



# Re:DOM

Starting Front Dev with RE:DOM

으로 시작하는 프론트 개발

2020-01-18

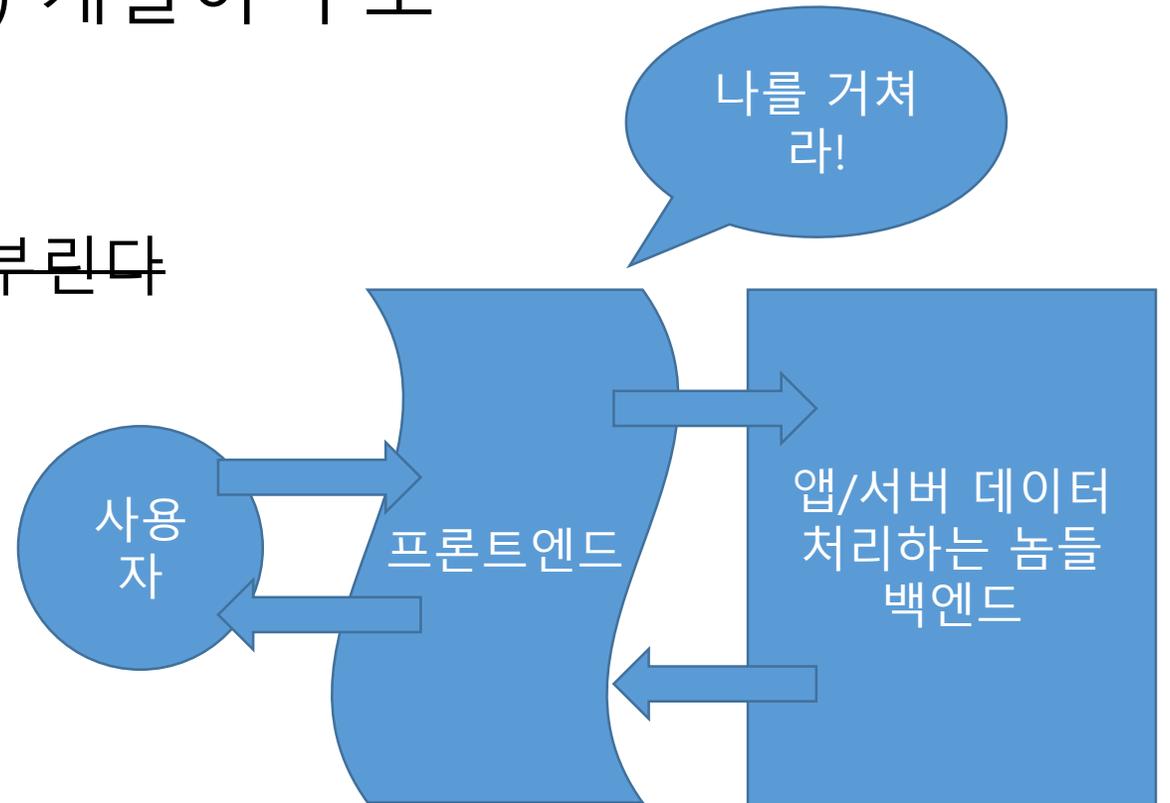
이인용@UPnL

# 발표자 소개

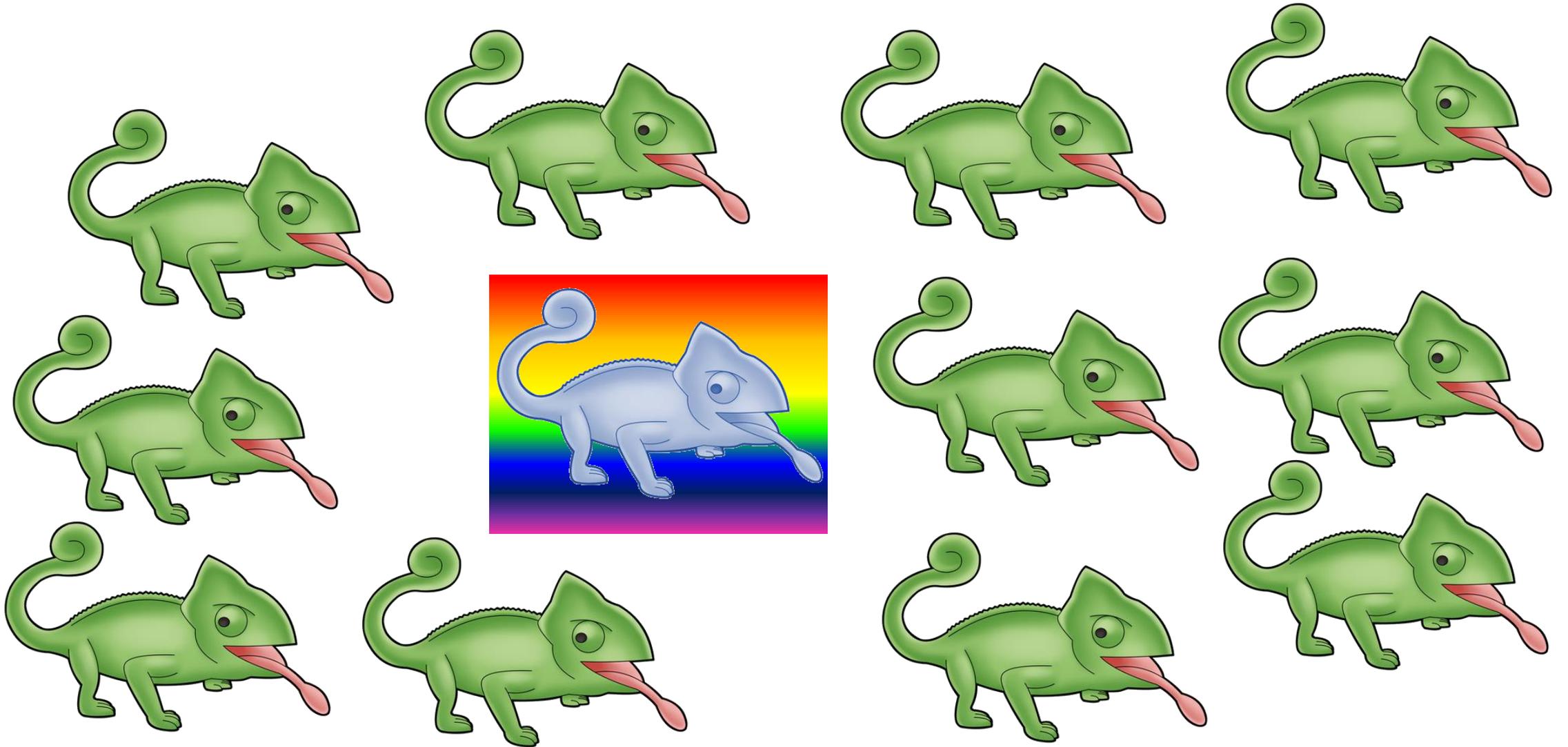
- 이인용
- 컴퓨터공학부 14학번
- 14.09 정회원
- 유피넬 홈페이지 개발자(프론트 담당)
- 전전전전 와장
- 19.09 소집해제
  
- 리제로는 뭔지 모름

# 프론트엔드!

- 프론트엔드란 무엇인가
- 주로 사용자의 인터페이스(UI) 개발이 주로
- 사용자랑 붙어있다?
  - 예쁘면 좋다.....
  - 고객들을 상대한다 → ~~진상을 부린다~~
  - 쓰기 편해야 한다
  - 쉬워야 한다
  - .....



