

## 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

1

2

3

4

5

6

7

8

9

10

11

12

13

14

Tidy

CSS

1

JavaScript + No-Library (pure JS)

Tidy

1

2

3

4

5

6

7

8

SIT120 Ass2 task

task1

43

j

j

wiki

simon

HTML

1

2

3

4

5

6

7

8

9

10

11

Tidy

CSS

1

JavaScript + No-Library (pure JS)

Tidy

1

2

3

4

SIT120 Ass2 task

task1

study 120 is a interest thing.

HTML▼

Tidy

CSS▼

```
1 <html>
2 <h1>
3 SIT120 Ass2 task
4 </h1>
5 <h2>
6 task1
7 </h2>
8 </html>
9
10 <p id="lowerCase">Down</p>
11
```

JavaScript + No-Library (pure JS)▼

```
1 let text = document.getElementById("lowerCase").innerHTML;
2 document.getElementById("lowerCase").innerHTML = text.toLowerCase();
3
4
```

SIT120 Ass2 task

task1

down

HTML▼

Tidy

CSS▼

```
1 <html>
2 <h1>
3 SIT120 Ass2 task
4 </h1>
5 <h2>
6 task1
7 </h2>
8 </html>
9
10 <p id="upperCase">up</p>
11
```

JavaScript + No-Library (pure JS)▼

```
1 let text = document.getElementById("upperCase").innerHTML;
2 document.getElementById("upperCase").innerHTML = text.toUpperCase();
3
4
```

SIT120 Ass2 task

task1

UP

HTML ▼

Tidy

CSS ▼

1

```
1 <html>
2 <div>
3   SIT120 Ass2 task
4 </div>
5 <div>
6   task1
7 </div>
8 </html>
9
10 <p id="concatMethod"></p>
11
```

JavaScript + No-library (pure JS) ▼

```
1 let text1 = "SIT1";
2 let text2 = "120";
3 let text3 = text1.concat(" ", text2);
4 document.getElementById("concatMethod").innerHTML = text3;
5
6
```

SIT120 Ass2 task

task1

SIT 120

HTML ▼

Tidy

CSS ▼

1

```
1 <html>
2 <div>
3   SIT120 Ass2 task
4 </div>
5 <div>
6   task1
7 </div>
8 </html>
9
10 <p id = "stringTrim"></p>
11
```

JavaScript + No-library (pure JS) ▼

```
1 let text = "  SIT120  ";
2 document.getElementById("stringTrim").innerHTML = text.trim();
3
4
```

SIT120 Ass2 task

task1

SIT120

HTML ▼

Tidy

CSS ▼

1

```
1 <html>
2 <div>
3   SIT120 Ass2 task
4 </div>
5 <div>
6   task1
7 </div>
8 </html>
9
10 <p id = "stringPadding"></p>
11 <p id = "stringPadding1"></p>
12
```

JavaScript + No-library (pure JS) ▼

```
1 let text = "1";
2 document.getElementById("stringPadding").innerHTML = text.padStart(5,0);
3 let text1 = "2";
4 document.getElementById("stringPadding1").innerHTML = text1.padEnd(5,0);
5
6
```

SIT120 Ass2 task

task1

00001  
20000

HTML

```
1 <html>
2 <h1>
3 SIT120 Ass2 task
4 </h1>
5 <div>
6 task1
7 </div>
8 </html>
9 <p id = "charAt"></p>
10 <p id = "codeAt"></p>
11 <p id = "propertyAccess"></p>
12 <p id = "convertString"></p>
```

CSS

1

JavaScript + No-Library (pure JS)

Tidy

```
1 var text = "SIMON";
2 document.getElementById("charAt").innerHTML = text.charAt(2);
3
4 let text1 = "SIMON";
5 document.getElementById("codeAt").innerHTML = text.charCodeAt(0);
6
7 var str = "XINYUJIANG";
8 document.getElementById("propertyAccess").innerHTML = str[1];
9
10 let text2 = "s, j, k, a";
11 const myArray = text2.split(",");
12 document.getElementById("convertString").innerHTML = myArray[1];
```

