



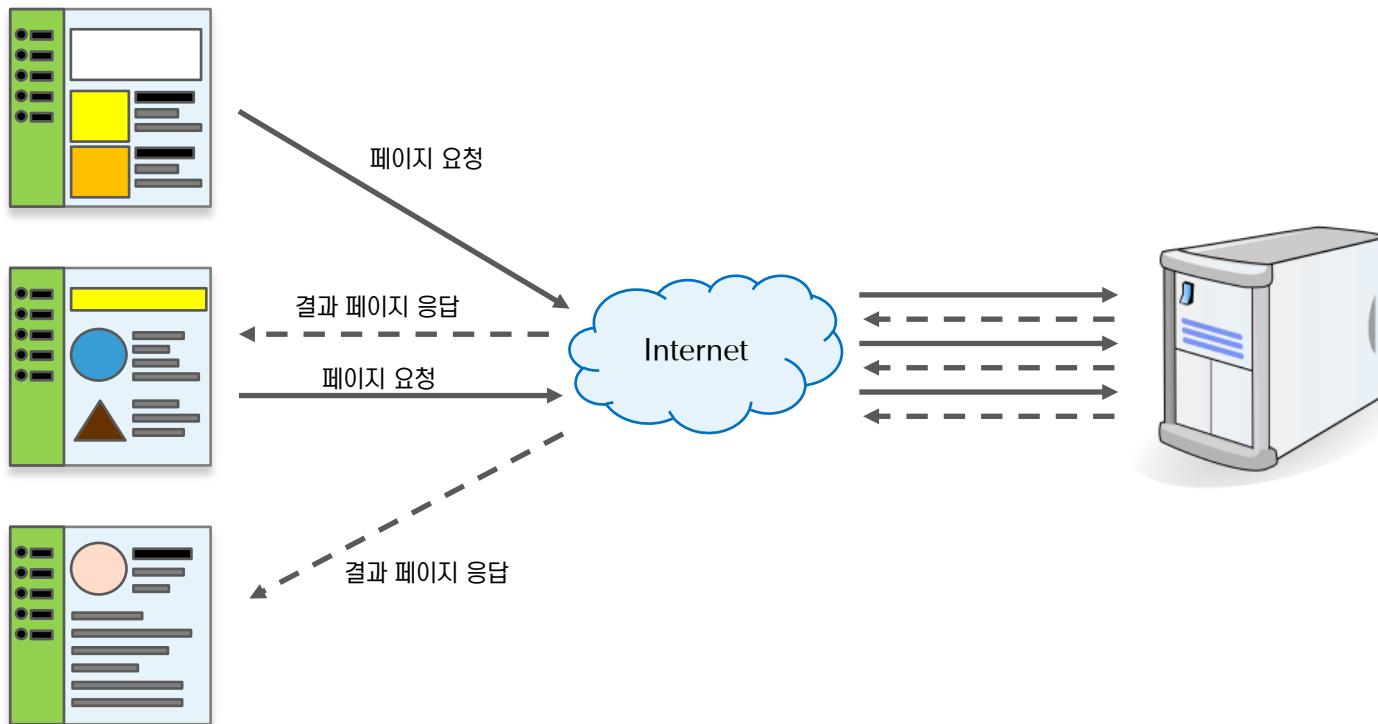
# 1. Intro to React

---

- 1.1 MPA vs SPA
- 1.2 React 소개
- 1.3 React Application 개발 구성 요소
- 1.4 개발 환경 구성

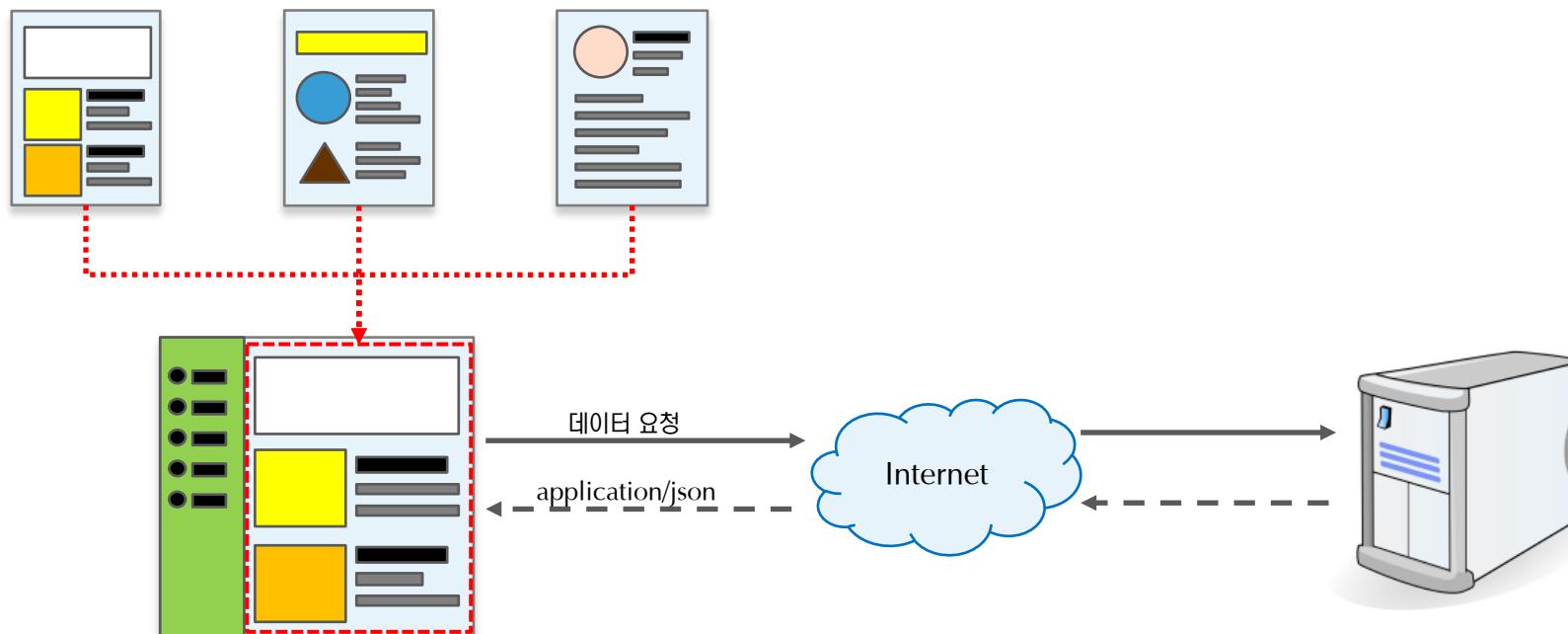
## 1.1 MPA vs SPA[1/2]

- ✓ MPA(Multi-Page Application)는 클라이언트의 요청에 따라 서버에서 페이지를 생성하고 이를 반환하는 형태입니다.
- ✓ 클라이언트의 입장에서는 서버에 요청하는 모든 페이지가 존재하기 때문에 사용자 요구에 따라 요청만 진행합니다.
- ✓ MPA 방식에서 서버는 모든 클라이언트들의 요청에 각각 대응하며 모든 페이지를 생성하고 응답해야 합니다.
- ✓ MPA 방식은 페이지의 작은 변화에도 서버에 전체 페이지를 요청하고 화면을 갱신합니다.



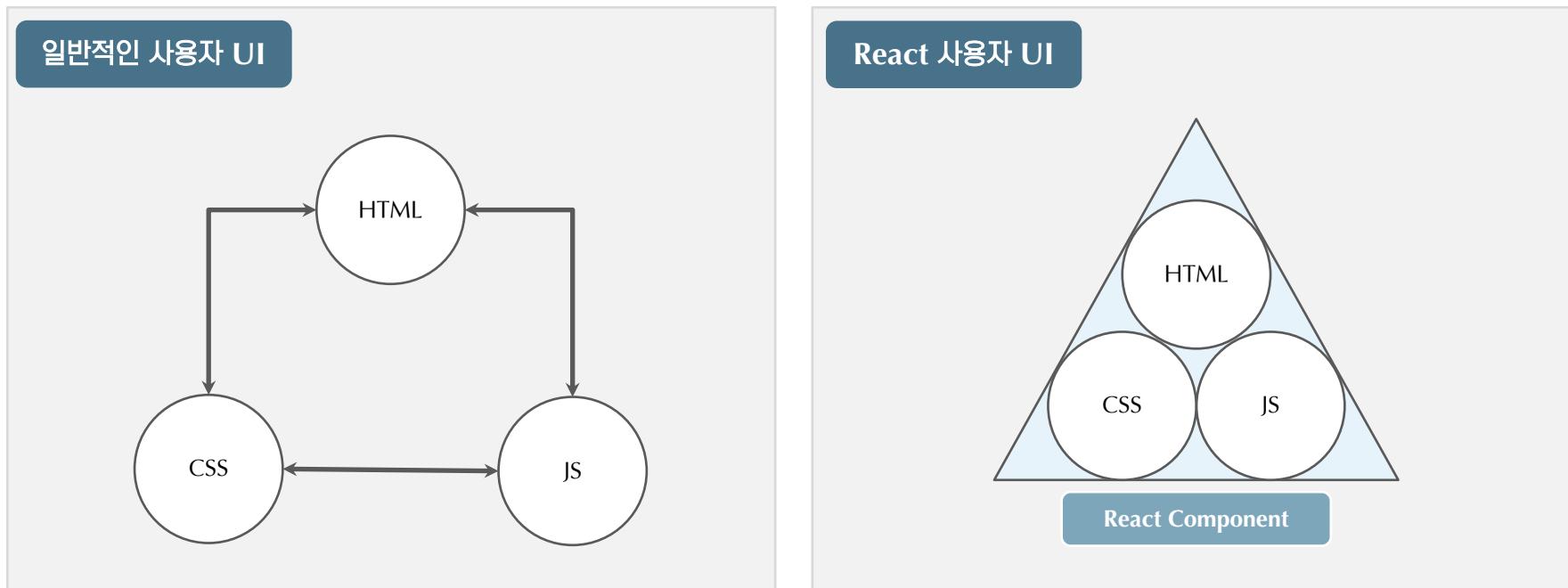
## 1.1 MPA vs SPA[2/2]

- ✓ SPA(Single-Page Application)는 서버로 부터 단일 페이지를 응답받고 클라이언트에서 화면을 구성하는 방식입니다.
- ✓ 클라이언트는 서버에 데이터(JSON)를 요청하고 응답 받은 데이터를 이용해 화면을 갱신합니다.
- ✓ 서버의 입장에서 다수의 클라이언트로부터 요청이 발생해도 페이지를 생성하지 않고 요청 데이터만 반환하기 때문에 부하가 적습니다.



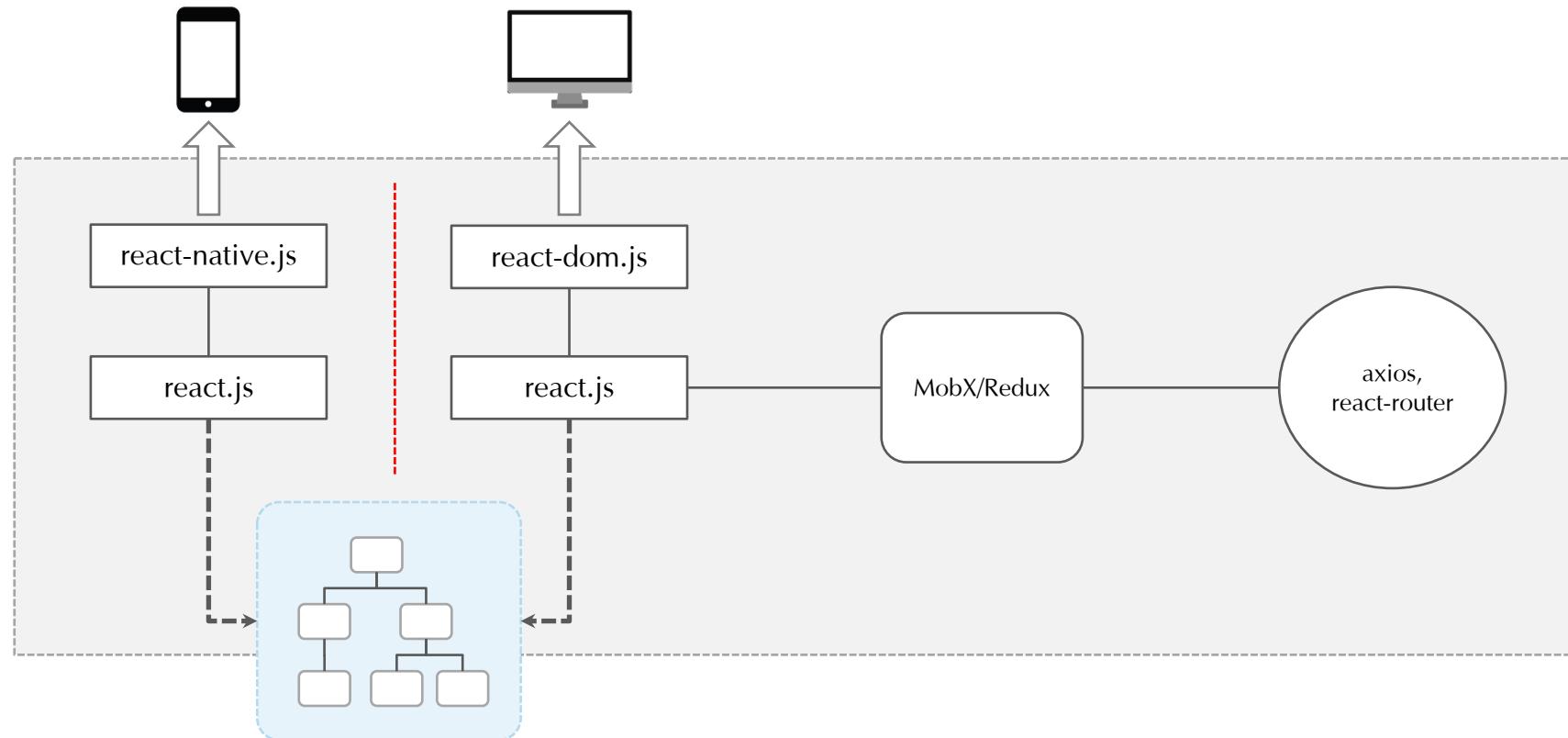
## 1.2 React 소개

- ✓ React는 SPA 기반 웹 사용자 UI(User Interface) 구성을 위한 JavaScript 라이브러리(Library)입니다.
- ✓ React는 독립적이며 재사용이 가능한 UI 컴포넌트를 손쉬운 방법으로 생성할 수 있도록 합니다.
- ✓ 컴포넌트를 만든다는 것은 기존에 구분해서 관리하던 HTML, CSS, JS를 하나의 요소로 묶는 것을 의미합니다.
- ✓ React UI를 구성하는 컴포넌트를 만드는 것의 핵심은 구성하고자 하는 화면을 어떻게 컴포넌트로 분리하느냐 입니다.



## 1.3 React Application 개발 구성 요소

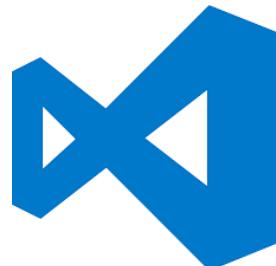
- ✓ React 어플리케이션을 구성하는 기본 단위는 캡슐화 된 컴포넌트 입니다.
- ✓ React 컴포넌트를 정의하고 활용하기 위해 사용하는 라이브러리는 react.js 코어 라이브러리 입니다.
- ✓ react-dom.js, react-native.js 라이브러리는 UI 요소들을 화면에 렌더링하는 라이브러로 react.js와 함께 사용합니다.
- ✓ React는 UI 요소들을 화면에 표현하는 것에만 집중한 라이브러리 입니다. 따라서, 다양한 종류의 Third-party 라이브러리와 함께 어플리케이션을 구성합니다.