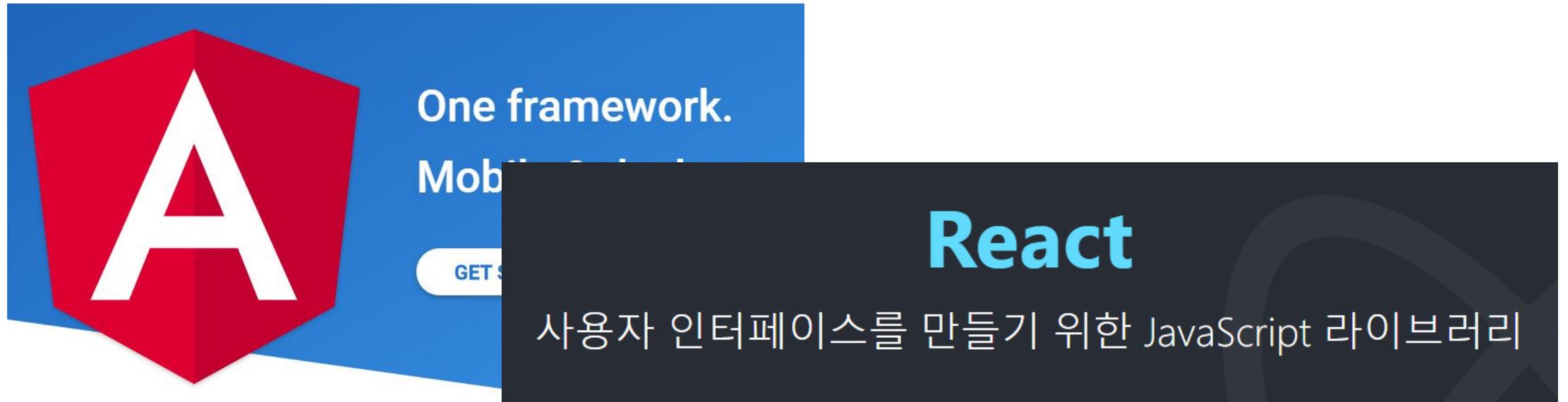
# 카멜레온을 골라보자



# 프레임워크란?

## Vue.js가 무엇인가요?

Vue(/vju:/ 로 발음, **view** 와 발음이 같습니다.)는 사용자 인터페이스를 만들기 위한 **프로그레시브 프레임워크** 입니다. 다른 단일형 프레임워크와 달리 Vue는 점진적으로 채택할 수 있도록 설계하였습니다.



# 프레임워크란?

- 제어의 역전을 활용하여 사용자가 작성할 코드를 줄여준다
- 제어의 역전?
  - 몰라 찾아봐 용어는 중요하지 않아
  - 내가 짠 코드를 다른 프로그램/함수에게 줘서 다른 프로그램이 내 코드 돌리게 하는 거
  - 아무 코드나 주고 돌리게는 안 해주고, 정해진 인터페이스 등을 상속해야 돌리게 해줌
    - Runnable 넣어주면 돌려줄게
    - `addEventListener('click', clickHandler);` 클릭오면 돌려주겠음

# 프레임워크 예시

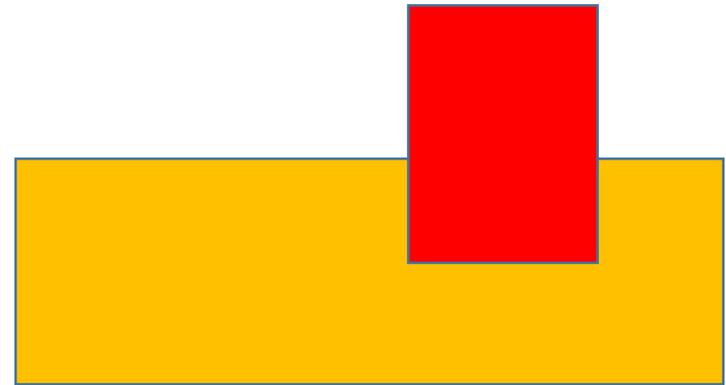
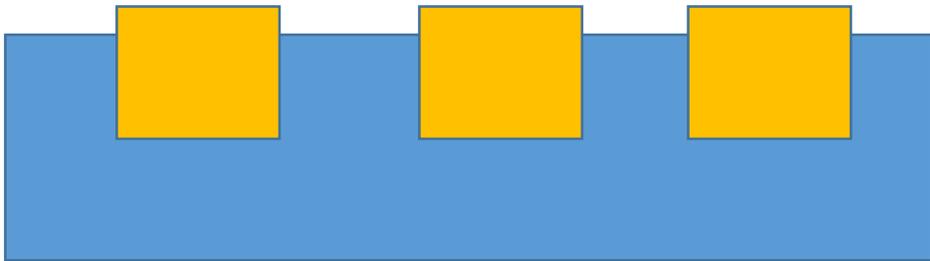
HTTP 서버를 작성하려면  
포트 정해서 TCP 바인드  
클라이언트 소켓 기다리고  
연결되면  
HTTP 프로토콜 파싱하고  
세션도 가져오고  
URL에 따라서  
할 일을 한다  
뭐 리턴하고  
소켓 닫음

HTTP 서버 프레임워크로는  
포트 정해서 TCP 바인드  
클라이언트 소켓 기다리고  
연결되면  
HTTP 프로토콜 파싱하고  
세션도 가져오고  
URL에 따라서  
할 일을 한다  
뭐 리턴하고  
소켓 닫음

계속 나오니까 우리가 해줄게요!

