

COMPONENT VUE 3X



# III NỘI DUNG CHÍNH

I

Component

II

## 1. Component là gì

- Component là một trong những tính năng quan trọng nhất trong Vue. Nó giúp chúng ta có thể kế thừa các phần tử HTML, có thể tái sử dụng code, giúp code chúng ta nhìn ngắn gọn, sạch sẽ hơn. Những đoạn code chúng ta có thể khai báo trong một component sẽ là HTML, CSS hay là cả Javascript, chúng được gói gọn vào trong một component rồi sau đó chúng ta có thể gọi tới component và tái sử dụng chúng.

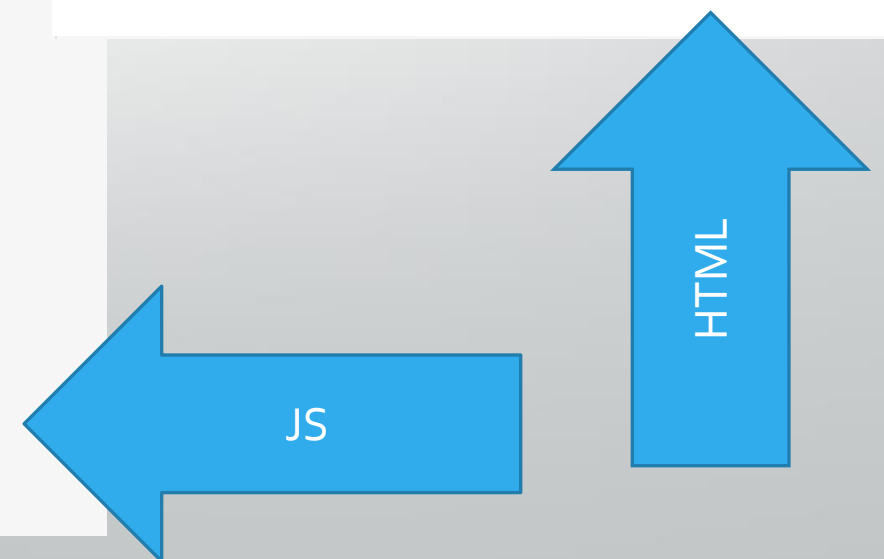
## 2. Khai báo component

- **Cách 1:** cách này thường được gọi là **string template**. Các đoạn code html sẽ được khai báo trong **template: ' ... '**

```
// Create a Vue application
const app = Vue.createApp({})

// Define a new global component called button-counter
app.component('button-counter', {
  data() {
    return {
      count: 0
    }
  },
  template: `
    <button @click="count++">
      You clicked me {{ count }} times.
    </button>`
})
```

```
<div id="components-demo">
  <button-counter></button-counter>
</div>
```



## 2. Khai báo component

- Có thể tái sử dụng component bằng cách khai báo nhiều **button-counter**

```
<div id="components-demo">  
  <button-counter></button-counter>  
  <button-counter></button-counter>  
  <button-counter></button-counter>  
</div>
```

Kết quả

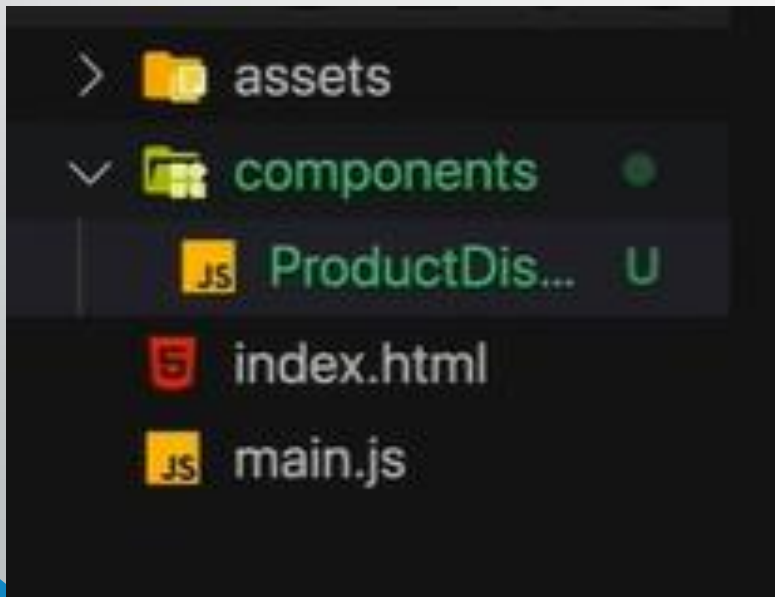
You clicked me 0 times. You clicked me 0 times. You clicked me 0 times.

