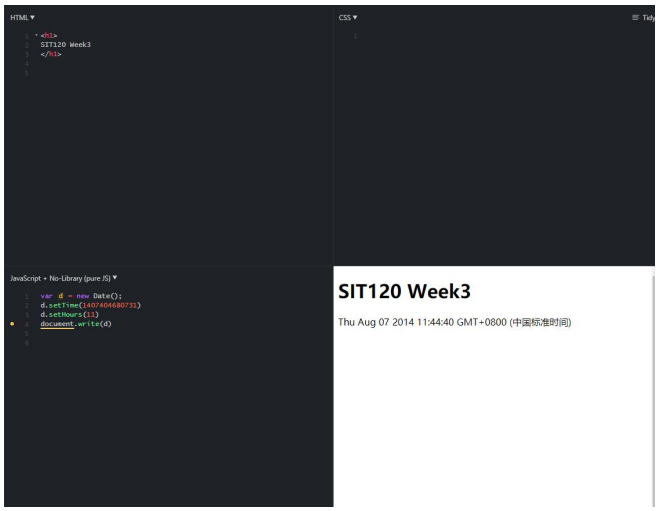
```
1
0
5
5.11
NaN
NaN
NaN


597456000000


-2
-2
2
NaN


10
10.33
NaN


1.7976931348623157e+308
```

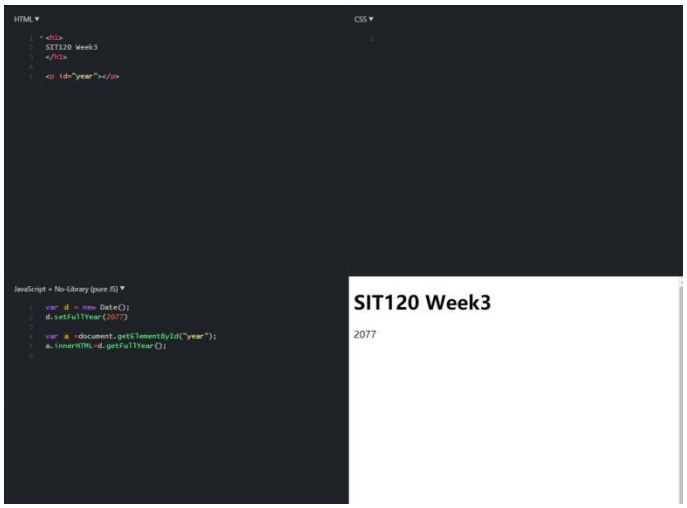


```
HTML
<p id="TE"></p><br>
<p id="toFixed"></p><br>
<p id="toPrecision"></p><br>
<p id="valueOf"></p><br>
<p id="number"></p><br>
<p id="day"></p><br>
<p id="parseint"></p><br>
<p id="parseFloat"></p><br>
<p id="MAX_VALUE"></p><br>
<p id="Number.MIN_VALUE"></p><br>
<p id="infinity"></p><br>
<p id="infinity1"></p><br>
<p id="VALUE"></p>

JavaScript + No-Library (pure JS)

let simon8 = 17 / -0;
document.getElementById("infinity").innerHTML = simon8;

document.getElementById("infinity1").innerHTML = Number.NaN;

let simon9 = 2345345678967 8;
document.getElementById("VALUE").innerHTML = simon9.MAX_VALUE;

CSS
```

```
597456000000


-2
-2
2
NaN


10
10.33
NaN


1.7976931348623157e+308


5e-324


-Infinity


NaN


undefined
```

Task3





Task4

Computer attributes :

Computer attributes are used to describe that the value of one attribute depends on the value of another attribute. When an interpolation expression is used to bind a calculated attribute to a page element, the calculated attribute will automatically update the DOM element when the dependent attribute value changes. When calculating some complex calculation problems, the calculation will be very prone to

errors as the difficulty of the calculation increases. At this time, we need a mechanism to help create and save the data that has been calculated to avoid errors. We can also use the watch listener method to monitor changes in certain data. The difference is that calculated attributes are only data operations performed after changes in dependent data, while watch focuses more on a series of business logic operations performed after a certain data in monitoring changes. In general, both computer properties and listeners are very efficient to help us manage our code.

Class and Style Bindings:

Class encapsulates properties and methods like a container, and it is used to manipulate its own members. Clss is the definition of a certain object and has behavior. It describes what an object can do and the method it can do, and the procedures and processes that it can operate on this object. It contains information about how the object behaves, including its name, properties, methods, and events.

In short: Class and Style Bindings help users bind the class list of data elements and their inline styles, and only calculate the final string after the data binding expression.

List Rendering:

The v-for instruction renders a list based on an array. It needs a special syntax of "item in items" items refers to the source data array. List rendering is mainly to render the data of the array to the specified place, simplifying the code workload, and the list rendering is to improve the performance of the loop display.

Event Handling:

In Event Handling, there are monitoring events, method event handlers, methods in inline handlers, event modifiers, Key Modifiers, and system modifiers. Event Handling makes programming easier and makes mistakes easier.

Form Input Bindings:

We can use the v-model command to create two-way data binding on the form <input>, <textarea> and <select> elements. It will automatically select the correct method to update the element based on the control type.

Two-way binding makes it possible for each form control to have a record variable in memory that corresponds to it in two directions. Regardless of whether it is

requesting an operation on the variable in the memory or the view is changed, the corresponding value will be updated.

Components Basics:

In this module, it contains a lot of useful components, such as reusable components, which can be reused as many times as needed. Organizing components can organize the application into a nested component tree, and props instructs it to make props pass data to sub-components. , A Single Root Element it can add any content, Listening to Child Components Events it can add the postFontSize data property in the main function to facilitate us to control the sub-function), dynamic components in general: these components make it easier for us to write code.

Component Registration:

Component Registration includes Global Registration and Local Registration. Local Registration means that local registration is to register and import in the vue file you want to use. Global registration is usually not ideal because it increases the amount of JavaScript that users must download and adds unnecessary workload.

Props:

Props include (camelCase vs kebab-case) (static, dynamic Props), props verification, one-way data flow, non-props properties, where props can pass numbers, Boolean values, arrays, and object properties. Props verification: The data specifications of the passed props parameters can be verified. If the data specifications are not met, a warning will be issued. All props form a one-way downward binding between the child property and the parent property. Non-props properties: are the properties passed to the component, but the corresponding props are not defined.

Custom Events：

This module contains many useful modules such as event names, but it should be noted that it does not provide any automatic case conversion. The name of the emitted event must exactly match the name used to listen for the event. v-model uses value as props and input events on components, and binds native events to components: it can directly listen to native events on the root element of the component. We can "two-way binding" props and use sync to decorate symbol.

Slots：

If we want to display the inserted new label in the subcomponent, we can use slots.

Dynamic & Async Components：

Keep-alive will cache inactive component instances to preserve component state or avoid re-rendering. If we need to load these components after successful asynchronous requests in the project, we can use Async Components. It allows us to reduce a lot of work.

Handling Edge Cases

Handling Edge Cases include element and component access. First, Accessing the Root Instance is very convenient for demonstrations or applications with a small number of components. Accessing the Parent Component Instance, and then Accessing Child Component Instances & Child Elements use the ref attribute to assign child components A reference ID, and finally Dependency Injection, Programmatic Event Listeners are used to listen for events. In general: the small components of this module can help users to manipulate the code more easily.