## 1.4 개발 환경 구축 (1/5)

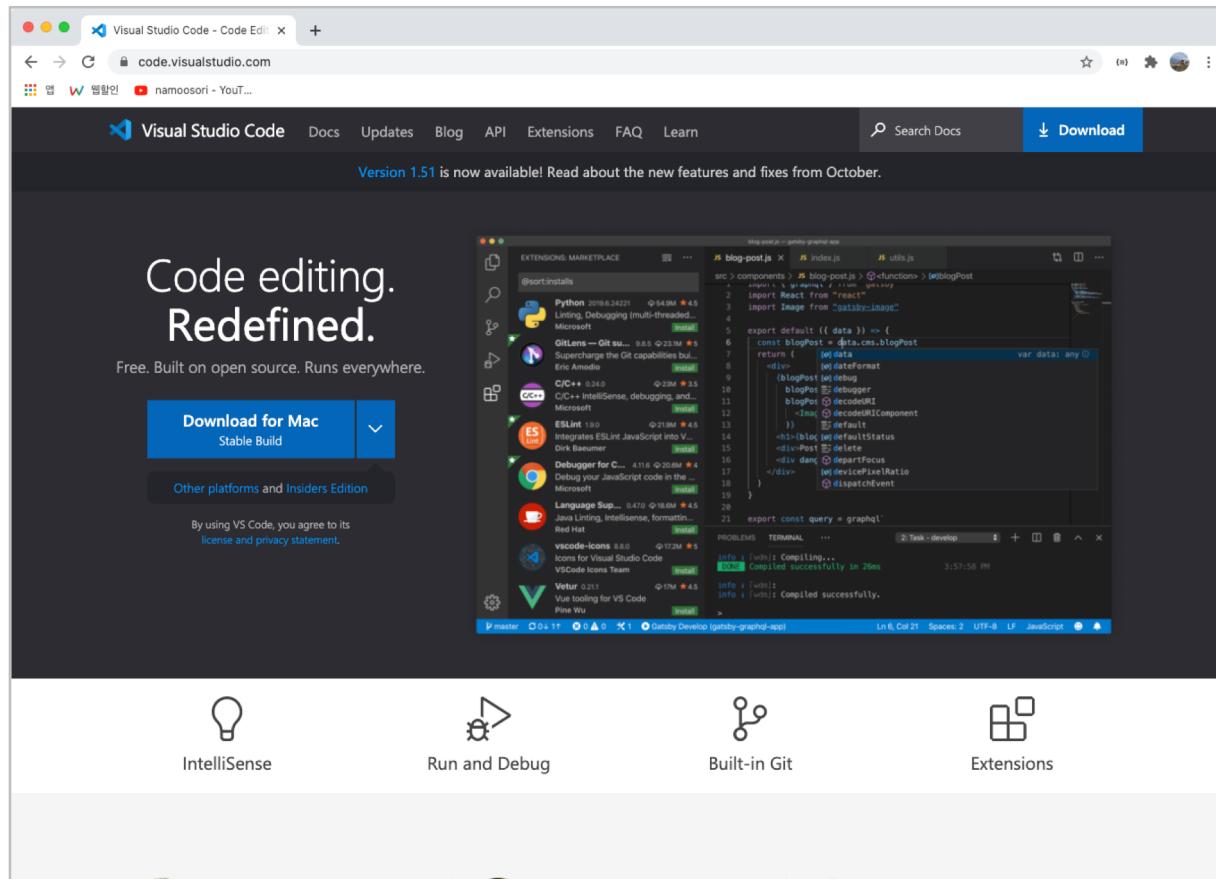
✓ React 개발에는 다음의 환경이 필요합니다.

- Code Editor: 실제 코드를 작성하기 위한 편집 도구입니다. 대표적인 도구로는 Atom, Bracket, VS code(Visual Studio Code) 등이 있습니다.
- node.js & npm: node.js는 JavaScript가 동작 할 수 있는 실행 환경입니다. npm(node package manager)은 다양한 JavaScript 패키지를 관리하기 위한 도구이며 패키지는 node.js에서 사용 가능한 모듈들의 집합을 뜻합니다.
- Web Browser: React를 이용한 UI 개발에 적합한 일반적인 웹브라우저는 일반적으로 Google의 Chrome을 사용합니다.
- git: 특정 패키지나 모듈의 경우 git 설치를 필요로 합니다.



# 1.4 개발 환경 구축 (2/5)

- ✓ Visual Studio Code는 MS에서 개발한 코드 편집 도구입니다.
- ✓ 다음의 URL을 통해 해당 사이트에서 개발 컴퓨터에 적합한 버전을 다운로드 합니다.
  - <https://code.visualstudio.com>