## 2. Khai báo component

- **Cách 2:** Đây là cách phổ biến nhất mà nhiều người áp dụng hơn đó là single-file component. Khi chúng ta có một hệ thống lớn kết hợp nhiều files với nhau thì mỗi component sẽ được tách một file **.vue**, điều này sẽ giúp chúng ta dễ dàng kiểm soát code hơn cũng như là bảo trì code.
- Cách đặt tên cho các files component nên được đặt dưới dạng PascalCase (in hoa mỗi chữ cái đầu) hoặc là kebab-case (chữ thường toàn bộ và có có gạch nối - giữa các từ).
- Ví dụ
  - ✓ **DefaultAvatar.vue**
  - or
  - ✓ **default-avatar.vue**

## 2. Khai báo component

- Cú pháp để tạo 1 thành phần:
- *`App.component('tên component', {})`*
- Ví dụ:
  - Tạo component tên **ProductDisplay.js** trong thư mục tên **components**



## 2. Khai báo component

### ➤ Tempalte: Components\ProductDisplay.js

```
app.component('product-display', {
  template:
    `/*html*/
    <div class="product-display">
      <div class="product-container">
        <div class="product-image">
          
        </div>
        <div class="product-info">
          <h1>{{ title }}</h1>

          <p v-if="inStock">In Stock</p>
          <p v-else>Out of Stock</p>

          <div
            v-for="(variant, index) in variants"
            :key="variant.id"
            @mouseover="updateVariant(index)"
            class="color-circle"
            :style="{ backgroundColor: variant.color }">
          </div>

          <button
            class="button"
            :class="{ disabledButton: !inStock }"
            :disabled="!inStock"
            v-on:click="addToCart">
            Add to Cart
          </button>
        </div>
      </div>
    </div>`
})
```



## 2. Khai báo component

- **Data và Methods:** Trong thành phần **ProductDisplay.js** ta cung cấp cho nó dữ liệu và phương thức nhưng vẫn chạy được trên **main.js**

```
app.component('product-display', {
  template:
    /*html*/
    `<div class="product-display">
      ...
    </div>`,
  data() {
    return {
      product: 'Socks',
      brand: 'Vue Mastery',
      selectedVariant: 0,
      details: ['50% cotton', '30% wool', '20% polyester'],
      variants: [
        { id: 2234, color: 'green', image: './assets/images/socks_green.jpg', quantity: 50 },
        { id: 2235, color: 'blue', image: './assets/images/socks_blue.jpg', quantity: 0 },
      ]
    }
  },
  methods: {
    addToCart() {
      this.cart += 1
    },
    updateVariant(index) {
      this.selectedVariant = index
    }
  },
  computed: {
    title() {
      return this.brand + ' ' + this.product
    },
    image() {
      return this.variants[this.selectedVariant].image
    },
    inStock() {
      return this.variants[this.selectedVariant].quantity
    }
  }
})
```

## 2. Khai báo component

➤ main.js

```
const app = Vue.createApp({  
  data() {  
    return {  
      cart: 0  
    }  
  },  
  methods: {}  
})
```

## 2. Khai báo component

- Trong class index.html import vào component vừa tạo

```
<!-- Import App -->
<script src="./main.js"></script>

<!-- Import Components -->
<script src="./components/ProductDisplay.js"></script>

<!-- Mount App -->
<script>
  const mountedApp = app.mount('#app')
</script>
```

03:00

## 2. Khai báo component

### ➤ Index.html

```
<div id="app">  
  <div class="nav-bar"></div>  
  
  <div class="cart">Cart({{ cart }})</div>  
  <product-display></product-display>  
</div>
```



- Nếu muốn tái sử dụng lại thì gọi lại nhiều component

```
<div id="app">  
  <div class="nav-bar"></div>  
  
  <div class="cart">Cart({{ cart }})</div>  
  <product-display></product-display>  
  <product-display></product-display>  
  <product-display></product-display>  
</div>
```

### 3. Vùng hoạt động của Component

- **Global Component:** đối với global component thì chúng ta có thể sử dụng ở bất cứ đâu trong một Vue instance nào đó

```
const app = Vue.createApp({})

app.component('component-a', {
  /* ... */
})
app.component('component-b', {
  /* ... */
})
app.component('component-c', {
  /* ... */
})

app.mount('#app')
```

```
<div id="app">
  <component-a></component-a>
  <component-b></component-b>
  <component-c></component-c>
</div>
```

