

23

Week 43

THURSDAY
Day (298-069)

| OCTOBER | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|
| S | M | T | W | T | F | S | S | M | T | W | T |
| | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 |
| OCT | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | | |

Appointments

09 Episode - 11 Data is New Oil

10 \Rightarrow Higher Order Components :- Higher Order Components that takes a component and returns a component that wraps original. It takes component and enhances it and returns a component.

13 Suppose we have to promote some restaurant so we will give our card to higher order component & then give it promoted label & return a component.

15 Syntax :-

```
16 const EnhancedComponent = (component needs to be enhanced)
17     => { return () => {
18         return jsx of enhanced component
19     }
20 }
```

export const withPromotedLabel = (RestaurantCard) => {
 return () => {
 return (
 <div>
 <label> Promoted </label>
 <RestaurantCard />
 </div>
);
 };
}

OCTOBER

| S | S | M | T | W | F | S |
|----|----|----|----|----|----|----|
| 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| 08 | 09 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | | | | | |

FRIDAY

Week 43

Day (297)

2

Appointments

So, Higher Order component takes a component as input and returns a component (function) that receives a piece of jsx with it.

11

Use it like this

const RestaurantAndPromoted = withPromoteLabel(Restaurant)

{/Pass props}

13

isPromoted? (<RestaurantAndPromoted

resData = {resData} />)

14

o (<RestaurantAndPromoted

resData = {resData} />)

15

~~Notes~~ // Receiving Props

16

with

export default withPromotedLabel = (RestaurantAndPromoted)

return (props) => {

return (

<div>

<label> Promoted </label>

<RestaurantAndPromoted ...props

</div>

) ;

y;

y;

$w - 6/12$ is half width of page in tailwind

25

SATURDAY

Week 43

Day (298-067)

OCTOBER

| S | M | T | W | T | F | S | S | M | T | W | F | S |
|---|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
| T | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| | 25 | 26 | 27 | 28 | 29 | 30 | 31 | | | | | |

Appointments We are not changing the Restaurants card component we are returning the same component with some modifications.

10

All react applications have 2 layers

11

- 1 Data Layer
- 2 UI Layer

UI Layer is powered by Data Layer, means props, state, local variables and JavaScript

Code all include in Data Layer. So, UI Layer is contain JSX code.

15

The @type is not a valid JS object or variable

so to use this in JS we do

c. card?.card?.["@type"] = something

17

⇒ React Dev Tools

26 SUNDAY

React Developers Tools is good for react developers it can show you all the components & their hierarchy. It can also show means almost the virtual DOM. This is very - very useful for debugging.

⇒ Now we want to build a feature that if we open an accordion then all other accordions will be closed automatically.

2025

Restaurant Menu (Parent)

OCTOBER

| S | M | T | W | T | F | S |
|----|----|----|----|----|----|----|
| | 01 | 02 | 03 | 04 | 05 | 06 |
| 07 | 08 | 09 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | | | | |

RestaurantCategory (Child)

27

Appointments

- For this, we have to give power to
- show the category list to the parent of RestaurantCategory.
- Because right now every category has its own state variable instead
- Restaurant Menu will control state of every category.

12

Controlled & Uncontrolled Components

- Here now Restaurant Menu is controlling their children (RestaurantCategory), so it is called Restaurant controlled component and when RestaurantCategory has its own state (showlist) then it was uncontrolled component.
- component in the component tree.

16

Lifting state up → It means passing the state from lower level (child) component to a higher level (parent or common ancestor).

To achieve we will use the state variable and pass both variable & state function in children.

NOVEMBER

DECEMBER

RestaurantMenu.js

```
const [showIndex, setShowIndex] = useState(null);
```

< RestaurantCategory

```
& showIndex { false }
```

```
setShowIndex = {} => setShowIndex(23) />
```

28

TUESDAY

Week 44

Day (301-064)

OCTOBER

| S | O | M | T | W | T | F | S | S | M | T | W | F | S |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | S |
| T | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | | | | | | | |

Appointments

Restaurant Category.js

09

Cont RestaurantCategory = ({ showIndex, setShowIndex }) => {

10

<div>

11

const handleclick = () => {
 setShowIndex();
}

12

> return <div onClick={handleClick}>

13

showIndex & (<

14

<div>

15

Item List

16

</div>

17

</div>

)};

Here what RestaurantMenu is doing is
creating a setShowIndex function for
every category

For index: 0

setShowIndex = () => setShowIndex(0);

For index 1

setShowIndex = () => setShowIndex(1);
and so on.

OCTOBER

| S | M | T | W | T | F | S | S |
|----|----|----|----|----|----|----|----|
| | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| 0 | 08 | 09 | 10 | 11 | 12 | 13 | 14 |
| V | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 25 | 26 | 27 | 28 | 29 | 30 | | |

29

WEDNESDAY

Week 44

Day (302-063)

Appointments

Child doesn't care about index. Closure remembers the index.