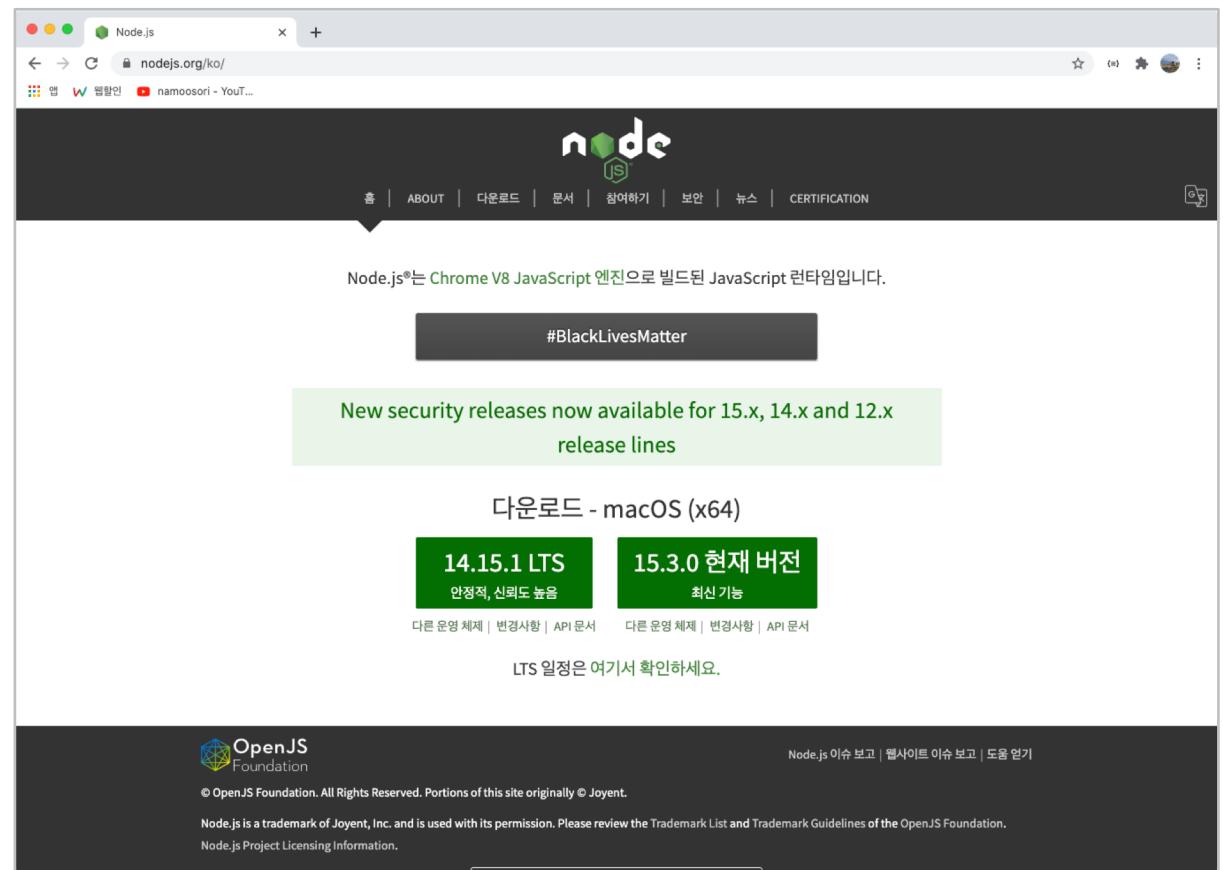
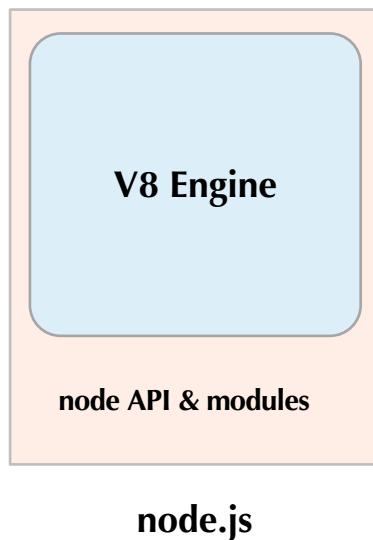
The screenshot shows the Visual Studio Code interface running on a Windows system. The title bar indicates it's 'App.js - react\_workspace - Visual Studio Code'. The interface includes the Explorer sidebar on the left showing project files like 'App.js', 'index.js', and 'EmployeeList.js'. The main area is the code editor with the following code:

```
JS App.js
emp_exam > src > JS App.js > ...
1 import React, { Component } from 'react';
2 import { Grid, Segment } from 'semantic-ui-react';
3
4 import SearchBar from './components/SearchBar';
5 import EmployeeList from './components/EmployeeList';
6 import EmployeeDetail from './components/EmployeeDetail';
7
8 import Employees from './components/Employees';
9
10 class App extends Component {
11
12   constructor(props) {
13     super(props);
14
15     this.state = {
16       employees: Employees,
17       selectedEmployee: Employees[0]
18     }
19
20   }
21
22   searchByName(name) {
23     let updateList = Employees;
24     updateList = updateList.filter(employee => {
25       return employee.name.toLowerCase().search(name) !== -1;
26     });
27     this.setState(
28       { employees: updateList }
29     )
30
31   }
32
33   render() {
34     return (
35       <div className='App'>
36         <Segment>
37           <SearchBar onSearchByName={this.searchByName.bind(this)} />
38         </Segment>
39         <Grid columns={2} stackable>
40           <Grid.Column>
41             <Segment>
42               <EmployeeList
43                 onEmployeeSelect={selectedEmployee => this.setState({ selectedEmp
44                 employees:(this.state.employees) />
45               </Segment>
46             </Grid.Column>
47             <Grid.Column>
48               <EmployeeDetail employee={this.state.selectedEmployee} />
49             </Grid.Column>
50           </Grid>
51         </div>
52       )
53     )
54   }
55
56   componentDidMount() {
57     this.searchByName();
58   }
59
60   componentDidUpdate() {
61     this.searchByName();
62   }
63
64   componentWillUnmount() {
65     this.searchByName();
66   }
67
68   render() {
69     return (
70       <div>
71         <h1>Employee Management</h1>
72         <p>A simple application to manage employees. You can search by name, add new employees, and view details of an employee.</p>
73         <hr/>
74         <table border="1">
75           <thead>
76             <tr>
77               <th>Employee ID</th>
78               <th>Name</th>
79             </tr>
80           </thead>
81           <tbody>
82             <tr>
83               <td>1</td>
84               <td>John Doe</td>
85             </tr>
86             <tr>
87               <td>2</td>
88               <td>Jane Smith</td>
89             </tr>
90             <tr>
91               <td>3</td>
92               <td>Mike Johnson</td>
93             </tr>
94             <tr>
95               <td>4</td>
96               <td>Sarah Lee</td>
97             </tr>
98           </tbody>
99         </table>
100        <hr/>
101        <h2>Employee Details</h2>
102        <table border="1">
103          <thead>
104            <tr>
105              <th>Employee ID</th>
106              <th>Name</th>
107              <th>Position</th>
108              <th>Salary</th>
109            </tr>
110          </thead>
111          <tbody>
112            <tr>
113              <td>1</td>
114              <td>John Doe</td>
115              <td>Software Engineer</td>
116              <td>$60,000</td>
117            </tr>
118            <tr>
119              <td>2</td>
120              <td>Jane Smith</td>
121              <td>Project Manager</td>
122              <td>$70,000</td>
123            </tr>
124            <tr>
125              <td>3</td>
126              <td>Mike Johnson</td>
127              <td>Data Analyst</td>
128              <td>$55,000</td>
129            </tr>
130            <tr>
131              <td>4</td>
132              <td>Sarah Lee</td>
133              <td>Marketing Specialist</td>
134              <td>$65,000</td>
135            </tr>
136          </tbody>
137        </table>
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
199  )
200
200  )
201
201  )
202
202  )
203
203  )
204
204  )
205
205  )
206
206  )
207
207  )
208
208  )
209
209  )
210
210  )
211
211  )
212
212  )
213
213  )
214
214  )
215
215  )
216
216  )
217
217  )
218
218  )
219
219  )
220
220  )
221
221  )
222
222  )
223
223  )
224
224  )
225
225  )
226
226  )
227
227  )
228
228  )
229
229  )
230
230  )
231
231  )
232
232  )
233
233  )
234
234  )
235
235  )
236
236  )
237
237  )
238
238  )
239
239  )
240
240  )
241
241  )
242
242  )
243
243  )
244
244  )
245
245  )
246
246  )
247
247  )
248
248  )
249
249  )
250
250  )
251
251  )
252
252  )
253
253  )
254
254  )
255
255  )
256
256  )
257
257  )
258
258  )
259
259  )
260
260  )
261
261  )
262
262  )
263
263  )
264
264  )
265
265  )
266
266  )
267
267  )
268
268  )
269
269  )
270
270  )
271
271  )
272
272  )
273
273  )
274
274  )
275
275  )
276
276  )
277
277  )
278
278  )
279
279  )
280
280  )
281
281  )
282
282  )
283
283  )
284
284  )
285
285  )
286
286  )
287
287  )
288
288  )
289
289  )
290
290  )
291
291  )
292
292  )
293
293  )
294
294  )
295
295  )
296
296  )
297
297  )
298
298  )
299
299  )
300
300  )
301
301  )
302
302  )
303
303  )
304
304  )
305
305  )
306
306  )
307
307  )
308
308  )
309
309  )
310
310  )
311
311  )
312
312  )
313
313  )
314
314  )
315
315  )
316
316  )
317
317  )
318
318  )
319
319  )
320
320  )
321
321  )
322
322  )
323
323  )
324
324  )
325
325  )
326
326  )
327
327  )
328
328  )
329
329  )
330
330  )
331
331  )
332
332  )
333
333  )
334
334  )
335
335  )
336
336  )
337
337  )
338
338  )
339
339  )
340
340  )
341
341  )
342
342  )
343
343  )
344
344  )
345
345  )
346
346  )
347
347  )
348
348  )
349
349  )
350
350  )
351
351  )
352
352  )
353
353  )
354
354  )
355
355  )
356
356  )
357
357  )
358
358  )
359
359  )
360
360  )
361
361  )
362
362  )
363
363  )
364
364  )
365
365  )
366
366  )
367
367  )
368
368  )
369
369  )
370
370  )
371
371  )
372
372  )
373
373  )
374
374  )
375
375  )
376
376  )
377
377  )
378
378  )
379
379  )
380
380  )
381
381  )
382
382  )
383
383  )
384
384  )
385
385  )
386
386  )
387
387  )
388
388  )
389
389  )
390
390  )
391
391  )
392
392  )
393
393  )
394
394  )
395
395  )
396
396  )
397
397  )
398
398  )
399
399  )
400
400  )
401
401  )
402
402  )
403
403  )
404
404  )
405
405  )
406
406  )
407
407  )
408
408  )
409
409  )
410
410  )
411
411  )
412
412  )
413
413  )
414
414  )
415
415  )
416
416  )
417
417  )
418
418  )
419
419  )
420
420  )
421
421  )
422
422  )
423
423  )
424
424  )
425
425  )
426
426  )
427
427  )
428
428  )
429
429  )
430
430  )
431
431  )
432
432  )
433
433  )
434
434  )
435
435  )
436
436  )
437
437  )
438
438  )
439
439  )
440
440  )
441
441  )
442
442  )
443
443  )
444
444  )
445
445  )
446
446  )
447
447  )
448
448  )
449
449  )
450
450  )
451
451  )
452
452  )
453
453  )
454
454  )
455
455  )
456
456  )
457
457  )
458
458  )
459
459  )
460
460  )
461
461  )
462
462  )
463
463  )
464
464  )
465
465  )
466
466  )
467
467  )
468
468  )
469
469  )
470
470  )
471
471  )
472
472  )
473
473  )
474
474  )
475
475  )
476
476  )
477
477  )
478
478  )
479
479  )
480
480  )
481
481  )
482
482  )
483
483  )
484
484  )
485
485  )
486
486  )
487
487  )
488
488  )
489
489  )
490
490  )
491
491  )
492
492  )
493
493  )
494
494  )
495
495  )
496
496  )
497
497  )
498
498  )
499
499  )
500
500  )
501
501  )
502
502  )
503
503  )
504
504  )
505
505  )
506
506  )
507
507  )
508
508  )
509
509  )
510
510  )
511
511  )
512
512  )
513
513  )
514
514  )
515
515  )
516
516  )
517
517  )
518
518  )
519
519  )
520
520  )
521
521  )
522
522  )
523
523  )
524
524  )
525
525  )
526
526  )
527
527  )
528
528  )
529
529  )
530
530  )
531
531  )
532
532  )
533
533  )
534
534  )
535
535  )
536
536  )
537
537  )
538
538  )
539
539  )
540
540  )
541
541  )
542
542  )
543
543  )
544
544  )
545
545  )
546
546  )
547
547  )
548
548  )
549
549  )
550
550  )
551
551  )
552
552  )
553
553  )
554
554  )
555
555  )
556
556  )
557
557  )
558
558  )
559
559  )
560
560  )
561
561  )
562
562  )
563
563  )
564
564  )
565
565  )
566
566  )
567
567  )
568
568  )
569
569  )
570
570  )
571
571  )
572
572  )
573
573  )
574
574  )
575
575  )
576
576  )
577
577  )
578
578  )
579
579  )
580
580  )
581
581  )
582
582  )
583
583  )
584
584  )
585
585  )
586
586  )
587
587  )
588
588  )
589
589  )
590
590  )
591
591  )
592
592  )
593
593  )
594
594  )
595
595  )
596
596  )
597
597  )
598
598  )
599
599  )
600
600  )
601
601  )
602
602  )
603
603  )
604
604  )
605
605  )
606
606  )
607
607  )
608
608  )
609
609  )
610
610  )
611
611  )
612
612  )
613
613  )
614
614  )
615
615  )
616
616  )
617
617  )
618
618  )
619
619  )
620
620  )
621
621  )
622
622  )
623
623  )
624
624  )
625
625  )
626
626  )
627
627  )
628
628  )
629
629  )
630
630  )
631
631  )
632
632  )
633
633  )
634
634  )
635
635  )
636
636  )
637
637  )
638
638  )
639
639  )
640
640  )
641
641  )
642
642  )
643
643  )
644
644  )
645
645  )
646
646  )
647
647  )
648
648  )
649
649  )
650
650  )
651
651  )
652
652  )
653
653  )
654
654  )
655
655  )
656
656  )
657
657  )
658
658  )
659
659  )
660
660  )
661
661  )
662
662  )
663
663  )
664
664  )
665
665  )
666
666  )
667
667  )
668
668  )
669
669  )
670
670  )
671
671  )
672
672  )
673
673  )
674
674  )
675
675  )
676
676  )
677
677  )
678
678  )
679
679  )
680
680  )
681
681  )
682
682  )
683
683  )
684
684  )
685
685  )
686
686  )
687
687  )
688
688  )
689
689  )
690
690  )
691
691  )
692
692  )
693
693  )
694
694  )
695
695  )
696
696  )
697
697  )
698
698  )
699
699  )
700
700  )
701
701  )
702
702  )
703
703  )
704
704  )
705
705  )
706
706  )
707
707  )
708
708  )
709
709  )
710
710  )
711
711  )
712
712  )
713
713  )
714
714  )
715
715  )
716
716  )
717
717  )
718
718  )
719
719  )
720
720  )
721
721  )
722
722  )
723
723  )
724
724  )
725
725  )
726
726  )
727
727  )
728
728  )
729
729  )
730
730  )
731
731  )
732
732  )
733
733  )
734
734  )
735
735  )
736
736  )
737
737  )
738
738  )
739
739  )
740
740  )
741
741  )
742
742  )
743
743  )
744
744  )
745
745  )
746
746  )
747
747  )
748
748  )
749
749  )
750
750  )
751
751  )
752
752  )
753
753  )
754
754  )
755
755  )
756
756  )
757
757  )
758
758  )
759
759  )
760
760  )
761
761  )
762
762  )
763
763  )
764
764  )
765
765  )
766
766  )
767
767  )
768
768  )
769
769  )
770
770  )
771
771  )
772
772  )
773
773  )
774
774  )
775
775  )
776
776  )
777
777  )
778
778  )
779
779  )
780
780  )
781
781  )
782
782  )
783
783  )
784
784  )
785
785  )
786
786  )
787
787  )
788
788  )
789
789  )
790
790  )
791
791  )
792
792  )
793
793  )
794
794  )
795
795  )
796
796  )
797
797  )
798
798  )
799
799  )
800
800  )
801
801  )
802
802  )
803
803  )
804
804  )
805
805  )
806
806  )
807
807  )
808
808  )
809
809  )
810
810  )
811
811  )
812
812  )
813
813  )
814
814  )
815
815  )
816
816  )
817
817  )
818
818  )
819
819  )
820
820  )
821
821  )
822
822  )
823
823  )
824
824  )
825
825  )
826
826  )
827
827  )
828
828  )
829
829  )
830
830  )
831
831  )
832
832  )
833
833  )
834
834  )
835
835  )
836
836  )
837
837  )
838
838  )
839
839  )
840
840  )
841
841  )
842
842  )
843
843  )
844
844  )
845
845  )
846
846  )
847
847  )
848
848  )
849
849  )
850
850  )
851
851  )
852
852  )
853
853  )
854
854  )
855
855  )
856
856  )
857
857  )
858
858  )
859
859  )
860
860  )
861
861  )
862
862  )
863
863  )
864
864  )
865
865  )
866
866  )
867
867  )
868
868  )
869
869  )
870
870  )
871
871  )
872
872  )
873
873  )
874
874  )
875
875  )
876
876  )
877
877  )
878
878  )
879
879  )
880
880  )
881
881  )
882
882  )
883
883  )
884
884  )
885
885  )
886
886  )
887
887  )
888
888  )
889
889  )
890
890  )
891
891  )
892
892  )
893
893  )
894
894  )
895
895  )
896
896  )
897
897  )
898
898  )
899
899  )
900
900  )
901
901  )
902
902  )
903
903  )
904
904  )
905
905  )
906
906  )
907
907  )
908
908  )
909
909  )
910
910  )
911
911  )
912
912  )
913
913  )
914
914  )
915
915  )
916
916  )
917
917  )
918
918  )
919
919  )
920
920  )
921
921  )
922
922  )
923
923  )
924
924  )
925
925  )
926
926  )
927
927  )
928
928  )
929
929  )
930
930  )
931
931  )
932
932  )
933
933  )
934
934  )
935
935  )
936
936  )
937
937  )
938
938  )
939
939  )
940
940  )
941
941  )
942
942  )
943
943  )
944
944  )
945
945  )
946
946  )
947
947  )
948
948  )
949
949  )
950
950  )
951
951  )
952
952  )
953
953  )
954
954  )
955
955  )
956
956  )
957
957  )
958
958  )
959
959  )
960
960  )
961
961  )
962
962  )
963
963  )
964
964  )
965
965  )
966
966  )
967
967  )
968
968  )
969
969  )
970
970  )
971
971  )
972
972  )
973
973  )
974
974  )
975
975  )
976
976  )
977
977  )
978
978  )
979
979  )
980
980  )
981
981  )
982
982  )
983
983  )
984
984  )
985
985  )
986
986  )
987
987  )
988
988  )
989
989  )
990
990  )
991
991  )
992
992  )
993
993  )
994
994  )
995
995  )
996
996  )
997
997  )
998
998  )
999
999  )
1000
1000  )
1001
1001  )
1002
1002  )
1003
1003  )
1004
1004  )
1005
1005  )
1006
1006  )
1007
1007  )
1008
1008  )
1009
1009  )
1010
1010  )
1011
1011  )
1012
1012  )
1013
1013  )
1014
1014  )
1015
1015  )
1016
1016  )
1017
1017  )
1018
1018  )
1019
1019  )
1020
1020  )
1021
1021  )
1022
1022  )
1023
1023  )
1024
1024  )
1025
1025  )
1026
1026  )
1027
1027  )
1028
1028  )
1029
1029  )
1030
1030  )
1031
1031  )
1032
1032  )
1033
1033  )
1034
1034  )
1035
1035  )
1036
1036  )
1037
1037  )
1038
1038  )
1039
1039  )
1040
1040  )
1041
1041  )
1042
1042  )
1043
1043  )
1044
1044  )
1045
1045  )
1046
1046  )
1047
1047  )
1048
1048  )
1049
1049  )
1050
1050  )
1051
1051  )
1052
1052  )
1053
1053  )
1054
1054  )
1055
1055  )
1056
1056  )
1057
1057  )
1058
1058  )
1059
1059  )
1060
1060  )
1061
1061  )
1062
1062  )
1063
1063  )
1064
1064  )
1065
1065  )
1066
1066  )
1067
1067  )
1068
1068  )
1069
1069  )
1070
1070  )
1071
1071  )
1072
1072  )
1073
1073  )
1074
1074  )
1075
1075  )
1076
1076  )
1077
1077  )
1078
1078  )
1079
1079  )
1080
1080  )
1081
1081  )
1082
1082  )
1083
1083  )
1084
1084  )
1085
1085  )
1086
1086  )
1087
1087  )
1088
1088  )
1089
1089  )
1090
1090  )
1091
1091  )
1092
1092  )
1093
1093  )
1094
1094  )
1095
1095  )
1096
1096  )
1097
1097  )
1098
1098  )
1099
1099  )
1100
1100  )
1101
1101  )
1102
1102  )
1103
1103  )
1104
1104  )
1105
1105  )
1106
1106  )
1107
1107  )
1108
1108  )
1109
1109  )
1110
1110  )
1111
1111  )
1112
1112  )
1113
1113  )
1114
1114  )
1115
1115  )
1116
1116  )
1117
1117  )
1118
1118  )
1119
1119  )
1120
1120  )
1121
1121  )
1122
1122  )
1123
1123  )
1124
1124  )
1125
1125  )
1126
1126  )
1127
1127  )
1128
1128  )
1129
1129  )
1130
1130  )
1131
1131  )
1132
1132  )
1133
1133  )
1134
1134  )
1135
1135  )
1136
1136  )
1137
1137  )
1138
1138  )
1139
1139  )
1140
1140  )
1141
1141  )
1142
1142  )
1143
1143  )
1144
1144  )
1145
1145  )
1146
1146  )
1147
1147  )
1148
1148  )
1149
1149  )
1150
1150  )
1151
1151  )
1152
1152  )
1153
1153  )
1154
1154  )
1155
1155  )
1156
1156  )
1157
1157  )
1158
1158  )
1159
1159  )
1160
1160  )
1161
1161  )
1162
1162  )
1163
1163  )
1164
1164  )
1165
1165  )
1166
1166  )
1167
1167  )
1168
1168  )
1169
1169  )
1170
1170  )
1171
1171  )
1172
1172  )
1173
1173  )
1174
1174  )
1175
1175  )
1176
1176  )
1177
1177  )
1178
1178  )
1179
1179  )
1180
1180  )
1181
1181  )
1182
1182  )
1183
1183  )
1184
1184  )
1185
1185  )
1186

```

## 1.4 개발 환경 구축 (3/5)

- ✓ JavaScript는 기본적으로 Web 기반의 언어로 개발되었습니다. 따라서, 실행 범위가 Web Browser로 국한되었습니다.
- ✓ node.js는 V8 엔진 기반의 JavaScript 실행 환경을 제공합니다. 즉, Web Browser 이외의 환경에서도 JavaScript를 실행할 수 있습니다.
- ✓ 다음의 URL을 통해 개발 환경에 맞는 node.js를 설치합니다.
  - <https://nodejs.org/>



## 1.4 개발 환경 구축 (4/5)

- ✓ npm(Node Package Module)은 JavaScript 패키지와 모듈(modules)을 관리하는 도구입니다.
- ✓ npm은 node.js를 설치하면 기본적으로 설치 됩니다. 설치의 확인은 다음 명령을 통해 확인할 수 있습니다.
  - node -v : node 버전 확인.
  - npm -v : npm 버전 확인.
  - yarn -v : yarn 버전 확인.

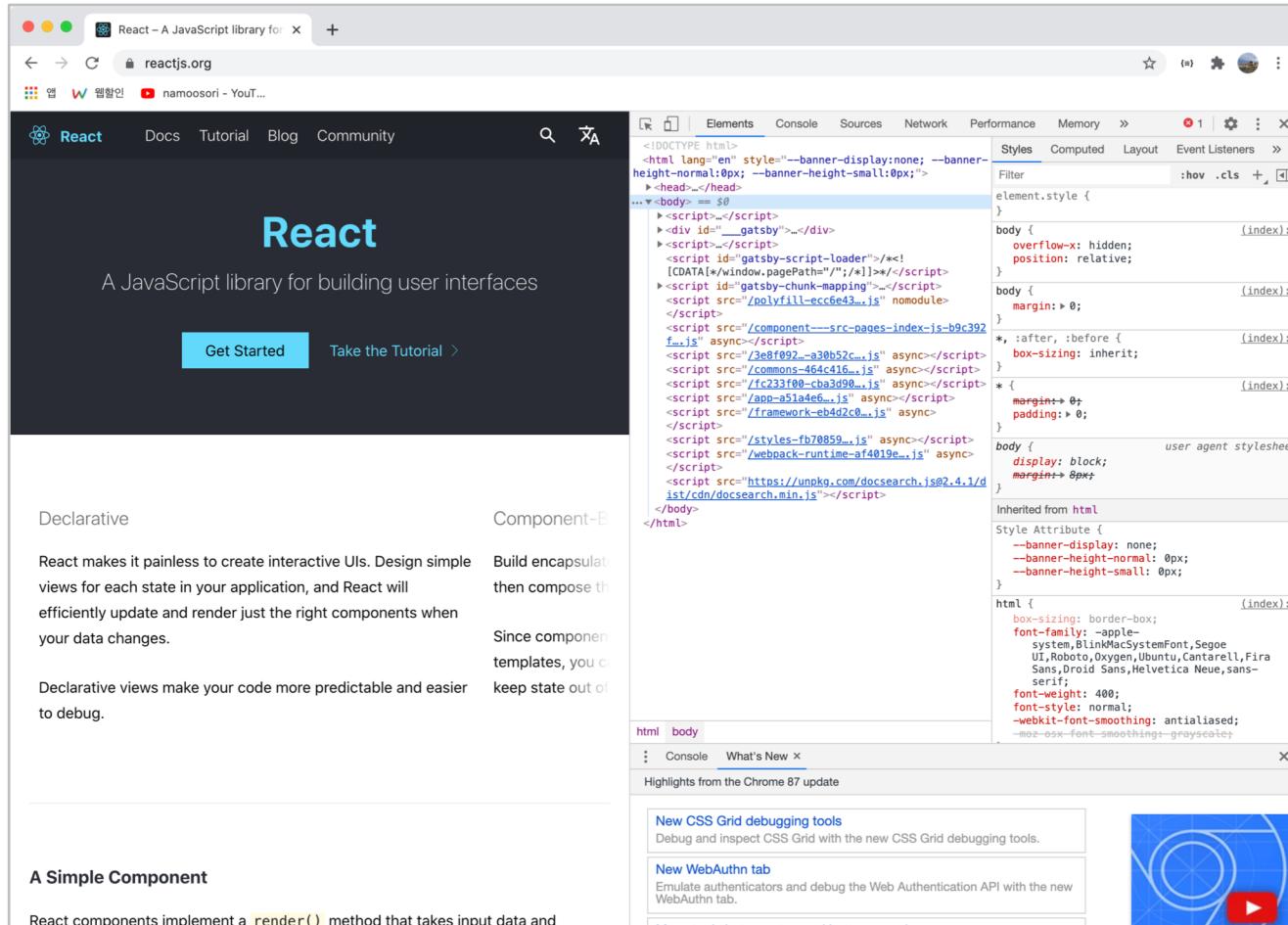


A screenshot of a macOS terminal window titled "kwon-kijin — kwon-kijin@Kwon-Kiui-MacBook-Pro — ~ — zsh — 80x20". The window shows the following command history:

```
[~] ~ node -v
v10.16.3
[~] ~ npm -v
6.9.0
[~] ~ yarn -v
1.19.0
[~]
```