### 3. Vùng hoạt động của Component

- **Lobal Component:** nếu không muốn component sử dụng trong tất cả các Vue instance mà chỉ sử dụng ở trong một Vue instance nào đó thì chúng ta sử dụng Lobal component
- Khác với global component, nó sẽ không đăng ký trực tiếp component bằng cách khai báo `app.component` nữa, thay vào đó chúng ta định nghĩa một component như là một object

```
const ComponentA = {  
  /* ... */  
}  
const ComponentB = {  
  /* ... */  
}  
const ComponentC = {  
  /* ... */  
}
```

### 3. Vùng hoạt động của Component

- Sau đó định nghĩa 1 component mà muốn sử dụng

```
const app = Vue.createApp({  
  components: {  
    'component-a': ComponentA,  
    'component-b': ComponentB  
  }  
})
```

- **Lưu ý:** các thành phần đăng ký là global sẽ không có sẵn trong thành phần con. Ví dụ nếu muốn component A có sẵn trong component B thì phải đăng ký như sau:

```
const ComponentA = {  
  /* ... */  
}  
  
const ComponentB = {  
  components: {  
    'component-a': ComponentA  
  }  
  // ...  
}
```

không đăng ký sử dụng component đó thì chúng ta sẽ không nhận được kết quả

### 3. Vùng hoạt động của Component

- **Lobal cho một hệ thống Module:** Nếu chúng ta sử dụng hệ thống module như webpack, Babel, thì nên tạo một thư mục tên **component** để chứa tất cả các file component
- Sau đó import mỗi component vào trước khi đăng ký cục bộ chúng
- Ví dụ: Có component sử dụng 2 ComponentA và Component C thì

```
import ComponentA from './ComponentA'  
import ComponentC from './ComponentC'
```

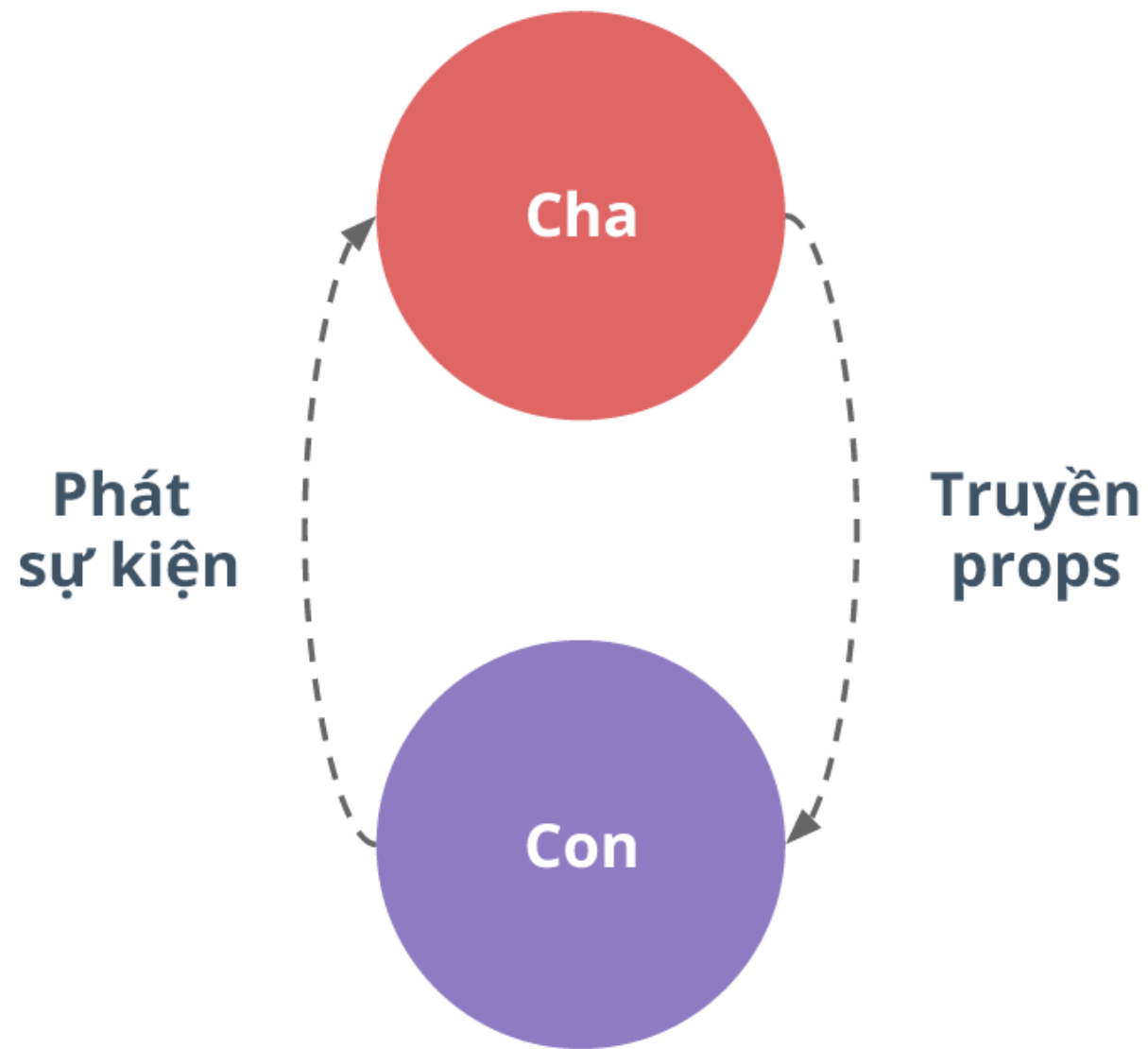
```
export default {  
  components: {  
    ComponentA,  
    ComponentC  
  }  
  // ...  
}
```