SIT120 Ass2 task

task1

M

83

l

a

## Task2

HTML

```
1 <div>
2 </div>
3
4 <p id = "toString"></p>
5 <p id = "join"></p>
6 <p id = "pop"></p>
7 <p id = "popl"></p>
8
9 <button onclick="simon()">Do it</button>
10 <p id="push"></p>
11 <p id="push1"></p>
12
13
14
15
16
17
18
19
20
21
```

CSS

1

JavaScript + No-Library (pure JS)

```
1 const countries = ["China", "UK", "US", "Australia"];
2
3 document.getElementById("toString").innerHTML = countries.toString();
4 document.getElementById("join").innerHTML = countries.join(" & ");
5
6 countries.pop();
7 document.getElementById("pop").innerHTML = countries.join("-");
8 document.getElementById("popl").innerHTML = countries.pop();
9
10 document.getElementById("push").innerHTML = countries;
11
12
13
14
15
16
17
18
19
20
21
```

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

```
1 <html>
2 <h1>
3 SIT120 Ass2 task
4 </h1>
5 <div>
6 task2
7 </div>
8 </html>
9 <p id="shift"></p>
10 <p id="unshift3"></p>
11 <p id="shift1"></p>
12 <p id="shift2"></p>
13 <p id="AM"></p>
14 <p id="AM1"></p>
```

CSS

1

JavaScript + No-Library (pure JS)

Tidy

```
1 const countries = ["China", "UK", "US", "Australia"];
2
3 document.getElementById("shift").innerHTML = countries;
4 countries.shift();
5
6 document.getElementById("unshift3").innerHTML = countries.unshift("India");
7 document.getElementById("shift1").innerHTML = countries.join(" * ");
8 document.getElementById("shift2").innerHTML = countries.shift();
9
10 document.getElementById("AM").innerHTML = countries;
11 countries[0] = "French";
12 document.getElementById("AM1").innerHTML = countries;
13
14
15
16
```

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



HTML ▼	
11 <p id="TE"></p><div>	
12 <div id="toFixed"></div>	2
13 <div id="toFixed"></div>	2.330000
14 <div id="toFixed"></div>	
15 <div id="toFixed"></div>	
16 <div id="toFixed"></div>	
17 <div id="toFixed"></div>	
18 <div id="toFixed"></div>	1.245
19 <div id="toFixed"></div>	1
20 <div id="toFixed"></div>	1.245
21 <div id="toFixed"></div>	
22 <div id="toFixed"></div>	
23 <div id="toFixed"></div>	
JavaScript + No Library (pure JS) ▼	
17	
18 let simon3 = 1.245;	
19 document.getElementById("toFixed").innerHTML =	1
20 simon3.toFixed(1) + " " +	0
21 simon3.toFixed(1) + " " +	5
22 simon3.toFixed(1);	5.11
23	NaN
24 let simon4 = 3.345;	NaN
25 document.getElementById("toFixed").innerHTML =	NaN
26 simon4.toFixed(1) + " " +	NaN
27 (3.345).toFixed(1) + " " ;	
28	
29 document.getElementById("toFixed").innerHTML =	
30 Number(true) + " " +	597456000000
31 Number(false) + " " +	
32 Number(" 5 ") + " " ;	