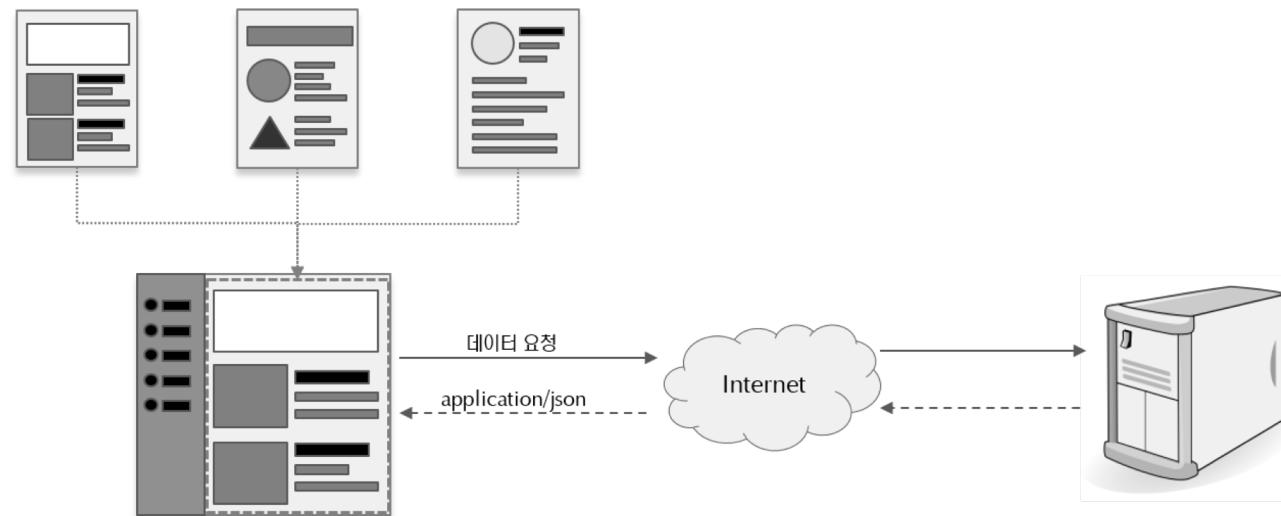
# 1.4 개발 환경 구축 (5/5)

- ✓ 웹브라우저는 IE, Chrome, Firefox 등 별도의 설치 없이도 사용 가능합니다.
- ✓ 개발에 사용할 웹 브라우저는 반드시 개발자 도구를 사용할 수 있어야 합니다.
- ✓ 대부분의 웹브라우저는 개발자 도구를 지원하고 있습니다.



# 요약

- ✓ React는 JavaScript로 만들어진 UI 컴포넌트 라이브러리이며 독립적인 이 컴포넌트들을 이용해 UI를 구성합니다.
- ✓ MPA 방식은 클라이언트의 요청이 있을 때 서버로 부터 새 페이지에 대한 전체 정보를 받아 표현합니다.
- ✓ SPA는 첫 페이지를 제외하고 서버로부터 필요한 데이터만 받습니다.





## 2. React Basic

---

**2.1 React 동작 방식**

**2.2 React Element**

**2.3 Virtual DOM**

**2.4 JSX**

## 2.1 React 동작 방식

- ✓ 기존 웹 애플리케이션은 클라이언트의 요청이 있으면 서버에서 UI(화면 구성 + 데이터)를 생성하여 응답합니다.
- ✓ React는 서버로부터 필요한 데이터를 요청하고 서버로부터 응답 받은 데이터를 이용해 사용자 UI를 구성합니다.
- ✓ 서버를 통해서는 필요 데이터만 받고, 클라이언트는 사용자 UI 전체를 관리합니다.
- ✓ 클라이언트에서 사용자 UI를 표현하는 것을 Client Side Rendering(CSR)이라고 합니다.



## 2.2 React Element

- ✓ React Element는 브라우저 DOM 엘리먼트에 대한 정보를 담고 있는 객체입니다.
- ✓ 실제 브라우저 DOM이 어떻게 만들어져야 하는지에 대한 설명을 가지고 있는 객체 리터럴입니다.
- ✓ 실제 DOM이 아닌 단순 객체이기 때문에 생성이 쉽습니다.
- ✓ ReactElement는 포함하는 값을 변경할 수 없으며 변경이 필요한 경우 새롭게 생성합니다.

```
let h1 = React.createElement('h1', null, 'Hello React!');  
// React Element 생성  
ReactDOM.render(  
  h1,  
  document.getElementById('root')  
);  
// 브라우저 DOM에 랜더링  
console.log(h1);  
// React Element 정보 출력
```

Hello React!

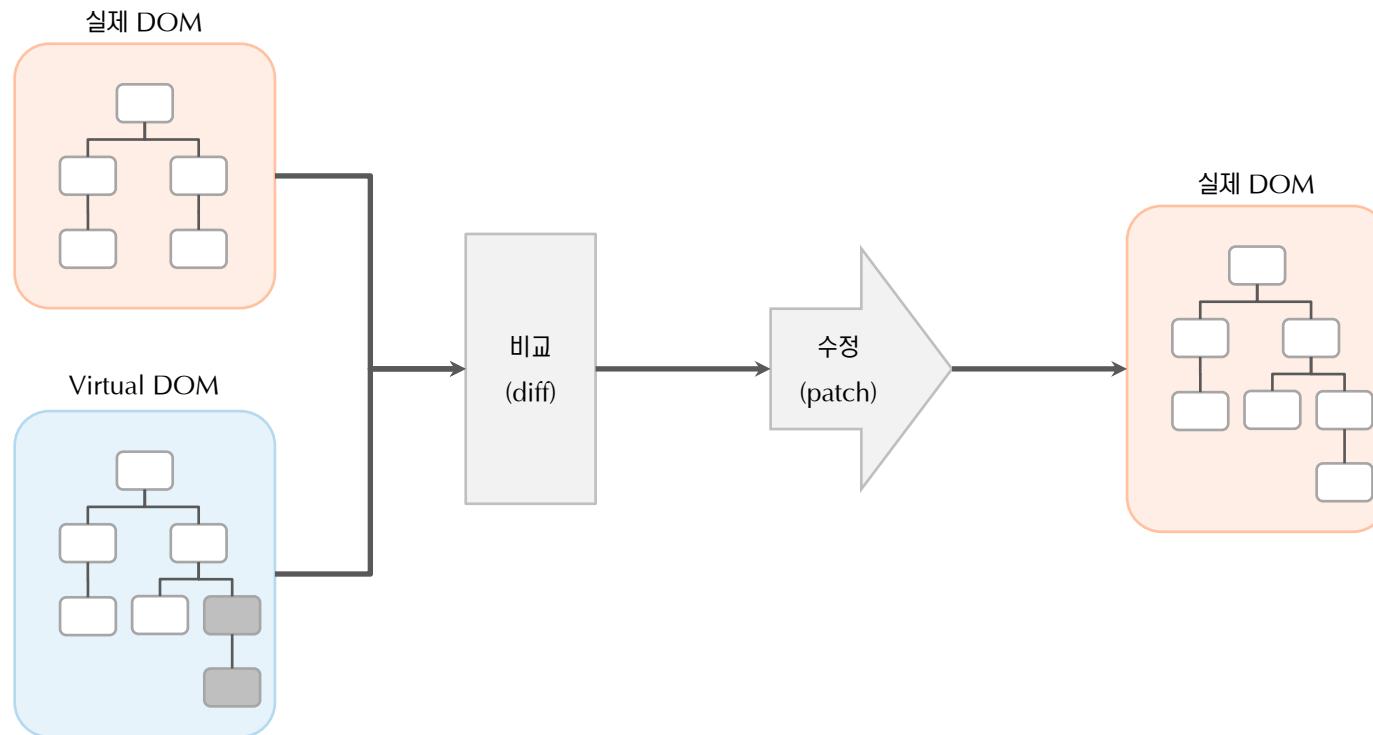
```
function tick() {  
  let element = React.createElement('h1', null, new Date().toLocaleTimeString());  
  ReactDOM.render(element, document.getElementById('root'));  
}  
setInterval(tick, 1000);
```

오후 1:45:54

```
1. {$$typeof: Symbol(react.element), type: "h1",  
key: null, ref: null, props: {...}, ...}  
1. $$typeof: Symbol(react.element)  
2. key: null  
3. props: {children: "Hello React!"}  
4. ref: null  
5. type: "h1"  
6. _owner: null  
7. _store: {validated: false}  
8. _self: null  
9. _source: null  
10. __proto__: Object
```

## 2.3 Virtual DOM

- ✓ Virtual DOM은 브라우저의 실제 DOM을 반영하고 있는 데이터의 모음으로 ReactElement로 구성됩니다.
- ✓ 웹 브라우저에서 화면의 변경은 곧 실제 DOM 구조의 변화를 뜻하며 기존 방식은 직접 접근을 통한 변경 방식입니다.
- ✓ React는 Virtual DOM이라는 실제 DOM과 비교 가능한 데이터 구조를 통해 변경이 필요한 요소를 변경합니다.
- ✓ 실제 DOM과 Virtual DOM의 차이를 파악하고 변경하는 과정은 비교(diff)하고 수정(patch)을 거칩니다.

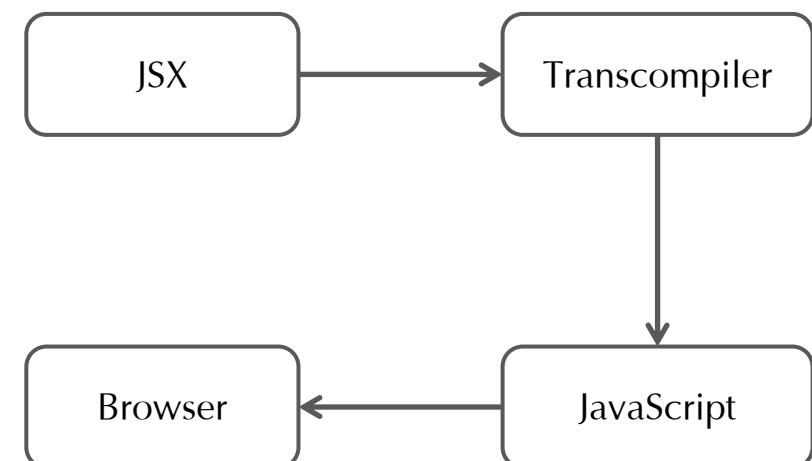


## 2.4 JSX (1/2)

- ✓ JSX(JavaScript XML)는 JavaScript의 확장으로 문법적 편의를 위해 만들어졌습니다.
- ✓ JSX로 작성한 코드는 트랜스컴파일러(Transcompiler)를 통해 JavaScript 코드로 변환 합니다.
- ✓ JSX를 이용하면 React.createElement() 함수 대신 HTML과 유사한 XML 형식의 코드를 작성할 수 있습니다.
- ✓ 중첩된 React 엘리먼트를 정의하려고 할 때 JavaScript 코드로 구현하는 것은 한계가 있으며 이런 경우 JSX를 이용하면 간결한 구현이 가능합니다.

```
//JSX
ReactDOM.render(
  <h1 className='heading'>Hello World</h1>,
  document.getElementById('content')
)
```

```
//JavaScript
ReactDOM.render(
  React.createElement(
    'h1',
    { className: 'heading' },
    'Hello World'
  )
)
```



## 2.4 JSX [2/2]

✓ JSX는 HTML과 유사한 형태이기 때문에 컴포넌트의 구조를 익숙한 방법으로 선언할 수 있습니다.

✓ JSX를 이용해 컴포넌트를 정의할 때 고려해야 할 몇가지 사항이 존재합니다.

- 대소문자 구분 : React 컴포넌트와 기본 HTML 요소와 대소문자로 구분하며, 컴포넌트는 Pascal Case로 작성합니다.  
예시) 컴포넌트-<List />, <Button/>      HTML 요소 : <h2 />, <input />, <a />
- {}의 사용 : JSX 코드상에서 JavaScript 코드를 반영하기 위해서는 {중괄호} 내부에 JavaScript 코드를 작성합니다.
- 예약어의 사용 : 표준 HTML의 사용에서 class, for 같은 예약어는 JSX에서 className, htmlFor를 대신하여 사용합니다.

```
class DateTimeNow extends React.Component {
  render(){
    let dateTimeNow = new Date().toLocaleDateString();
    return React.createElement(
      'span',
      null,
      '현재 시간 : ${dateTimeNow}'. // ECMAScript 6
    );
  }
}
```



```
class DateTimeNow extends React.Component {
  render(){
    let dateTimeNow = new Date().toLocaleDateString();
    return <span> 현재 시간 ${dateTimeNow}.</span>
  }
}
```

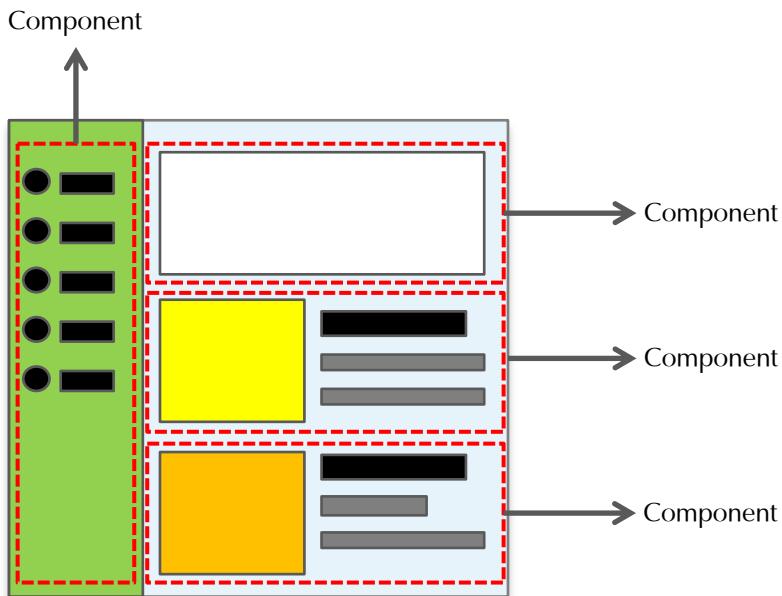


## 3. React Component

- 
- 3.1 React Component 개요
  - 3.2 Props 개요
  - 3.3 Props의 검증 - PropTypes
  - 3.4 Props의 적용