Import

Đăng ký lobal



## 4. Giao tiếp giữa các component



## 4. Giao tiếp giữa các component

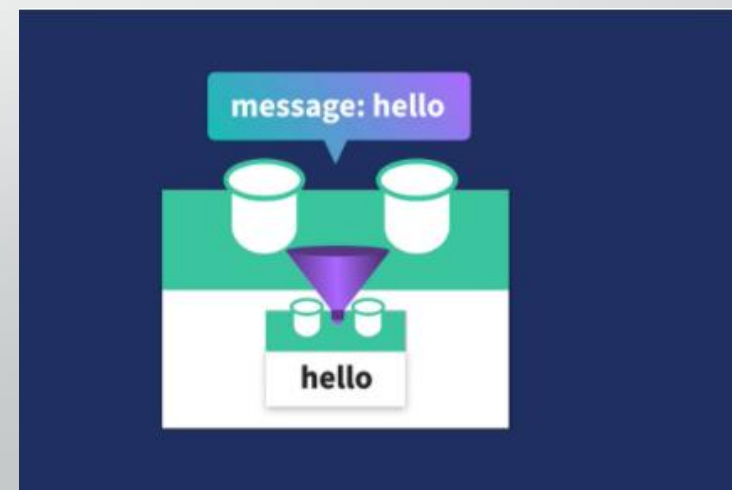
### Quan hệ Component cha - con:

- Mỗi component instance đều có một scope riêng của nó, nghĩa là mình không thể và cũng không nên trực tiếp gọi tới parent data trong child component template. Data có thể gửi xuống từ component cha thông qua một custom attribute là **props** (**Những data truyền từ component cha vào component con**)
- Mỗi component không được tự động thay đổi dữ liệu riêng cho mình, muốn thay đổi thì phải thông báo đến parent và nhiệm vụ của parent thực hiện đổi dữ liệu thông qua thuộc tính là **events up**

## 4. Giao tiếp giữa các component

### Props:

- Dùng để Component cha truyền dữ liệu xuống Component con
- Đây là các thuộc tính tùy chỉnh để truyền dữ liệu vào một thành phần. Chúng hoạt động giống như một cái phễu, nơi bạn có thể truyền dữ liệu mà thành phần cần.
- Cú pháp giống với thuộc tính, thuộc tính này do mình đặt ra, nên phải ràng buộc thuộc tính bằng **V-bind**



## 4. Giao tiếp giữa các component

### Props:

- Props nhận giá trị thì nó được hiểu như 1 object bao gồm có name và kiểu dữ liệu

```
props: {  
  title: String,  
  likes: Number,  
  isPublished: Boolean,  
  commentIds: Array,  
  author: Object,  
  callback: Function,  
  contactsPromise: Promise // or any other constructor  
}
```

## 4. Giao tiếp giữa các component

### Ví dụ 1:

Trong lớp main.js có một thuộc tính dữ liệu mới tên là **premium**, nếu là premium thì được Free vận chuyển. Vì vậy, **class product-display** cần quyền truy cập vào thuộc tính này, nên nó cần cha truyền dữ liệu cho con thông qua thuộc tính **props**

## 4. Giao tiếp giữa các component

## Ví dụ 1

Nhận về như 1 thuộc tính  
**premium**

```
main.js > [0] app > [0] data > [0] premium
1 const app = Vue.createApp({
2   data() {
3     return {
4       cart: 0,
5       premium: true
6     }
7   },
8 })
9
10 // Import the component
11 import ProductDisplay from './components/ProductDisplay.js'
12
13 // Register the component
14 app.component('product-display', ProductDisplay)
15
16 // Mount the app
17 app.mount('#app')
```