HTML ▼	
11 <p id="TE"></p><div>	
12 <div id="toFixed"></div>	2
13 <div id="toFixed"></div>	2.330000
14 <div id="toFixed"></div>	
15 <div id="toFixed"></div>	
16 <div id="toFixed"></div>	
17 <div id="toFixed"></div>	
18 <div id="toFixed"></div>	1.245
19 <div id="toFixed"></div>	1
20 <div id="toFixed"></div>	1.245
21 <div id="toFixed"></div>	
22 <div id="toFixed"></div>	
23 <div id="toFixed"></div>	
JavaScript + No Library (pure JS) ▼	
17	
18 Number("2.44") + " " +	1
19 Number("18 45") + " " +	0
20 Number("simon");	5
21	5.11
22 let simon5 = new Date("1988-12-07");	NaN
23 document.getElementById("toFixed").innerHTML =	NaN
24 document.getElementById("toFixed").innerHTML =	
25 parseInt("2") + " " +	
26 parseInt("2.3") + " " +	
27 parseInt("2 years 610") + " " +	
28 parseInt("years 1 and 2");	
29	
30 document.getElementById("toFixed").innerHTML =	
31 parseFloat("10") + " " +	
32 parseFloat("10.33") + " " +	
33 parseFloat(" ten miles of pizza ");	597456000000

HTML ▼	
11 <p id="TE"></p><div>	
12 <div id="toFixed"></div>	1
13 <div id="toFixed"></div>	0
14 <div id="toFixed"></div>	5
15 <div id="toFixed"></div>	5.11
16 <div id="toFixed"></div>	NaN
17 <div id="toFixed"></div>	NaN
18 <div id="toFixed"></div>	NaN
19 <div id="toFixed"></div>	
20 <div id="toFixed"></div>	
21 <div id="toFixed"></div>	
22 <div id="toFixed"></div>	
23 <div id="toFixed"></div>	
JavaScript + No Library (pure JS) ▼	
17	
18 let simon6 = Number.MAX_VALUE;	-2
19 document.getElementById("toFixed").innerHTML = simon6;	-2
20	NaN
21	
22 let simon7 = Number.MIN_VALUE;	10
23 document.getElementById("toFixed").innerHTML = simon7;	10.33
24	NaN
25 let simon8 = 37 / 0;	
26 document.getElementById("toFixed").innerHTML = simon8;	
27	
28 document.getElementById("toFixed").innerHTML = Number.NaN;	
29	
30 let simon9 = 23456789078;	1.7976931348623157e+308
31 document.getElementById("toFixed").innerHTML = simon9_MAX_VALUE;	