## 3.1 React Component 개요(1/2)

- ✓ 컴포넌트는 React UI 구성의 기본 단위이며, React는 컴포넌트를 생성하는 다양한 방법을 제공합니다.
- ✓ 캡슐화 된 컴포넌트는 재사용 및 재구성이 가능하며 컴포넌트들의 집합으로 사용자 UI를 구성합니다.
- ✓ React UI는 컴포넌트들의 집합으로 구성하며 컴포넌트는 클래스 기반의 컴포넌트와 함수 기반의 컴포넌트로 나뉩니다.
- ✓ 클래스 기반으로 생성하는 컴포넌트는 반드시 `render()` 메서드를 재정의 해야 하며 `render()` 메서드는 하나의 React Element를 반환합니다.



```
let element = React.createElement('h1', null, "Component Test");

class MyComponent extends React.Component {
  render(){
    return React.createElement('div', null, element, element);
  }
}

ReactDOM.render(
  React.createElement(MyComponent, null), document.getElementById('root')
)
```

Component Test  
Component Test

## 3.1 React Component 개요(2/2)

- ✓ 컴포넌트는 화면을 구성하는 다양한 요소(기능, 스타일, 마크업 등)를 그룹화 하는 방법입니다.
- ✓ 사용자 UI를 여러 부분(part)로 나누는 경계가 되며, 컴포넌트는 다른 컴포넌트를 포함할 수 있습니다.
- ✓ 컴포넌트는 독립적이면서도 재사용이 가능하기 때문에 필요한 기능들을 독자적으로 구성할 수 있습니다.
- ✓ 컴포넌트는 클래스 기반의 상태가 있는(stateful) 컴포넌트와 함수 기반 상태가 없는(stateless) 컴포넌트로 구분합니다.

```
class Customer extends React.Component {
  render() {
    const { id, name, orders } = this.props;

    return (
      <div className="customer">
        <h2>{id}</h2>
        <p>
          <span>이름 : {name}</span><br/>
          <span>주문 수량 : {orders.length} 개</span>
        </p>
      </div>
    )
  }
}
```

```
function App(){
  return <HelloMessage />
}

function HelloMessage(){
  const message = "Hello~~";
  return <h1> {message} </h1>
}

const App = () => {
  return <HelloMessage />
}

const HelloMessage = () => {
  const message = 'Hello~~';
  return <h1>{message}</h1>
}
```

## 3.2 Props 개요

- ✓ React.Component 클래스를 확장(extends)해 정의하는 클래스 기반의 컴포넌트는 props 객체를 갖습니다.
- ✓ props는 해당 컴포넌트의 생성 시점에 상위 컴포넌트로 부터 전달되는 데이터가 할당되는 객체입니다.
- ✓ props를 통해 상위 컴포넌트로부터 전달되는 데이터의 종류는 크게 2가지 형태입니다.
  - Html 요소의 속성 : href, title, style, class 등
  - 상위 컴포넌트로부터 할당 받는 임의의 데이터 : this.props.속성명
- ✓ props를 통해 전달 되는 데이터는 해당 컴포넌트에서 값을 변경할 수 없습니다.

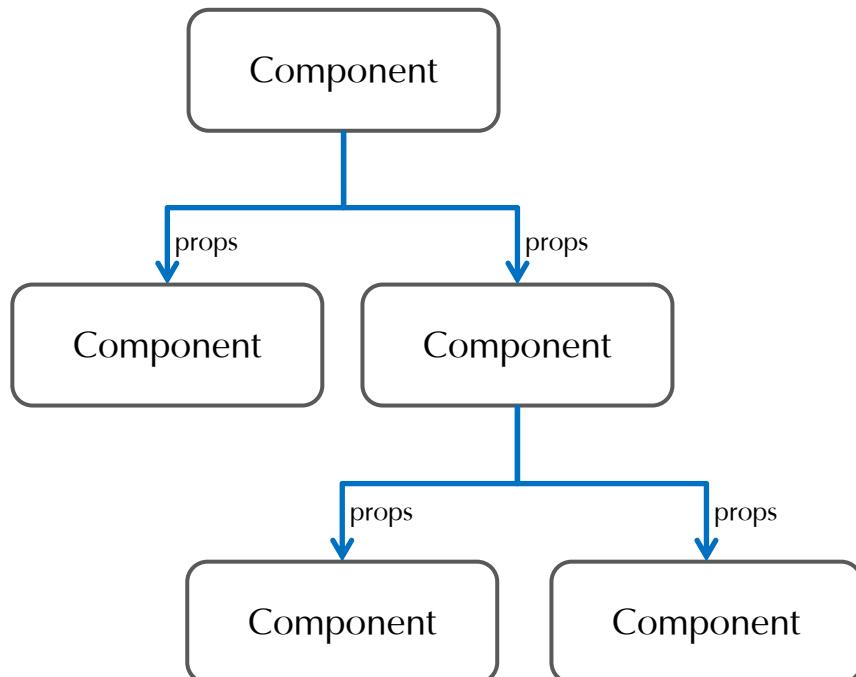
```
<Customer id='kim@namoosori.io' name='Kim' orders={['A0001', 'B0002', 'C0003']} />
```

```
class Customer extends React.Component {  
  render() {  
    const { id, name, orders } = this.props;  
    return (  
      <div className="customer">  
        <h2>{id}</h2>  
        <p>  
          <span>이름 : {name}</span><br/>  
          <span>주문 수량 : {orders.length} 개</span>  
        </p>  
      </div>  
    )  
  }  
}
```

kim@namoosori.io  
이름 : Kim  
주문 수량 : 3 개

### 3.3 Props의 검증 – PropTypes[1/2]

- ✓ JavaScript는 타입 검증이 느슨한 언어이며 이는 변수에 할당되는 데이터 타입의 제한이 느슨한(loosely) 것을 뜻합니다.
- ✓ Props로 전달 받는 데이터 타입의 검증은 데이터를 받는 해당 컴포넌트가 담당합니다.
- ✓ React.PropTypes는 상위 컴포넌트로 부터 전달받는 props의 타입에 대한 검증을 수행합니다.
- ✓ prop-types 라이브러리를 이용해서 해당 컴포넌트의 props에 할당된 데이터 유효성 검사를 진행합니다.



| Type     | Verification           |
|----------|------------------------|
| Array    | React.PropTypes.array  |
| Boolean  | React.PropTypes.bool   |
| Function | React.PropTypes.func   |
| Number   | React.PropTypes.number |
| Object   | React.PropTypes.object |
| String   | React.PropTypes.string |

### 3.3 Props의 검증 – PropTypes[2/2]

- ✓ PropTypes를 통해 상위 컴포넌트로부터 전달받는 props를 검증합니다.
- ✓ PropTypes는 props에 대한 타입 검증이며 타입이 다를 경우 개발자 콘솔(console)을 통해 경고가 표시됩니다.
- ✓ PropTypes를 반드시 적용해야 하는 것은 아니지만 버그의 예방과 디버깅을 위해 사용합니다.
- ✓ DefaultProps를 이용하면 props에 기본값을 설정하는 것도 가능합니다.

```
<Customer id='kim@namoosori.io' name='Kim' orders='A0001, B0002, C0003' />
```

```
class Customer extends Component {
  static propTypes = {
    id : PropTypes.string.isRequired,
    name : PropTypes.string.isRequired,
    orders : PropTypes.array.isRequired
  }

  render() {
    const { id, name, orders } = this.props;
    return (
      <div className="customer">
        <h2>{id}</h2>
        <p>
          <span>이름 : {name}</span><br/>
          <span>주문 수량 : {orders.length} 개</span>
        </p>
      </div>
    )
  }
}
```

kim@namoosori.io

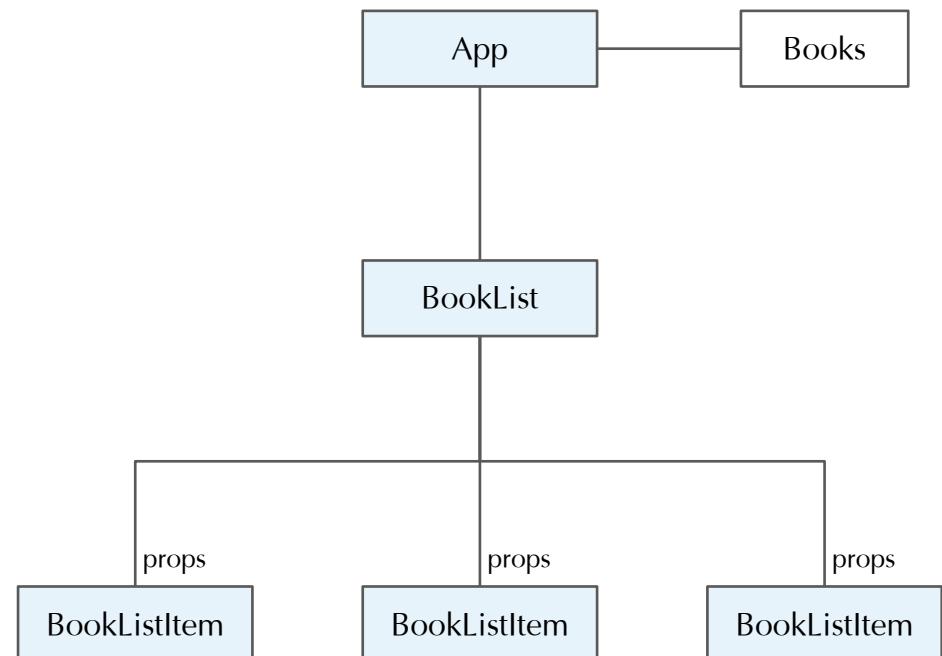
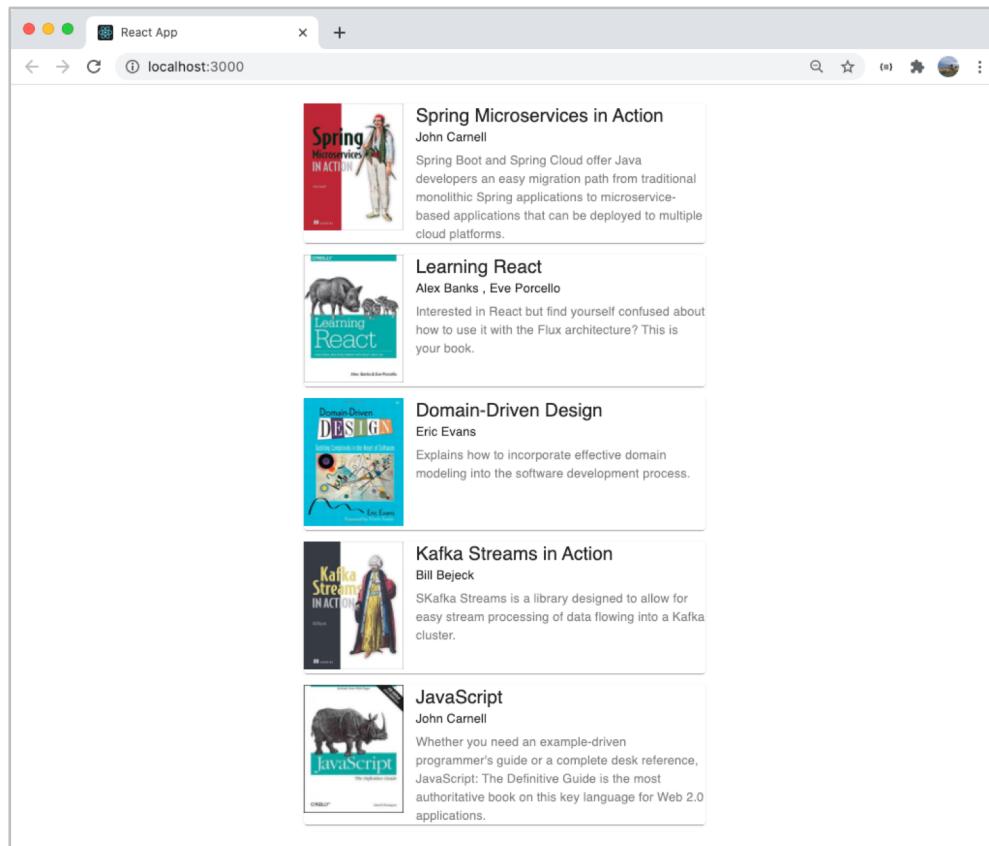
이름 : Kim

주문 수량 : 19 개

```
✖ ▶ Warning: Failed prop type: Invalid prop `orders` of
  type `string` supplied to `Customer`, expected `array`.
  in Customer (at App.js:14)
  in App (at src/index.js:7)
```

## 3.4 Props의 적용

- ✓ 다음은 props를 통해 포함 계층 구조에서 하위 컴포넌트에 값을 전달하는 예제입니다.
- ✓ App 컴포넌트는 Books.js를 통해 리스트에 표현할 값을 참조합니다.
- ✓ BookListItem 컴포넌트는 BookList로부터 props를 통해 표현할 데이터를 전달 받습니다.
- ✓ 컴포넌트의 표현을 위해 Material-UI React 를 사용합니다.





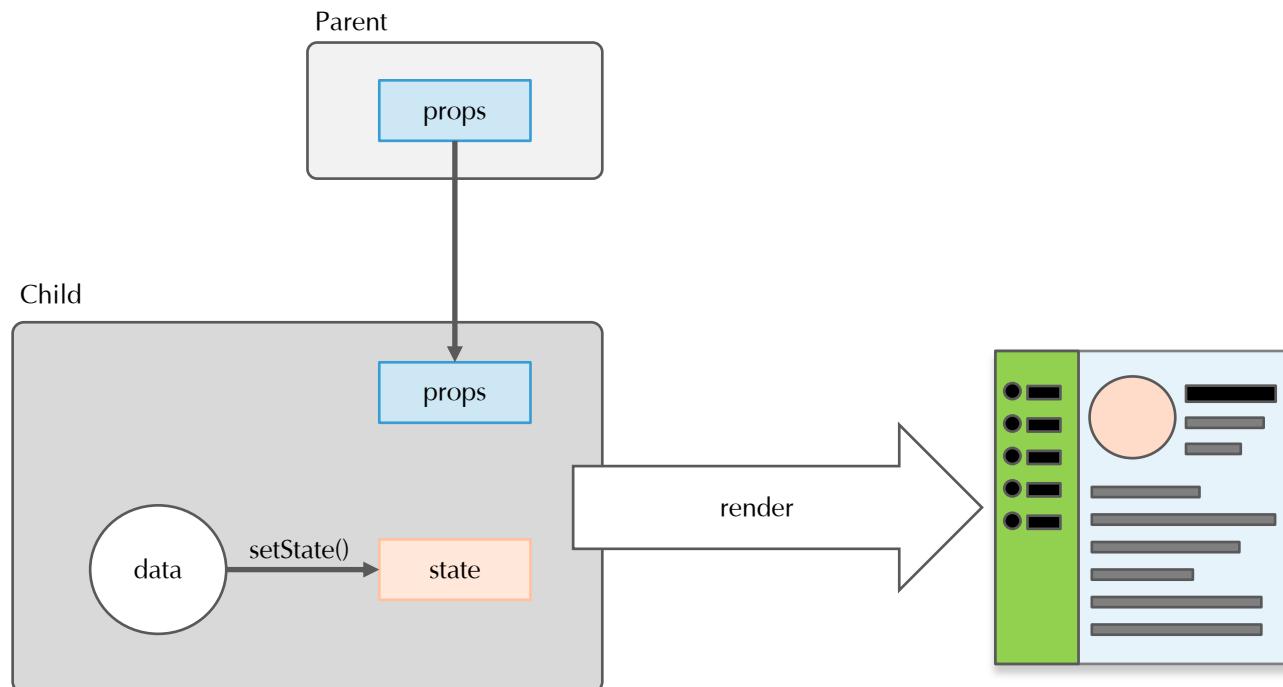
## 4. State

---

- 4.1 State 개요
- 4.2 State의 초기화와 변경
- 4.3 State의 적용

## 4.1 State 개요

- ✓ props는 해당 컴포넌트의 입장에서 불변(immutable) 데이터입니다.
- ✓ 컴포넌트에서 변경 가능한 데이터를 관리하기 위해 사용하는 객체는 state입니다.
- ✓ State는 React.Component 클래스를 상속한 클래스 기반의 컴포넌트에만 존재합니다.
- ✓ State 값의 초기화는 객체 필드의 선언부 혹은 생성자에서 구현하며, state 값의 변경은 setState() 메서드를 이용합니다.