## 4. Giao tiếp giữa các component

### Ví dụ 1

 index.html

```
<div id="app">
  <div class="nav-bar"></div>

  <div class="cart">Cart({{ cart }})</div>
  <product-display :premium="premium"></product-display>
</div>
```

## 4. Giao tiếp giữa các component

### Ví dụ 1

```
main.js
main.js > [app] > [data] > premium
1  const app = Vue.createApp({
2    data() {
3      return {
4        cart: 0,
5        premium: true
6      }
7    },
8  },
9  )
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
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
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Tên thuộc tính đặt trùng tên với **premium**



## 4. Giao tiếp giữa các component

### Ví dụ

component / ProductDisplay.js

```
template:
  /*html*/
  `

component / ProductDisplay.js



```
computed: {
  ...
  shipping() {
    if (this.premium) {
      return 'Free'
    }
    return 2.99
  }
}
```


```

## 4. Giao tiếp giữa các component

### Ví dụ 2: Trang App.vue

```
<template>
  <div id="app">
    
    <comp-header v-bind:titleheader="title"/>
    <CompListProduct/>
    <comp-footer/>
  </div>
</template>
<script>
import CompHeader from './components/CompHeader.vue';
import CompFooter from './components/CompFooter';
import CompListProduct from './components/CompListProduct.vue';
export default {
  name: 'app',
  data () {
    return {
      title: 'Welcome to Your Vue.js App'
    }
  },

```

Tên thuộc tính có thể đặt tên trùng với tên **title**

## 4. Giao tiếp giữa các component

### Ví dụ 2: CompHeader.vue

```
<template>
  <header><h1>{{titleheader}}</h1>
</template>
<script>
export default {
  name: 'comp-header',
  props: {
    titleheader: String
  },
  data() {
    return {
      text: 'Hello Vuejs Header'
    }
  }
}
</script>
```

Nhận về như  
thuộc tính: kiểu  
dữ liệu

Nếu như cha không  
truyền gì cả thì thuộc  
tính dưới dạng object  
như sau:

```
props: {
  titleheader: {
    type: String,
    default: 'Khi cha không truyền gì cả'
  }
},
```

## 4. Giao tiếp giữa các component

### Truyền dữ liệu từ component con lên component cha:

- **Event up:** Dùng để truyền thông điệp hay truyền sự kiện thông báo cho components cha biết là nó muốn thay đổi dữ liệu. Nhiệm vụ của component cha sau khi nhận được thông điệp sẽ tiến hành thay đổi data

- **Cú pháp:**