# 프레임워크와 라이브러리의 차이

- 내 코드 실행해주세요
- 내 코드를 큰 틀(프레임워크)에 끼워넣음
- 주객전도
- 언제 호출되는지?는 재가 정함
- 단순 함수 모음
- 결과 내놔라
- 호출 시점은 내 마음
- 함수니까...



# 프레임워크에 등록된 코드의 예시

```
@homepage.route("/next_season", methods=['POST'])
def next_season():
    if not session.__contains__('user_info'):
        return redirect(url_for('.login'))

    elif session['user_info'][0] == 'admin':
        seasons = [u"겨울 ", u"봄 ", u"여름 ", u"가을 "]
        season = dao.query(Admin).filter_by(name='season').first()
        index = seasons.index(season.semester)
        if index == 0:
            season.year = season.year + 1
            season.semester = seasons[(index + 1) % 4]
            dao.commit()
            return redirect(url_for('.admin'))

    else:
        return redirect(url_for('.index'))
```

다음 시즌으로 가는 "그" 소스 코드

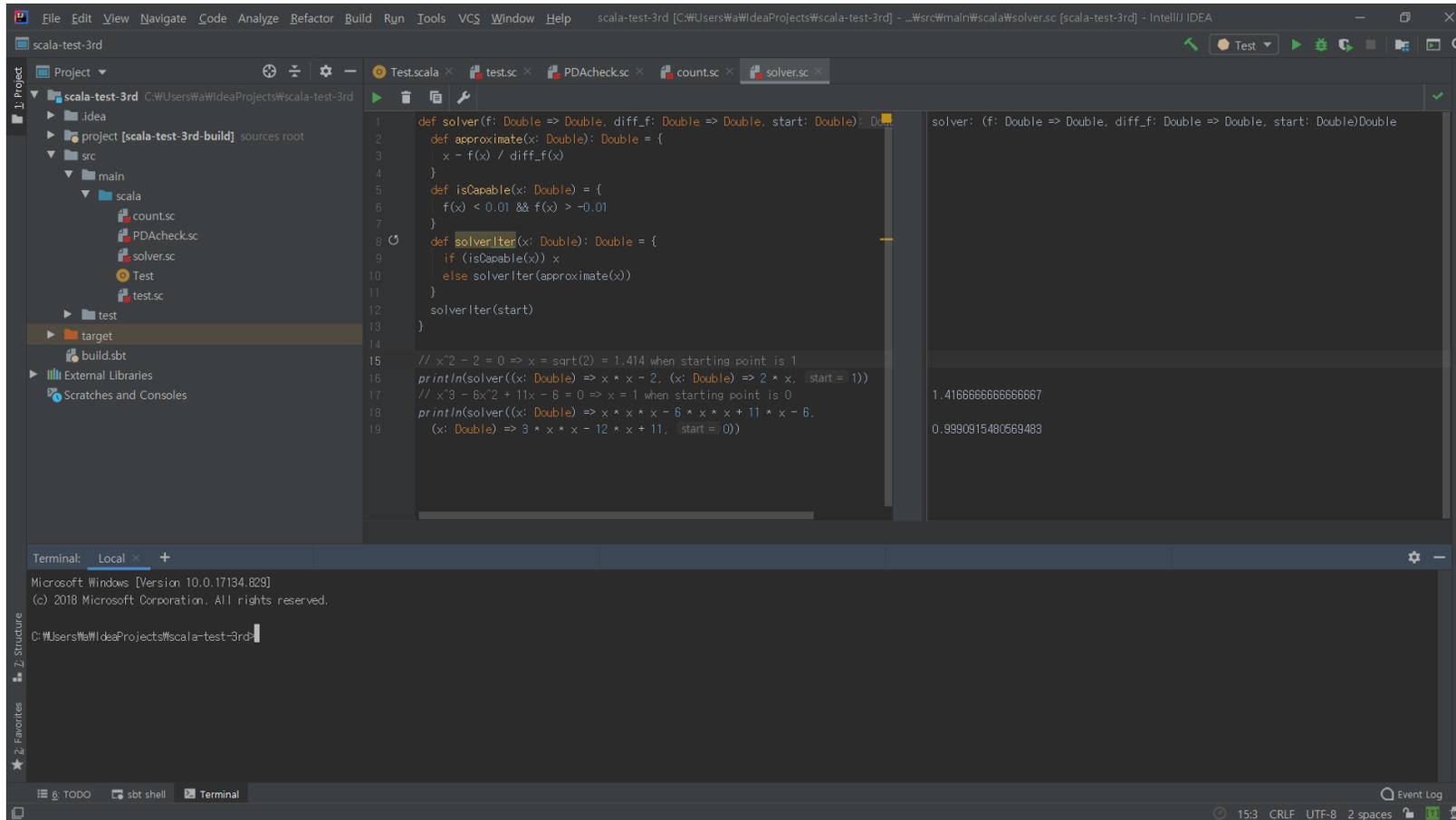
# 프론트엔드 개발이 하고 싶어요!