HTML ▼		597456000000
11	<div id="toFixed"></div>	
12	<div id="toFixed"></div>	-2
13	<div id="toFixed"></div>	-2
14	<div id="toFixed"></div>	2
15	<div id="toFixed"></div>	NaN
16	<div id="toFixed"></div>	NaN
17	<div id="toFixed"></div>	NaN
18	<div id="toFixed"></div>	NaN
19	<div id="toFixed"></div>	NaN
20	<div id="toFixed"></div>	NaN
21	<div id="toFixed"></div>	NaN
22	<div id="toFixed"></div>	NaN
23	<div id="toFixed"></div>	NaN
JavaScript + No Library (pure JS) ▼		
17		
18	let simon10 = 37 / 0;	
19	document.getElementById("toFixed").innerHTML = simon10;	1.7976931348623157e+308
20		
21	document.getElementById("toFixed").innerHTML = Number.NaN;	
22		
23	let simon11 = 23456789078;	5e-324
24	document.getElementById("toFixed").innerHTML = simon11_MAX_VALUE;	
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		
50		
51		
52		
53		
54		
55		
56		
57		
58		
59		
60		
61		
62		
63		
64		
65		
66		
67		
68		
69		
70		
71		
72		
73		
74		
75		
76		
77		
78		
79		
80		
81		
82		
83		
84		
85		
86		
87		
88		
89		
90		
91		
92		
93		
94		
95		
96		
97		
98		
99		
100		
101		
102		
103		
104		
105		
106		
107		
108		
109		
110		
111		
112		
113		
114		
115		
116		
117		
118		
119		
120		
121		
122		
123		
124		
125		
126		
127		
128		
129		
130		
131		
132		
133		
134		
135		
136		
137		
138		
139		
140		
141		
142		
143		
144		
145		
146		
147		
148		
149		
150		
151		
152		
153		
154		
155		
156		
157		
158		
159		
160		
161		
162		
163		
164		
165		
166		
167		
168		
169		
170		
171		
172		
173		
174		
175		
176		
177		
178		
179		
180		
181		
182		
183		
184		
185		
186		
187		
188		
189		
190		
191		
192		
193		
194		
195		
196		
197		
198		
199		
200		
201		
202		
203		
204		
205		
206		
207		
208		
209		
210		
211		
212		
213		
214		
215		
216		
217		
218		
219		
220		
221		
222		
223		
224		
225		
226		
227		
228		
229		
230		
231		
232		
233		
234		
235		
236		
237		
238		
239		
240		
241		
242		
243		
244		
245		
246		
247		
248		
249		
250		
251		
252		
253		
254		
255		
256		
257		
258		
259		
260		
261		
262		
263		
264		
265		
266		
267		
268		
269		
270		
271		
272		
273		
274		
275		
276		
277		
278		
279		
280		
281		
282		
283		
284		
285		
286		
287		
288		
289		
290		
291		
292		
293		
294		
295		
296		
297		
298		
299		
300		
301		
302		
303		
304		
305		
306		
307		
308		
309		
310		
311		
312		
313		
314		
315		
316		
317		
318		
319		
320		
321		
322		
323		
324		
325		
326		
327		
328		
329		
330		
331		
332		
333		
334		
335		
336		
337		
338		
339		
340		
341		
342		
343		
344		
345		
346		
347		
348		
349		
350		
351		
352		
353		
354		
355		
356		
357		
358		
359		
360		
361		
362		
363		
364		
365		
366		
367		
368		
369		
370		
371		
372		
373		
374		
375		
376		
377		
378		
379		
380		
381		
382		
383		
384		
385		
386		
387		
388		
389		
390		
391		
392		
393		
394		
395		
396		
397		
398		
399		
400		
401		
402		
403		
404		
405		
406		
407		
408		
409		
410		
411		
412		
413		
414		
415		
416		
417		
418		
419		
420		
421		
422		
423		
424		
425		
426		
427		
428		
429		
430		
431		
432		
433		
434		
435		
436		
437		
438		
439		
440		
441		
442		
443		
444		
445		
446		
447		
448		
449		
450		
451		
452		
453		
454		
455		
456		
457		
458		
459		
460		
461		
462		
463		
464		
465		
466		
467		
468		
469		
470		
471		
472		
473		
474		
475		
476		
477		
478		
479		
480		
481		
482		
483		
484		
485		
486		
487		
488		
489		
490		
491		
492		
493		
494		
495		
496		
497		
498		
499		
500		
501		
502		
503		
504		
505		
506		
507		
508		
509		
510		
511		
512		
513		
514		
515		
516		
517		
518		
519		
520		
521		
522		
523		
524		
525		
526		
527		
528		
529		
530		
531		
532		
533		
534		
535		
536		
537		
538		
539		
540		
541		
542		
543		
544		
545		
546		
547		
548		
549		
550		
551		
552		
553		
554		
555		
556		
557		
558		
559		
560		
561		
562		
563		
564		
565		
566		
567		
568		
569		
570		
571		
572		
573		
574		
575		
576		
577		
578		
579		
580		
581		
582		
583		
584		
585		
586		
587		
588		
589		
590		
591		
592		
593		
594		
595		
596		
597		
598		
599		
600		
601		
602		
603		
604		
605		
606		
607		
608		
609		

## Task3

```
HTML
<h3>
  SIT120 Week3
</h3>

CSS

JavaScript - No Library (pure JS)
var d = new Date();
d.setTime(1414444000000);
d.setHours(11);
document.write(d);
```

**SIT120 Week3**

Thu Aug 07 2014 11:44:40 GMT+0800 (中国标准时间)

```
HTML
<h3>
  SIT120 Week3
</h3>
<div id="year"></div>

CSS

JavaScript - No Library (pure JS)
var d = new Date();
d.setFullYear(2077);
var a = document.getElementById("year");
a.innerHTML = d.getFullYear();
```

**SIT120 Week3**

2077

## 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. Class 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.