## 4.2 State 초기화와 변경

- ✓ 생성자(constructor)는 클래스가 인스턴스화 될 때 한번 호출되는 특수한 메서드입니다.
- ✓ state 객체에 값을 대입(assign)하여 초기화 하는 것은 해당 컴포넌트 객체의 필드 선언부 혹은 생성자에서 진행합니다.
- ✓ state 초기화를 위해 생성자를 정의할 때에는 super 생성자를 호출하여 부모 클래스에 props를 전달합니다. 이는 this 참조를 사용하기 위한 코드입니다.
- ✓ state 객체에 값을 변경하기 위해서는 반드시 setState() 메서드를 호출하여 변경합니다.

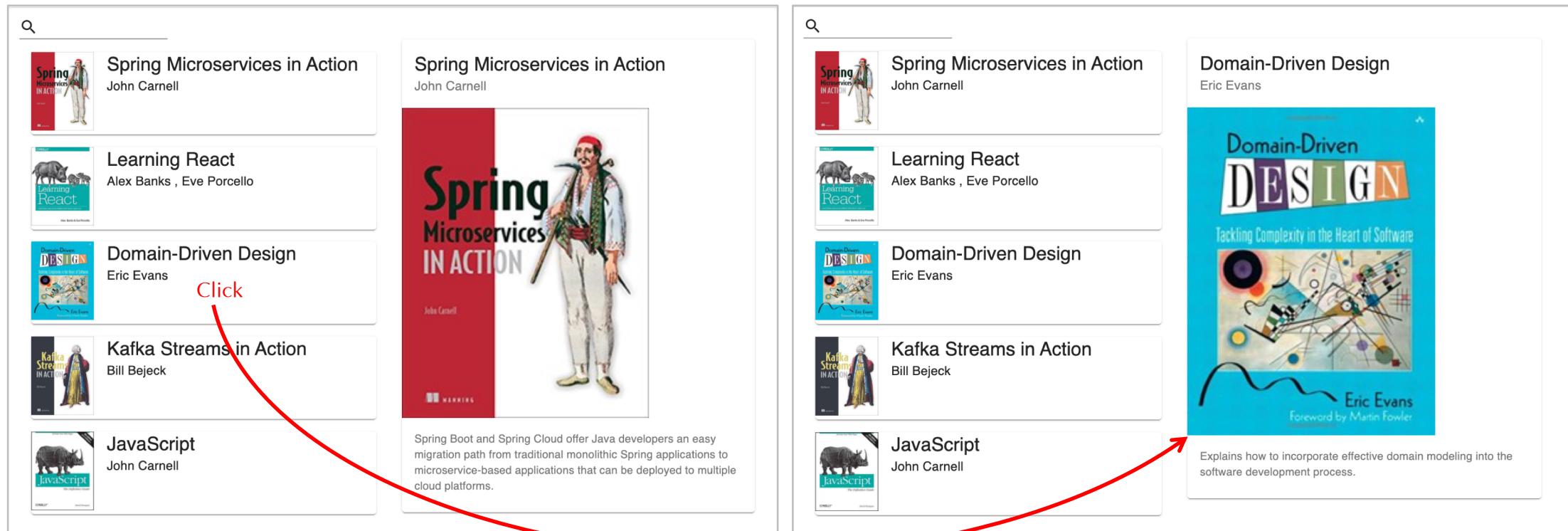
```
class StateApp extends Component {  
    constructor(props) {  
        super(props);  
        this.state = {  
            books: Books,  
            selectedBook: Books[0]  
        }  
    }  
    searchByTitle(title) {  
        let updateList = Books;  
        updateList = updateList.filter(book => {  
            return book.title.toLowerCase().search(title.toLowerCase()) !== -1;  
        });  
        this.setState({  
            books: updateList  
        })  
    }  
}
```

state 초기화

state 변경

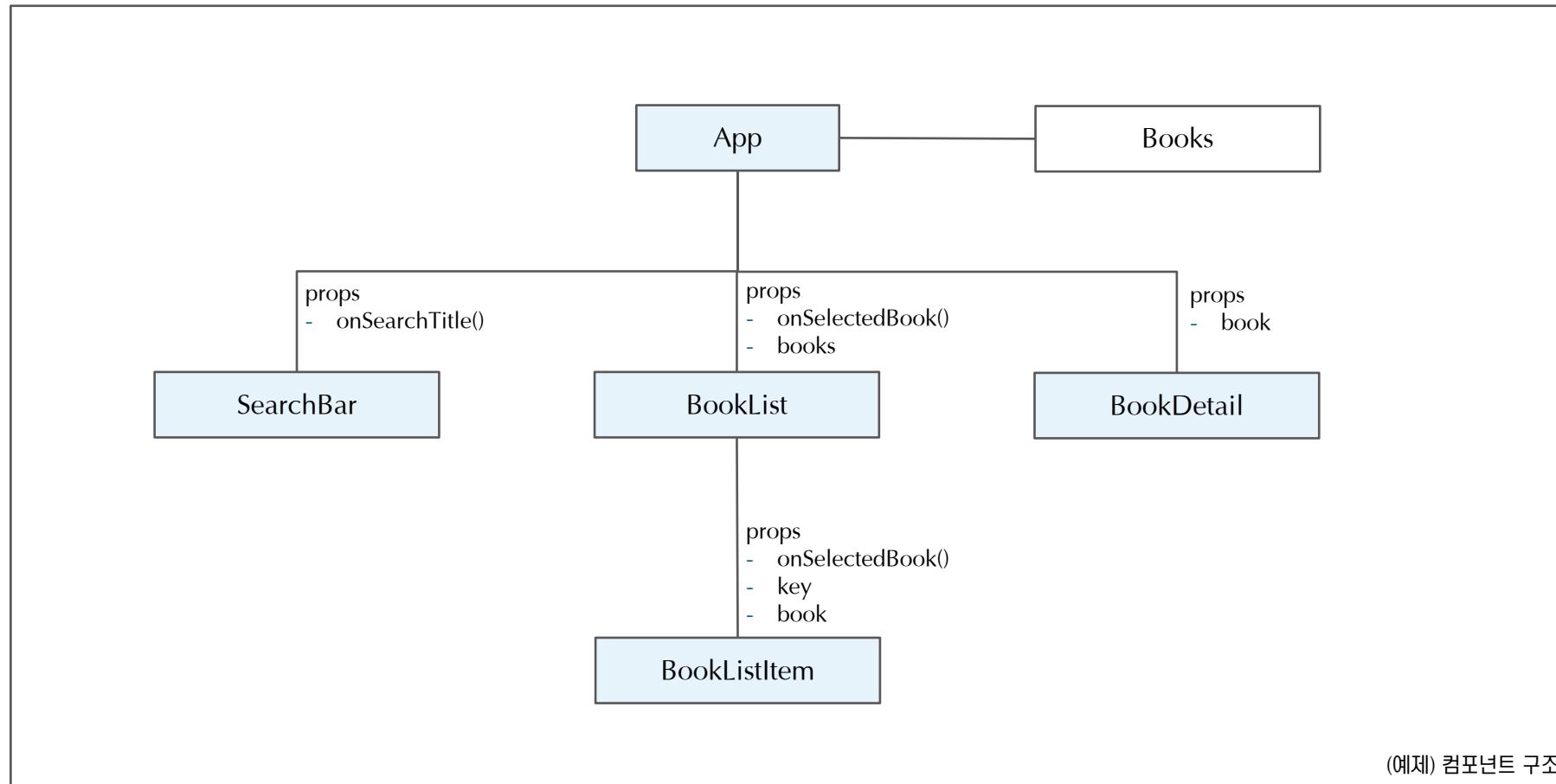
## 4.3 State의 적용 [1/2]

- ✓ state 객체는 해당 컴포넌트가 갖는 데이터를 관리하며, 이 데이터의 변경을 통해 화면을 갱신합니다.
- ✓ **React.Component** 클래스를 확장하는 모든 클래스 기반 컴포넌트는 state 객체를 갖습니다.
- ✓ 하나의 UI를 구성하는 다양한 컴포넌트가 각각 별도의 state를 갖고 관리하는 것은 프로그램의 복잡도를 올립니다.
- ✓ 루트(root)가 되는 컴포넌트가 전체 state를 갖게 하고 하위 컴포넌트들은 props를 통해 불변의 데이터를 받습니다.



## 4.3 State의 적용 [2/2]

- ✓ state는 JavaScript 객체인 만큼 다양한 종류의 데이터를 담을 수 있습니다.
- ✓ App 컴포넌트는 books 데이터를 가지며 state 객체를 통해 관리 합니다.
- ✓ SearchBar, BookList, BookDetail 컴포넌트는 App 컴포넌트로부터 props 객체 데이터로 책의 정보를 받습니다.





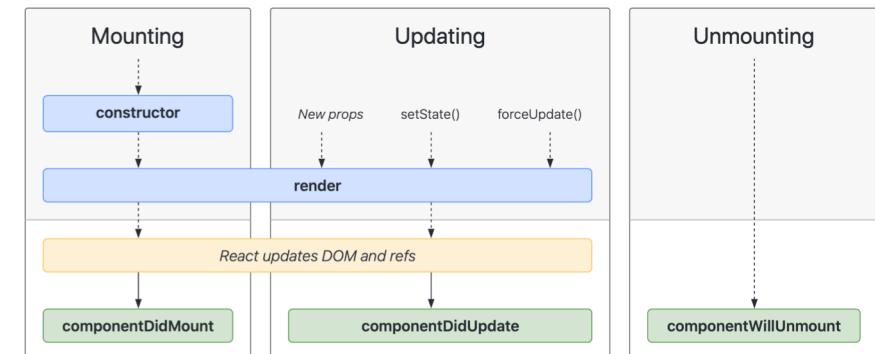
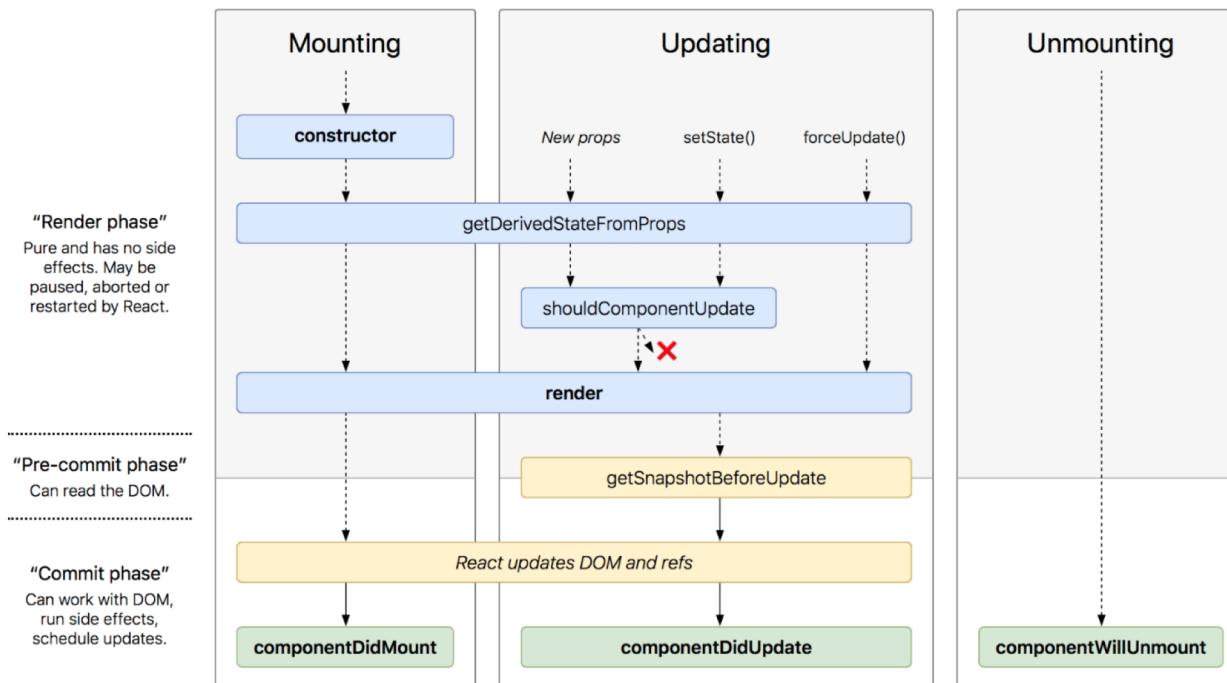
## 5. Component Lifecycle

5.1 Component Lifecycle 개요

5.2 주요 Component Lifecycle 메서드

# 5.1 Component Lifecycle 개요

- ✓ 생명주기(Lifecycle) 메서드는 클래스 기반 컴포넌트에 존재하는 메서드로 특정 시점에 호출되는 메서드입니다.
- ✓ 생명주기 메서드는 컴포넌트의 마운트(mounting), 업데이트(updating), 마운트 해제(unmounting) 시점에 따라 구분 하며 모두 8개의 메서드가 있습니다.
- ✓ 컴포넌트가 생성되고 `render()` 메서드를 통해 `ReactElement`를 반환 실제 DOM에 반영되는 것이 마운트입니다.
- ✓ `props, state`의 데이터가 변경되거나 `forceUpdate()` 메서드 호출을 통해 컴포넌트가 변경되는 것이 업데이트입니다.