```
<my-component v-on:my-event="doSomething"></my-component>
```

Tên sự kiện người dùng tự đặt và được kích hoạt thông qua những đoạn code

Hàm xử lý

## 4. Giao tiếp giữa các component

**Truyền dữ liệu từ component con lên component cha:**

- **Event up:** Để kích hoạt được sự kiện dùng từ khóa
- ***`this.$emit('myEvent')`, được gọi trong components con***
- **Lưu ý:**
  - ✓ Không được phép thay đổi trực tiếp props từ component cha
  - ✓ Truyền sự kiện thông báo ra bên ngoài
  - ✓ Kích hoạt sự kiện trong component con
  - ✓ Khi kích hoạt function bên ngoài component cha nó sẽ chạy
  - ✓ Khi function đó chạy nó sẽ được thay đổi dữ liệu

## 4. Giao tiếp giữa các component

**Ví dụ:** Chúng ta đã biết rằng props là một cách để truyền dữ liệu xuống một thành phần (cha truyền con), nhưng còn khi có điều gì đó xảy ra bên trong thành phần đó (thành phần con), chẳng hạn như một lần nhấp vào nút? Làm cách nào để chúng tôi thông báo cho các phần khác của ứng dụng biết rằng sự kiện đó đã xảy ra?



## 4. Giao tiếp giữa các component

The diagram illustrates the communication between a parent component and a child component in a Vue.js application. It is divided into two main sections: the parent component's HTML template and the child component's JavaScript logic.

**Parent Component (index.html):**

- The HTML template contains the `<product-display :premium="premium" @add-to-cart="updateCart">` tag. The `updateCart` attribute is highlighted with a red box.
- A red circle labeled `'add-to-cart'` represents the event being emitted by the child component.

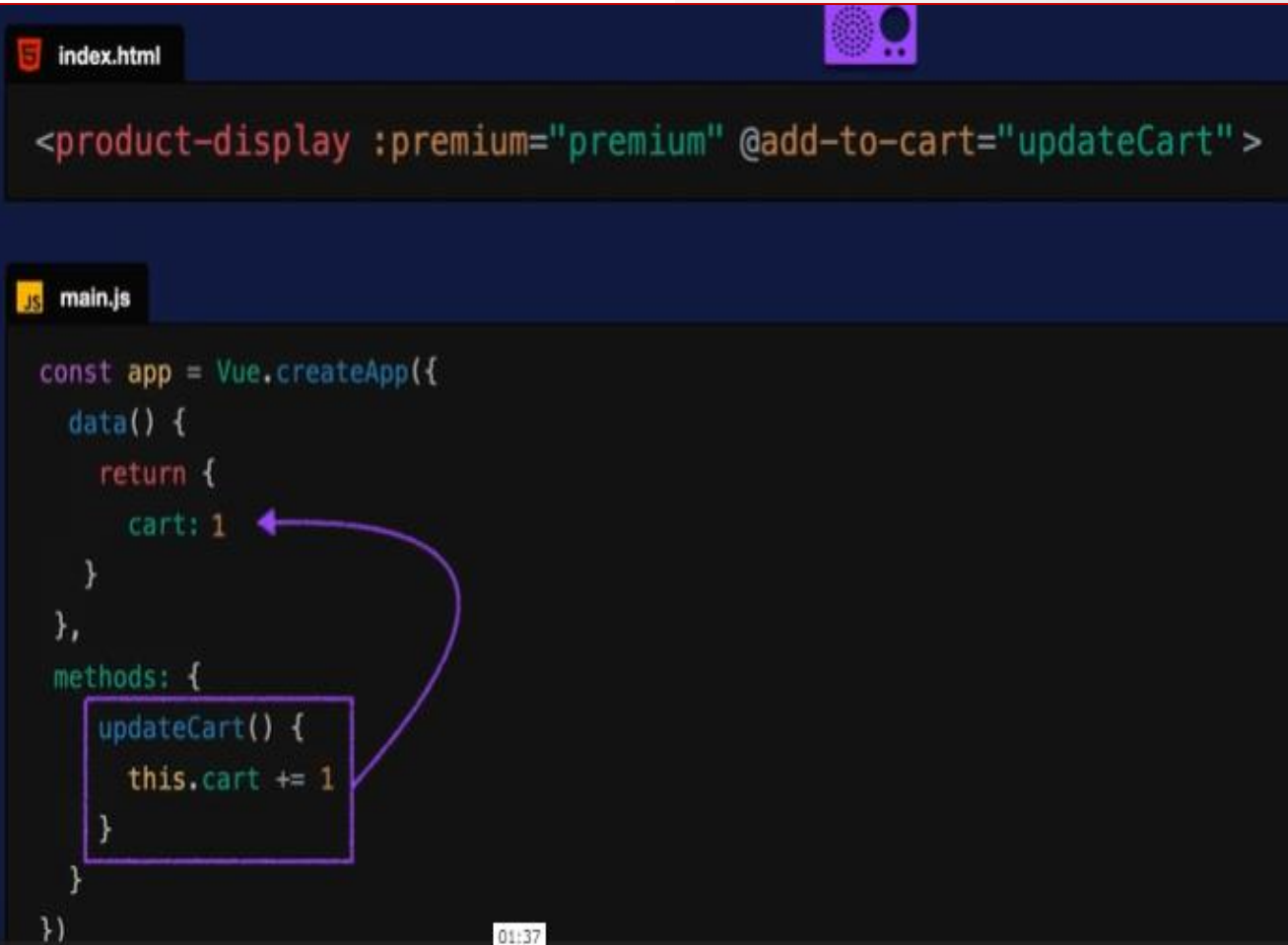
**Child Component (ProductDisplay.js):**

- The JavaScript code defines the `product-display` component with a `methods` object containing an `addToCart()` function.
- Inside `addToCart()`, the code `this.$emit('add-to-cart')` is shown, which triggers the event.
- A red radio tower icon and a red button labeled "Add to cart" are shown next to the `addToCart()` function, indicating the user action that triggers the event.

## 4. Giao tiếp giữa các component

 main.js

```
const app = Vue.createApp({
  data() {
    return {
      cart: [],
      ...
    }
  },
  methods: {
    updateCart() {
      this.cart += 1
    }
  }
})
```