- 뭘 배워야 하나요?
  - 몰라
  - 개념 원리
  - HTML, CSS, Javascript
- 사실 뭘 만들고 싶느냐가 주가 되겠지만
- 발표가 삼천포로 빠지니까 대충 알려주고 넘어감

# 프론트엔드 구성 요소

- HTML
  - 뼈대, xml, `<h1>큰 제목</h1>` `<button>버튼</button>` `<input/>`
- CSS
  - `button { width: 80px; height: 30px; background: red; }`
- Javascript
  - `button.onclick = function (event) { console.log('버튼 눌림'); }`

# 오늘 만들 것



# 오늘 만들 것

- IDE를 만들어보자
- 뱅이고 IDE같이 여러 패널이 있고, 윈도우가 있고, 서로 옮길 수 있는 것을 만들기

# 뭐부터 해야할까?

- 레이아웃부터 짜봐요!
- 디자인 모양을 CSS로 만들어봐요!



# 데이터 구조부터 짜라!

- 레이아웃 짜거나 CSS를 먹이는 건
  - 그런 일을 하는 사람이 있고(?) 퍼블리셔?라고 하던가
  - 아랫사람을 시키든가(??)
    - 물론 없으면 해야함ㅋㅋ
  - 디자인 라이브러리가 해주기도 하고
- 아무튼 일단 신경 쓰지말고
- 프론트엔드라고 들입다 HTML부터 짜지 말고 데이터 구조를 짜보세요
  - 뭐 만들지 작게 만들어보는 건 해야할 수도...

# 데이터 구조부터 짜라!

- 헤더 - 뭐 둘 수 없음
- 사이드바, 푸터
  - 어떤 화면을 넣어둘 수 있음, 열고 접을 수 있음, 사이즈 조절 가능
- 화면
  - 화면 안에 패널 여러 개가 탭으로 구성될 수 있음, 사이즈 조절 가능
- 패널
  - 화면 안의 도구



사이드바 예시

# 데이터 구조부터 짜라!

```
{
  right: {
    isFold: false, // 열렸니?
    windowHeights: [ // 고민 필요
      300,
    ],
  },
  windows: [
    { type: 'window', name: 'window1',
      selectedPannelIndex: 0, // 무슨 패널 선택됐는지
      pannels: [
        { type: 'panel', name: 'pannel1' },
        { type: 'panel', name: 'pannel2' }
      ]
    },
    { type: 'window', name: 'window2',
      selectedPannelIndex: 0,
      pannels: [
        { type: 'panel', name: 'pannel3' }
      ]
    }
  ]
}
```

# 이 데이터를 변경하는 걸 짜야 함

- 화면 구조 변경하는 건 뭐가 있을까요?
  - 다른 패널 선택
  - 패널 → 다른 윈도우에 놓기
  - 패널 → 제 3의 윈도우로 놓기
  - 패널 → 다른 사이드바나 푸터로 옮기기
  - 열기 접기
  - 각 컴포넌트 사이킷 조절