출처: <http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

## 5.2 주요 Component Lifecycle 메서드 (1/3)

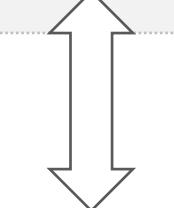
- ✓ 마운트 과정에서 호출되는 주요 메서드는 `constructor()` 메서드와 `componentDidMount()` 메서드입니다.
- ✓ 마운트가 된다는 것은 React 컴포넌트가 실제 DOM에 삽입 된다는 것을 의미 합니다.
- ✓ `constructor()` 메서드는 해당 컴포넌트의 인스턴스가 생성될 때 호출되는 메서드로 state 객체를 초기화하거나 특정 메서드를 바인딩(binding) 하는 작업을 진행합니다.
- ✓ `componentDidMount()` 메서드는 컴포넌트가 마운트 된 직후 호출되며 데이터 로딩과 같은 작업 등에 재정의 합니다.

```
constructor(props) {
  super(props); // 필수 코드
  this.state = {
    counter : 0
  }
  this.onButtonClick = this.onButtonClick.bind(this);
  // 이벤트 메소드 바인딩
}
```

[참고]

- `super()` 생성자 호출 이전에는 `this`가 할당되지 않습니다.
- `super()` 생성자 파라미터로 `props`의 전달의 목적은 생성자 내부에서 `this.props` 접근이 가능하도록 하기 위해서입니다.

```
componentDidMount(){
  // 데이터 로딩
}
```



## 5.2 주요 Component Lifecycle 메서드 (2/3)

- ✓ `render()` 메서드는 클래스 기반 컴포넌트에서 반드시 재정의 해야 하는 메서드로 React element를 반환합니다.
- ✓ React 16.x 이전 버전까지는 `render()` 메서드가 하나의 element만 반환해야 했지만 16.x 버전부터는 배열을 이용해 다수의 React element를 반환할 수 있습니다.
- ✓ `getDerivedStateFromProps()` 메서드는 `nextProps`, `prevState` 두 개의 전달인자를 받으며 새로운 `props` 값으로 state 값을 동기화 하고자 할 때 사용합니다.

```
import React, { Component } from 'react';
class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }
  onAddButtonHandler() {
    this.setState({
      count: this.state.count + 1,
    });
  }
  shouldComponentUpdate() {
    console.log('Lifecycle : shouldComponentUpdate()');
    return true;
  }
  getSnapshotBeforeUpdate() {
    console.log('Lifecycle : getSnapshotBeforeUpdate()');
    return null;
  }
  componentDidMount() {
    console.log('Lifecycle : componentDidMount()');
  }
  componentDidUpdate() {
    console.log('Lifecycle : componentDidUpdate()');
  }
}
```

```
render() {
  console.log('Lifecycle : render()');
  return (
    <div>
      <label>{this.state.count}</label>
      <br />
      <button type="button" onClick={this.onAddButtonHandler.bind(this)}>
        +
      </button>
    </div>
  );
}
export default Counter;
```

|                                       |               |
|---------------------------------------|---------------|
| Lifecycle : render()                  | Counter.js:30 |
| Lifecycle : render()                  | Counter.js:30 |
| Lifecycle : componentDidMount()       | Counter.js:24 |
| Lifecycle : shouldComponentUpdate()   | Counter.js:16 |
| Lifecycle : shouldComponentUpdate()   | Counter.js:16 |
| Lifecycle : render()                  | Counter.js:30 |
| Lifecycle : render()                  | Counter.js:30 |
| Lifecycle : getSnapshotBeforeUpdate() | Counter.js:20 |
| Lifecycle : componentDidUpdate()      | Counter.js:27 |

## 5.2 주요 Component Lifecycle 메서드 (3/3)

- ✓ `shouldComponentUpdate()` 메서드는 성능 최적화를 위해 사용하는 메서드입니다.
- ✓ React는 Virtual DOM과 실제 DOM의 상태를 비교하여 변경된 부분에 대해서만 새로운 렌더링을 진행합니다.
- ✓ `render()` 메서드는 Virtual DOM을 대상으로 렌더링을 진행하는 메서드입니다.
- ✓ `shouldComponentUpdate()`에서 이전 상태(state, props)를 비교하여 `render()` 호출 여부를 결정할 수 있습니다.
- ✓ `shouldComponentUpdate()`를 재정의 할 수도 있지만 `React.PureComponent`를 확장하여 사용합니다.

```
import React, { Component } from 'react';
class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }
  onAddButtonHandler() {
    this.setState({
      count: this.state.count + 1,
    });
  }
  shouldComponentUpdate() {
    console.log('Lifecycle : shouldComponentUpdate()');
    return false;
  }
  getSnapshotBeforeUpdate() {
    console.log('Lifecycle : getSnapshotBeforeUpdate()');
    return null;
  }
  componentDidMount() {
    console.log('Lifecycle : componentDidMount()');
  }
}
```

```
componentDidUpdate() {
  console.log('Lifecycle : componentDidUpdate()');
}
render() {
  console.log('Lifecycle : render()');
  return (
    <div>
      <label>{this.state.count}</label>
      <br />
      <button type="button" onClick={this.onAddButtonHandler.bind(this)}>
        +
      </button>
    </div>
  );
}
export default Counter;
```

|                                     |                               |
|-------------------------------------|-------------------------------|
| Lifecycle : render()                | <a href="#">Counter.js:31</a> |
| Lifecycle : render()                | <a href="#">Counter.js:31</a> |
| Lifecycle : componentDidMount()     | <a href="#">Counter.js:25</a> |
| Lifecycle : shouldComponentUpdate() | <a href="#">Counter.js:16</a> |
| Lifecycle : shouldComponentUpdate() | <a href="#">Counter.js:16</a> |



## 6. MobX

---

**6.1 state 관리 개요**

**6.2 flux**

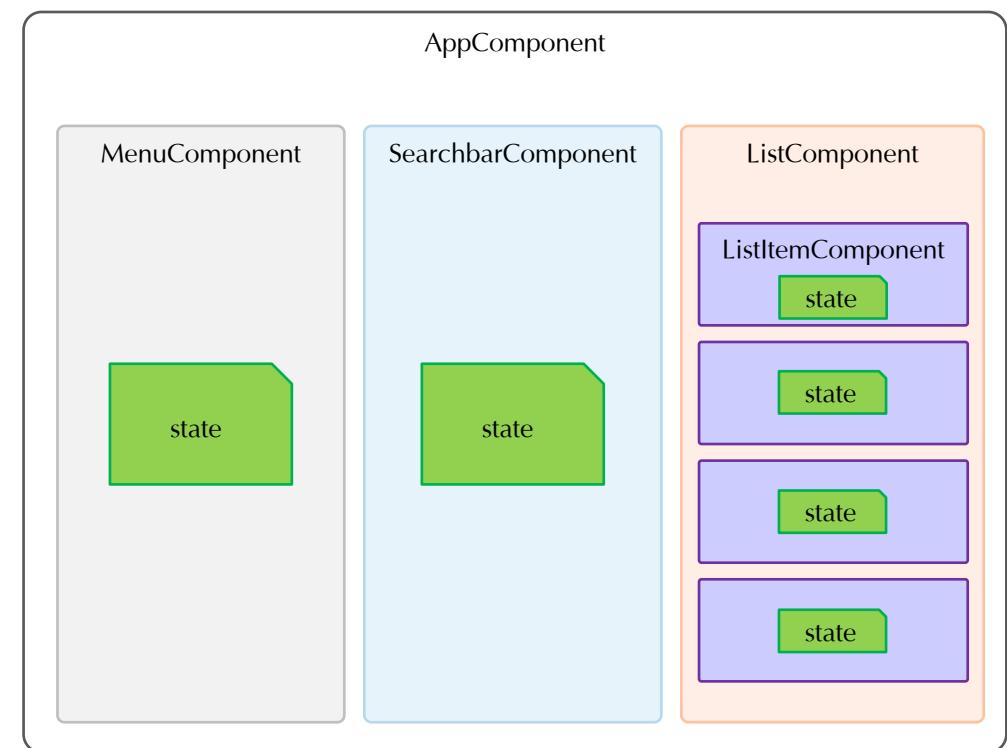
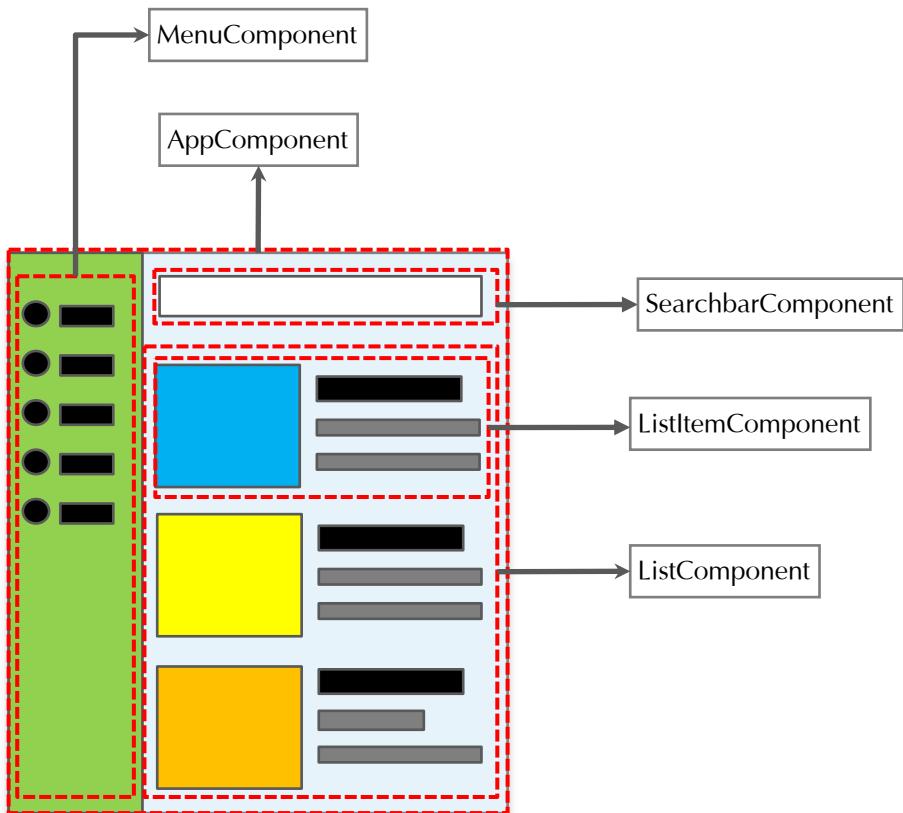
**6.3 MobX 개요**

**6.4 MobX의 적용**

**6.5 MobX 예제 실습**

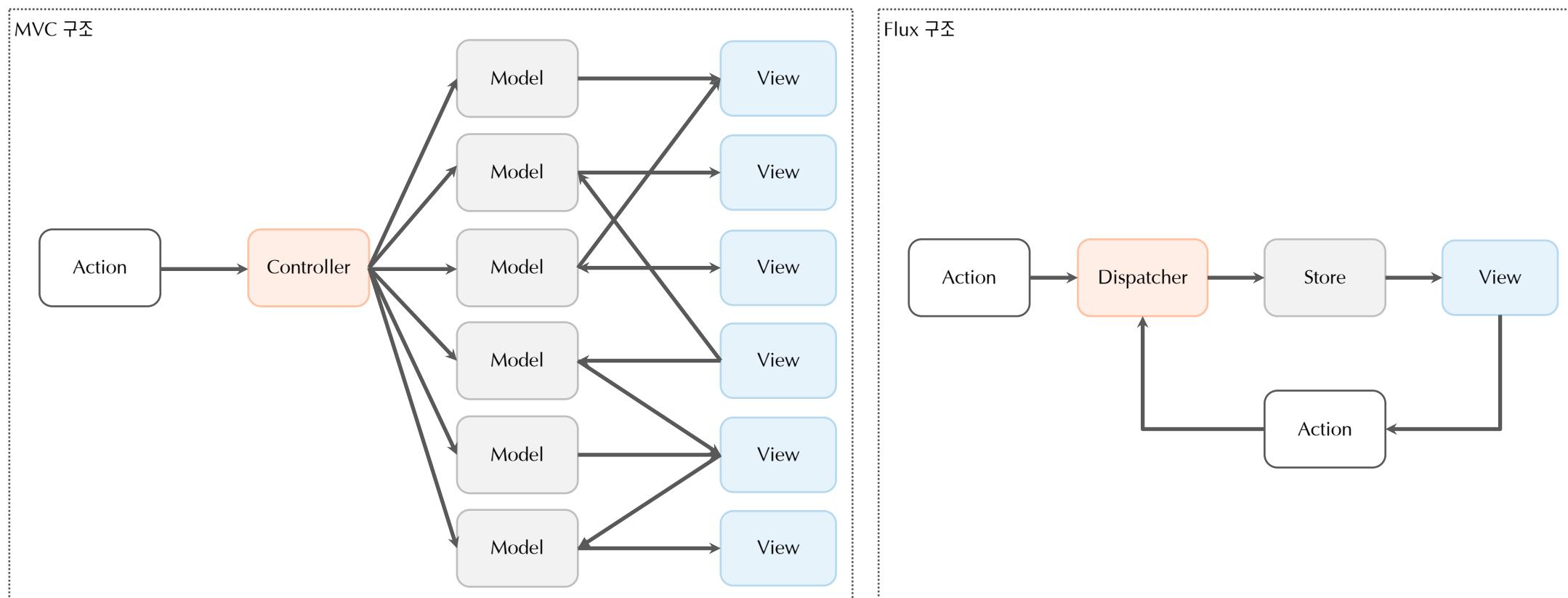
## 6.1 state 관리 개요

- ✓ 클래스 기반의 컴포넌트는 state 객체를 이용해 변경 가능한 데이터를 관리 할 수 있습니다.
- ✓ 사용자 UI는 다수의 컴포넌트를 통해 표현되며 각 컴포넌트가 상태를 관리할 경우 데이터 관리 및 제어가 어렵습니다.
- ✓ 각각의 컴포넌트에서 특정 데이터의 공유가 필요한 경우 이 데이터를 참조하거나 변경할 때 제약이 발생합니다.
- ✓ 공통의 데이터 관리 영역(store)을 두고 이를 통해 state를 관리하면 컴포넌트 간 데이터 공유가 가능합니다.



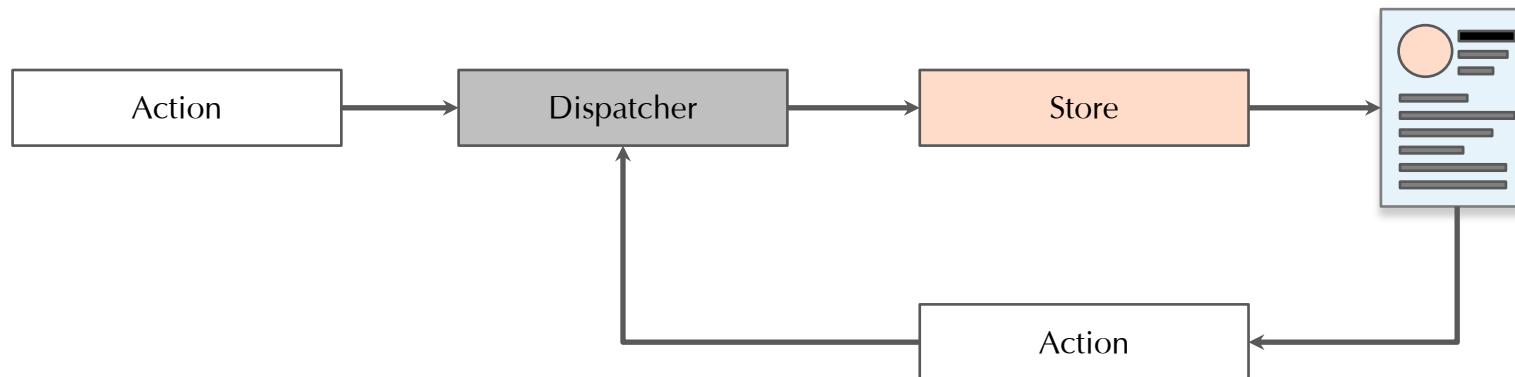
## 6.2 Flux[1/2]

- ✓ Flux는 React를 이용한 UI 구성에서 데이터의 흐름을 관리하는 어플리케이션 아키텍처입니다.
- ✓ MVC 구조의 데이터 흐름은 View와 Model 사이에 양방향 데이터 이동이 가능합니다.
- ✓ MVC 구조는 View가 많아 질수록 데이터의 흐름과 관리가 어렵습니다.
- ✓ React는 단방향 데이터 흐름(Model → View)만 가능하기 때문에 MVC 아키텍처는 적용하지 않습니다.
- ✓ 단방향 데이터 흐름은 구조를 단순화 할 수 있으며 데이터의 이동 또한 명확하게 확인할 수 있습니다.