```
index.html
<product-display :premium="premium" @add-to-cart="updateCart">

main.js
const app = Vue.createApp({
  data() {
    return {
      cart: 1
    }
  },
  methods: {
    updateCart() {
      this.cart += 1
    }
  }
})
```

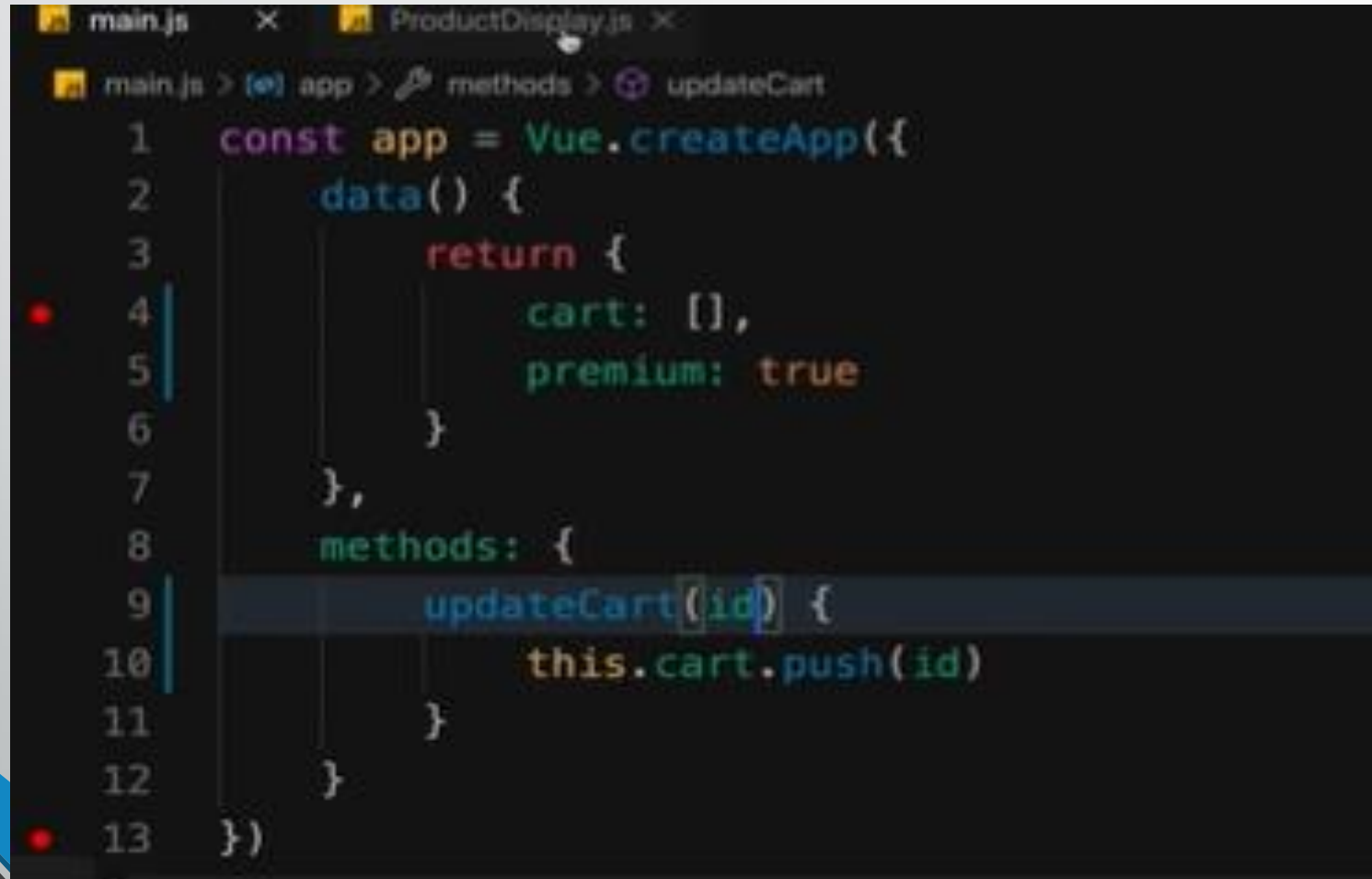
01:37



## 4. Giao tiếp giữa các component

**Ví dụ:** Truyền vào 1 id sản phẩm

**Main.js**

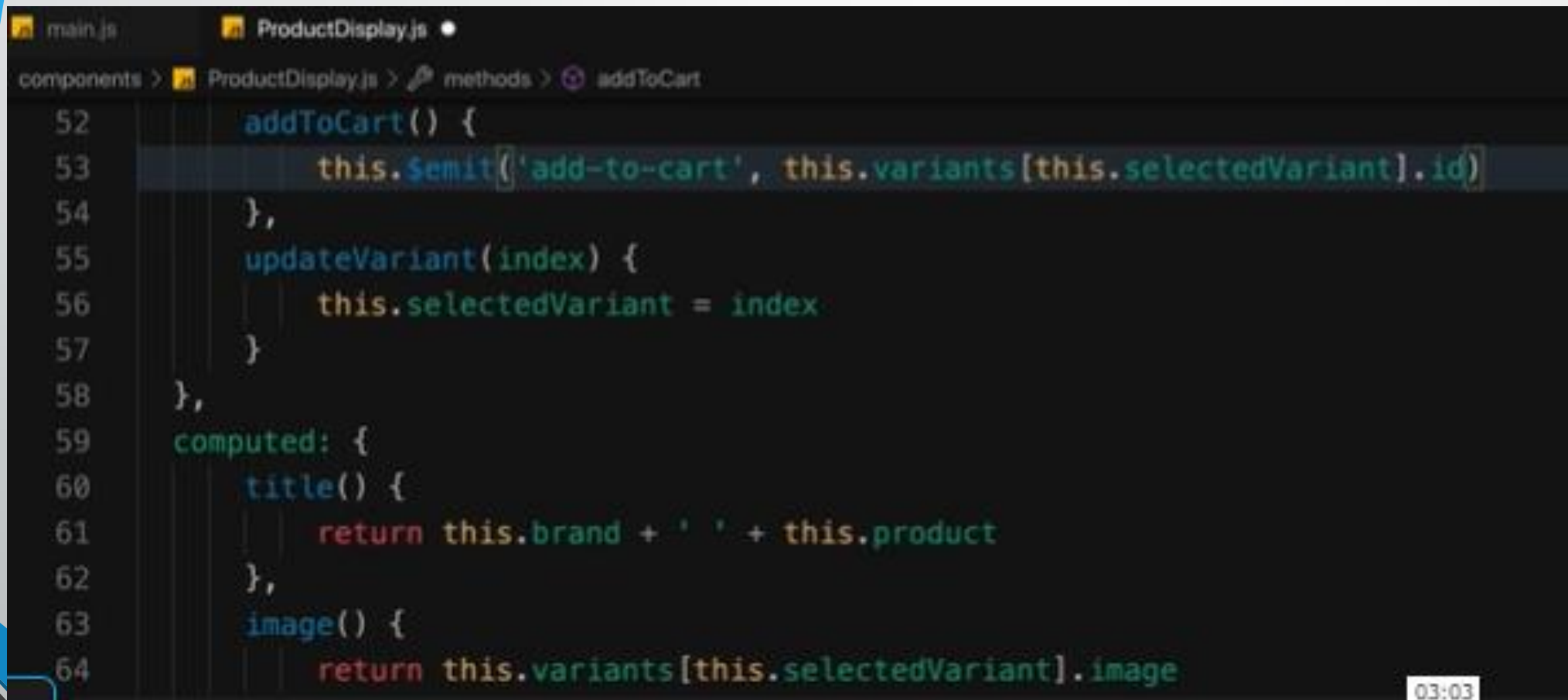