(\Rightarrow) setShowIndex(index)

Javascript remembers Index in closure

So, when child calls, Javascript run the setShowIndex for the specific function

Suppose for index: 2

Local:

setShowIndex f()

Closure (RestaurantMenu.js)

index: 2

setShowIndex f()

When only one accordion opens

Because in the parent

Showlist = { ShowIndex = =Index }

Only one index can open at a time.

Stick another category to update showIndex.

So, child is controlled by parent as it cannot open itself it only open when parent says showlist == false.

One-line :-

The parent already knows the index while rendering and passes a closure bound handler.

Sometimes you want the state of two components always change together. To do it remove the state from both of them, move it to their closest common parent then pass it down to them via props. This is known as lifting state up. This is also called single source of truth.

NOVEMBER

DECEMBER

30

Week 44

THURSDAY

Day (303-062)

| O | M | T | W | T | F | S | S | M | T | W | T | F | S |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C | | | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 |
| T | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| | 25 | 27 | 28 | 29 | 30 | 31 | | | | | | | 26 |

Appointments

- 09 To make the category close by clicking again
 10 we will do
 (1) ~~setShowIndex~~
 11 SetShowIndex = { C } \Rightarrow setShowIndex (ShowIndex == index ? null : index)
 12 So, if ShowIndex has index value already
 13 then make it null else give it
 index.
 14

~~#~~ props drilling

- 16 React has one-way data flow
 we have pass the data from parent
 to children to their children & so on.

Restaurant Menu (it has ~~children~~
 ↓
 category)

Restaurant Category (child of Restaurant
 ↓
 items)

Items List (Child of Restaurant Category)

But what if we had hundreds of components
 and if I want to pass the data from
 Restaurant menu to Item list

OCTOBER

| S | M | T | W | F | S | S |
|----|----|----|----|----|----|----|
| | 01 | 02 | 03 | 04 | 05 | 06 |
| 0 | 07 | 08 | 09 | | | |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |

V 17 18 19 20 21 22 23

25 26 27 28 29 30

31

FRIDAY

Week 44

Day (304-061)

Appointments then I have to make it go through all those components which is not using this and it's not a good way and this concept is called props drilling.

To overcome this we use React context so we can use some global component everywhere which can be accessed everywhere without using props. It is like a central store which keeps data.

Let's create a userContent to store information of logged in user.

userContent.js

```
import { createContent } from "react"
const userContent = createContent ({
```

```
    loggedInUser : "Default User",
```

```
});
```

Using it (we will use Content() hook to use it)

```
const loggedInUser = useContent(userContent);
```

We cannot use hooks in class based components. So, to access it we do.

NOVEMBER

DECEMBER

01

SATURDAY

Week 44

Day (305/1060)

import userContent from 'userContent.js'

Appointments

09

< UserContent.Consumer >

{ (data) => console.log(data) },

10

{ loggedInUser } => < h1 > { loggedInUser }

11

< UserContent.Consumer / >

12

How to change userContent

13

Suppose there's an api which gives us userInfo and give the data and change the content.

14

15 We will use Content Provider to update the value of the userInfo.

16

const appLayout = () => {

17

const [userName, setUsername] = useState(" ")
useEffect(() => {

2 SUNDAY

const data = [

name: "Akshay Saini",

setUsername(data.name);

], []);

return (

< UserContent.Provider value={ { loggedInUser } } >

2025

< div > </ div >

> < / UserContent.Provider >

Action Plan

Schedule your plan For Better future.

November
2025

| Wk | 44 | 45 | 46 | 47 | 48 |
|----|----|----|----|----|----|
|----|----|----|----|----|----|

| | | | | | |
|-----|---|--|--|--|--|
| MON | < UserContent.Provider value={ loggedInUser: userName }> | | | | |
| TUE | < APP > < UserContent.Provider value={ loggedInUser: "Elon Musk" }> | | | | |
| WED | < Header /> < UserContent.Provider /> < APP /> | | | | |
| THU | < UserContent.Provider /> So, Header will have Elon Musk as loggedInUser and other component in App will have Be Passionate | | | | |
| FRI | How to modify userContent value Follow your dynamically? We have to pass the setUserName function to change it dynamically | | | | |
| SAT | < UserContent.Provider value={ loggedInUser: success will follow userName, setUserName }> | | | | |
| SUN | < / UserContent.Provider > | | | | |

NOVEMBER

DECEMBER

03

Week 45

MONDAY
Day (307-058)

Appointments

| NOVEMBER | | | | | | |
|----------|----|----|----|----|----|----|
| S | M | T | W | T | F | S |
| O | | | | 01 | 02 | 03 |
| V | 10 | 11 | 12 | 13 | 14 | 15 |
| 25 | 24 | 25 | 26 | 27 | 28 | 29 |
| | | | | 16 | 17 | 18 |
| | | | | 19 | 20 | 21 |
| | | | | 22 | 23 | 08 |
| | | | | 09 | 10 | 00 |

09 Content Provider :- The content provider is a component that allows its children to subscribe to a content's changes. It accepts a value prop, which is the data that will be shared with the components that are the descendants of the provider. The Provider component is created using `React.createContent()` and then rendered as part of the component tree. It establishes the content and provides the data to the descendants.

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
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
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
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
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
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1