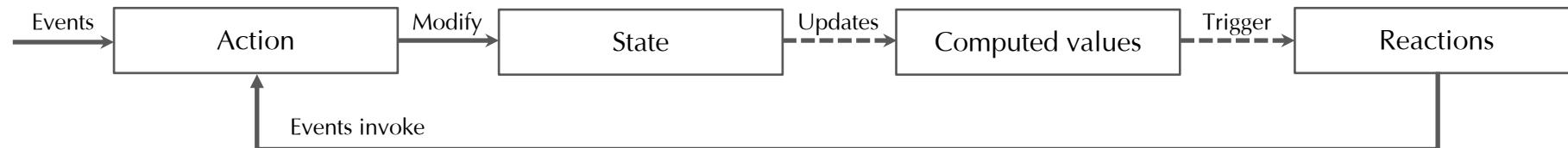
## 6.2 Flux[2/2]

- ✓ Flux는 단방향 데이터 흐름을 보완하기 위해 개발된 아키텍처이며 View로 분산된 state를 통합 관리합니다.
- ✓ Flux 아키텍처는 Action, Store, Dispatcher, View로 구성됩니다.
- ✓ View 각각의 state는 Store를 이용해 통합 관리되며, Store의 데이터는 Action을 이용해 제어합니다.
- ✓ Store에서 제어하는 state는 곧 연결된 View의 state와 다름없으며 Store의 state가 변경되면 View도 갱신됩니다.



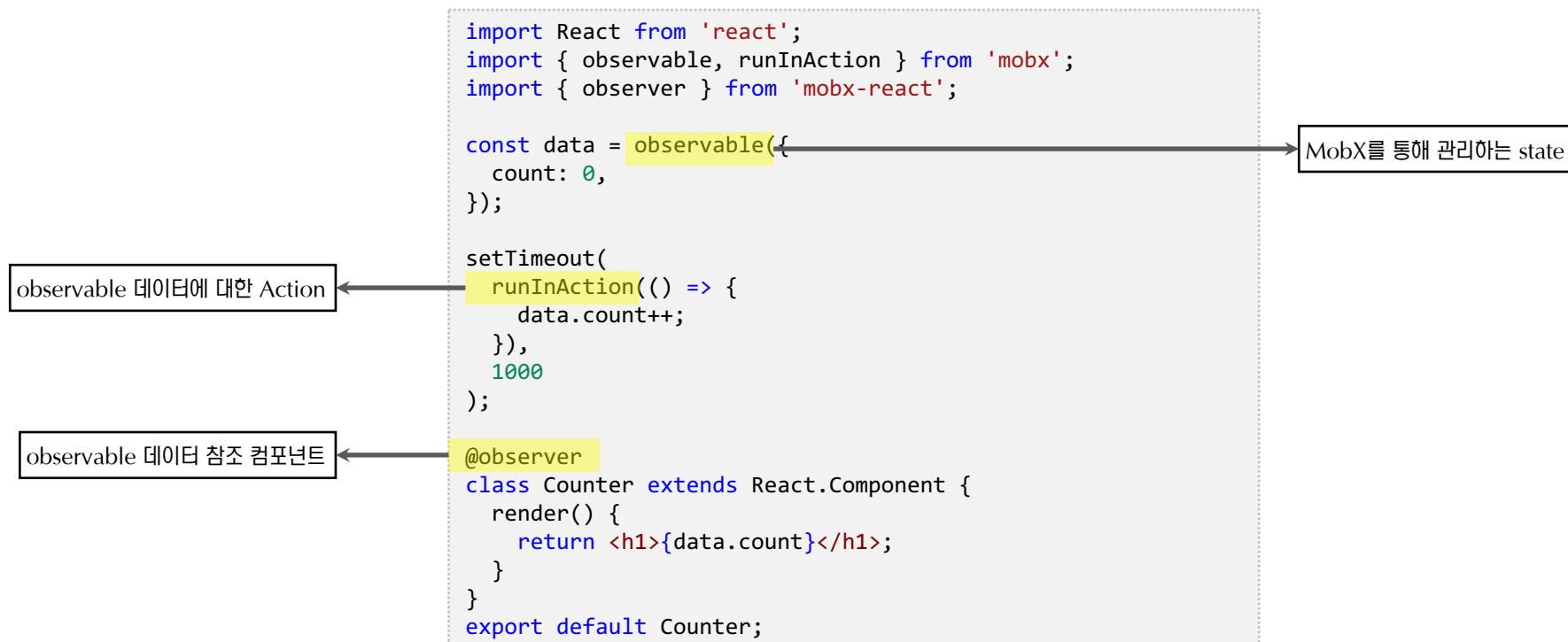
## 6.3 MobX 개요(1/4)

- ✓ MobX도 Flux와 마찬가지로 클라이언트 사이드에서 state를 관리하기 위해 사용하는 라이브러리입니다.
- ✓ MobX를 이용하면 컴포넌트의 state를 별도의 영역에서 관리하고 각 컴포넌트는 이에 접근할 수 있습니다.
- ✓ MobX를 적용하기 위해서는 mobx.js 라이브러리와 mobx-react.js 라이브러리가 필요합니다.
- ✓ MobX는 다수의 store를 관리할 수 있으며, 관리하는 데이터는 특정 데이터의 형태(Observable)로 관리 합니다.



## 6.3 MobX 개요(2/4)

- ✓ MobX가 제공하는 대표적인 API는 observable, action, observer, computed가 있습니다.
- ✓ MobX 라이브러리는 TypeScript가 적용되어 있으며 API의 사용은 데코레이터(@)를 이용하는 것이 일반적입니다.
- ✓ @observable API는 store에서 관리하고자 하는 state 데이터를 의미하며 Observable 객체를 통해 관리됩니다.
- ✓ @observer API는 @observable API로 관리되는 state를 참조하는 React 컴포넌트에 적용합니다.



## 6.3 MobX 개요(3/4)

- ✓ **@action**은 관찰 대상 데이터 즉, **observable state**의 값을 변경하는 메서드에 적용합니다.
- ✓ **state**에 대한 단순 조회와 같은 메서드에 적용하는 것은 의미가 없습니다.
- ✓ **@computed**는 **get** 메서드에 일반적으로 적용하거나 Model 객체간 전환 시점에 적용합니다.
- ✓ **@computed**가 적용된 메서드를 수행할 때 해당 **observable state**의 변화가 없을 경우 내부 로직을 생략합니다.

```
import { observable, computed, autorun } from 'mobx';

class TodoListStore {
  @observable todos = [];

  constructor(){
    autorun(() => {
      console.log("Number of unfinished : " + this.unfinishedTodoCount);
    });
  }

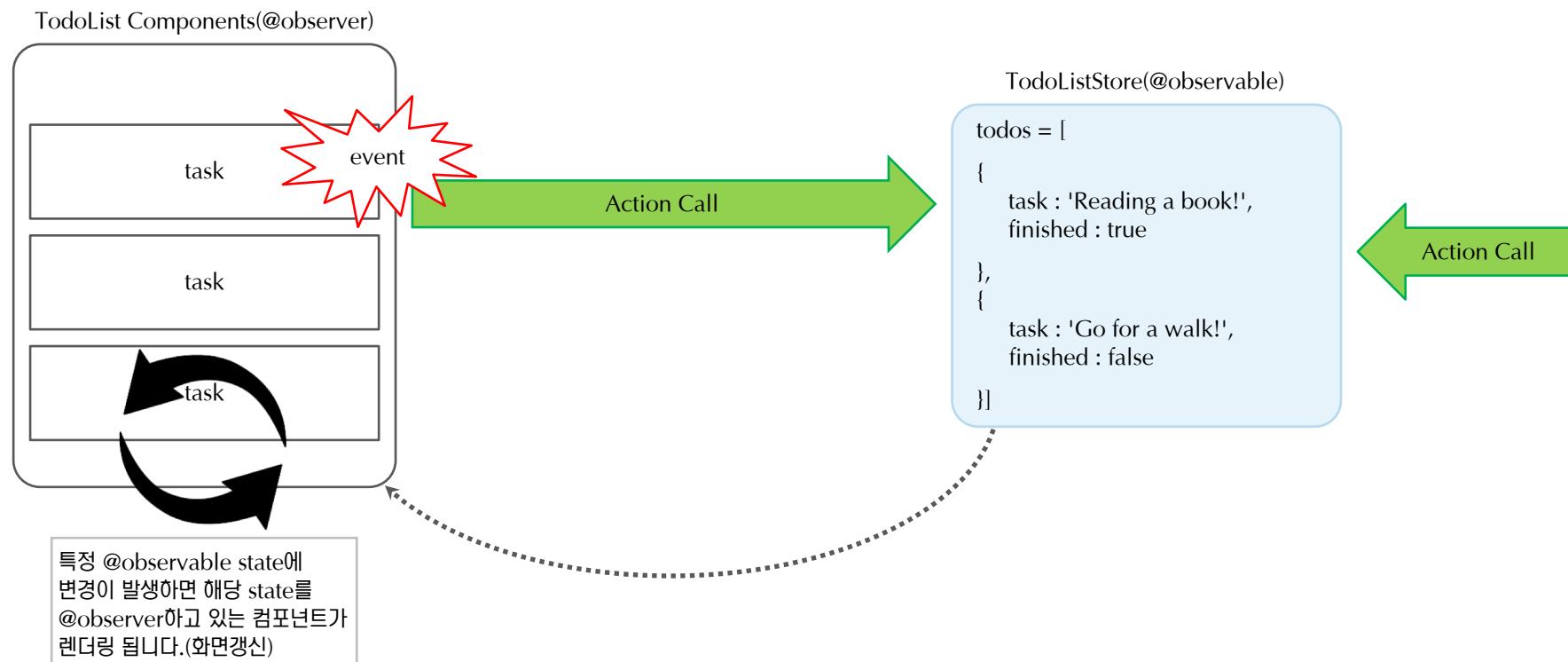
  @computed
  get unfinishedTodoCount(){
    return this.todos.filter(todo => !todo.finished).length;
  }

  @action
  addTodo(task){
    this.todos.push({
      task : task,
      finished: false
    });
  }
}

export default TodoListStore;
```

## 6.3 MobX 개요(4/4)

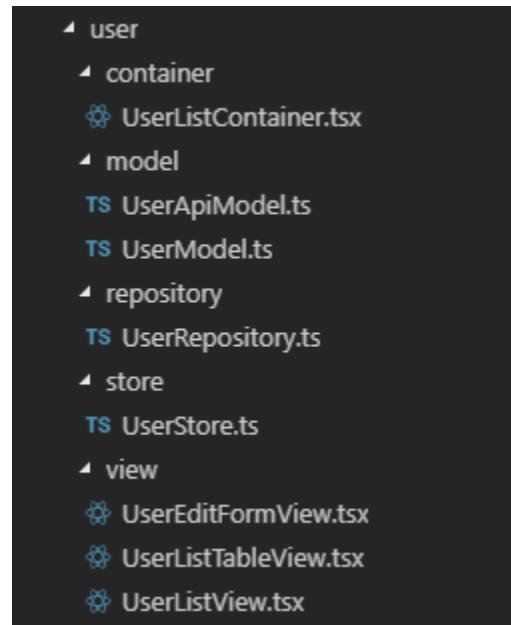
- ✓ `@observer`는 store를 통해서 state를 관리하는 `React.Component`에 적용합니다.
- ✓ `@observer`가 적용된 `React.Component`는 관련 observable state가 변경되면 렌더링을 수행합니다.
- ✓ MobX는 다수의 store를 구성하는 것이 가능하며 `@inject`를 이용해 해당 `@observer` 컴포넌트의 store를 주입합니다.
- ✓ 이외에도 MobX 라이브러리는 `@autorun`, `@transaction`과 같은 다양한 API를 제공합니다.



## 6.4 MobX의 적용[1/3]

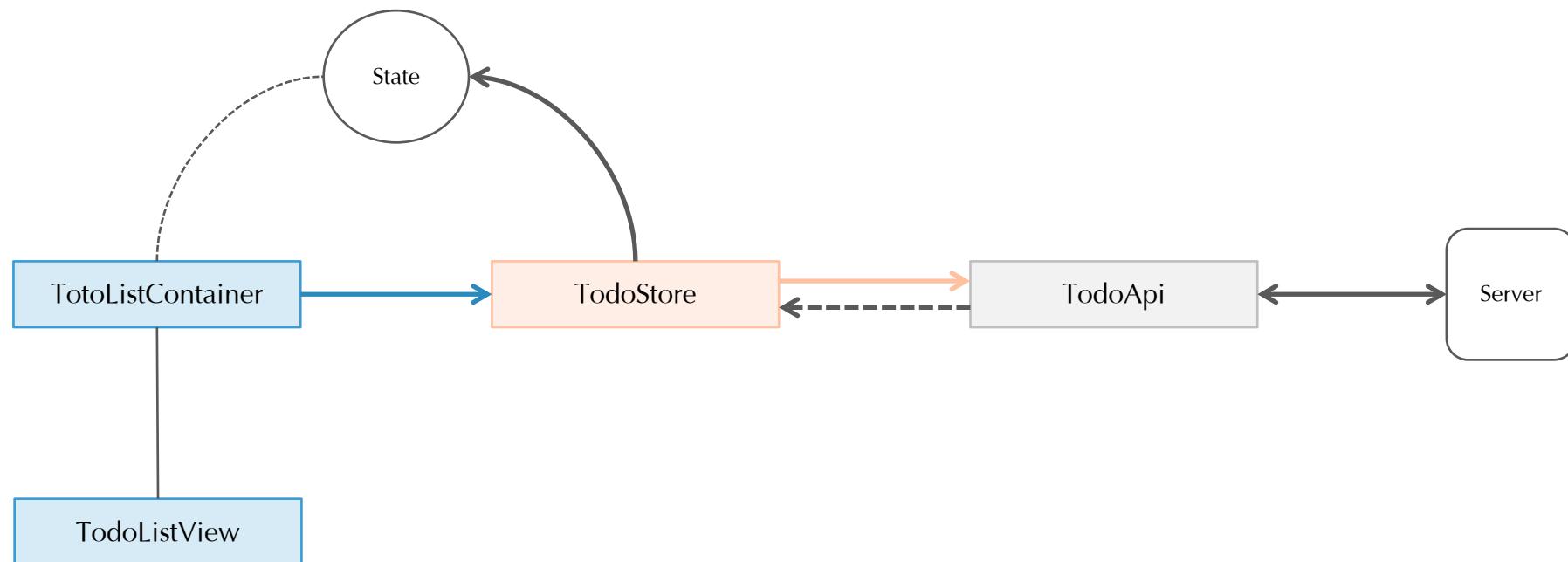
✓ React와 MobX를 통한 UI 구성은 다음과 같은 패키지의 구성을 갖습니다.

- container : React Component로 구성하며 store와 React Component를 연결하는 역할을 담당합니다.
- view : 순수 React Component로 구성하며 container에 포함됩니다.
- repository(or api) : 서버와 통신을 담당하는 클래스로 구성합니다.
- store : 전역 state를 관리하는 Store 클래스로 구성합니다.
- model : 서버의 model과 view model의 전환을 담당합니다.



## 6.4 MobX의 적용[2/3]

- ✓ container로 정의한 React 컴포넌트의 state는 store를 통해 관리합니다.
- ✓ store는 @action 메소드를 포함하며 이 메소드 호출을 통해 state를 제어합니다.
- ✓ store의 state 데이터에 변경이 일어나면 해당 state와 연결된 React Component는 다시 렌더링 됩니다.
- ✓ api는 서버와 통신을 담당하며 axios.js와 같은 서드-파티 라이브러리(Third-party Lib.)를 사용합니다.



## 6.4 MobX의 적용[3/3]

- ✓ MobX는 ES7의 스펙인 데코레이터(Decorator)를 지원하며 함수 형태의 사용도 가능합니다.
- ✓ 전역 형태의 Store를 구성하기 위해서는 Store 클래스를 정의 합니다.
- ✓ 정의한 Store 클래스에 사용하고자 하는 state를 정의하고 초기화 합니다.

```
class UserStore {  
    @observable  
    users = [];  
  
    @observable  
    user = {};  
  
    constructor(){  
        autoBind(this);  
    }  
}
```

```
class UserStore {  
    @observable  
    users: UserModel[];  
    @observable  
    user: UserModel;  
  
    constructor() {  
        autoBind(this)  
    }  
}
```

## 6.5 MobX 예제 실습