```
main.js  x  ProductDisplay.js  x
main.js > app > methods > updateCart
1  const app = Vue.createApp({
2    data() {
3      return {
4        cart: [],
5        premium: true
6      }
7    },
8    methods: {
9      updateCart(id) {
10        this.cart.push(id)
11      }
12    }
13  })
```

## 4. Giao tiếp giữa các component

**Ví dụ:** Truyền vào 1 id sản phẩm

**ProductDisplay.js**



```
52  addToCart() {  
53    this.$emit('add-to-cart', this.variants[this.selectedVariant].id)  
54  },  
55  updateVariant(index) {  
56    this.selectedVariant = index  
57  },  
58  },  
59  computed: {  
60    title() {  
61      return this.brand + ' ' + this.product  
62    },  
63    image() {  
64      return this.variants[this.selectedVariant].image
```

03:03

## 4. Giao tiếp giữa các component

Ví dụ: Truyền vào 1 id sản phẩm

Index.html

```
<body>
  <div id="app">
    <div class="nav-bar"></div>

    <div class="cart">Cart({{ cart.length }})</div>
    <product-display :premium="premium" @add-to-cart="updateCart"></product-display>
  </div>
```



### Vue Mastery Socks

In Stock

Shipping: 2.99

50% cotton  
30% wool  
20% polyester



Add to Cart

Cart(2)



The  
end