# 데이터를 변경하는 방법을 기술해라!

- 일일이 함수로 만들어요
- 아 귀찮아
- 아랫사람 시키자
- 없잖아?
- T

# 데이터를 변경하는 방법을 기술해라!

```
glggozz@kari: ~  
68 class WindowWrapper {  
69   constructor(name) {  
70     this.name = name;  
71     this.isFold = false;  
72     this.windowSizes = [];  
73     this.windows = [ ];  
74   }  
75  
76   addPannel(windowIndex, pannel) {  
77     if (this.windows.length <= windowIndex) {  
78       this.windows.push(new _Window());  
79     }  
80     this.windows[Math.min(windowIndex, this.windows.length - 1)].addPannel(pannel);  
81   }  
82  
83   removePannel(windowIndex, pannelIndex) {  
84     this.windows[windowIndex].removePannel(pannelIndex);  
85   }  
86 };  
87  
88 class _Window {  
89   constructor(name) {  
90     this.name = name;  
91     this.selectedPannelIndex = null;  
92     this.pannels = [ ];  
93   }  
94  
95   addPannel(pannel) {  
96     this.pannels.push(pannel);  
97     this.selectedPannelIndex = this.pannels.length - 1;  
98   }  
99  
100  removePannel(pannelIndex) {  
101    return this.pannels.splice(from, 1)[0];  
102  }  
103  
104  swap(from, to) {  
105    this.pannels.splice(to, 0, this.removePannel(from));  
106    this.selectedPannelIndex = to;  
107  }  
108  
109  select(index) {  
110    this.selectedPannelIndex = index;  
111  }  
112 }  
113  
114 class Pannel {  
115   constructor(name, content) {  
116     this.name = name;  
117     this.content = content;  
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 }  
1001 }  
1002 }  
1003 }  
1004 }  
1005 }  
1006 }  
1007 }  
1008 }  
1009 }  
1010 }  
1011 }  
1012 }  
1013 }  
1014 }  
1015 }  
1016 }  
1017 }  
1018 }  
1019 }  
1020 }  
1021 }  
1022 }  
1023 }  
1024 }  
1025 }  
1026 }  
1027 }  
1028 }  
1029 }  
1030 }  
1031 }  
1032 }  
1033 }  
1034 }  
1035 }  
1036 }  
1037 }  
1038 }  
1039 }  
1040 }  
1041 }  
1042 }  
1043 }  
1044 }  
1045 }  
1046 }  
1047 }  
1048 }  
1049 }  
1050 }  
1051 }  
1052 }  
1053 }  
1054 }  
1055 }  
1056 }  
1057 }  
1058 }  
1059 }  
1060 }  
1061 }  
1062 }  
1063 }  
1064 }  
1065 }  
1066 }  
1067 }  
1068 }  
1069 }  
1070 }  
1071 }  
1072 }  
1073 }  
1074 }  
1075 }  
1076 }  
1077 }  
1078 }  
1079 }  
1080 }  
1081 }  
1082 }  
1083 }  
1084 }  
1085 }  
1086 }  
1087 }  
1088 }  
1089 }  
1090 }  
1091 }  
1092 }  
1093 }  
1094 }  
1095 }  
1096 }  
1097 }  
1098 }  
1099 }  
1100 }  
1101 }  
1102 }  
1103 }  
1104 }  
1105 }  
1106 }  
1107 }  
1108 }  
1109 }  
1110 }  
1111 }  
1112 }  
1113 }  
1114 }  
1115 }  
1116 }  
1117 }  
1118 }  
1119 }  
1120 }  
1121 }  
1122 }  
1123 }  
1124 }  
1125 }  
1126 }  
1127 }  
1128 }  
1129 }  
1130 }  
1131 }  
1132 }  
1133 }  
1134 }  
1135 }  
1136 }  
1137 }  
1138 }  
1139 }  
1140 }  
1141 }  
1142 }  
1143 }  
1144 }  
1145 }  
1146 }  
1147 }  
1148 }  
1149 }  
1150 }  
1151 }  
1152 }  
1153 }  
1154 }  
1155 }  
1156 }  
1157 }  
1158 }  
1159 }  
1160 }  
1161 }  
1162 }  
1163 }  
1164 }  
1165 }  
1166 }  
1167 }  
1168 }  
1169 }  
1170 }  
1171 }  
1172 }  
1173 }  
1174 }  
1175 }  
1176 }  
1177 }  
1178 }  
1179 }  
1180 }  
1181 }  
1182 }  
1183 }  
1184 }  
1185 }  
1186 }  
1187 }  
1188 }  
1189 }  
1190 }  
1191 }  
1192 }  
1193 }  
1194 }  
1195 }  
1196 }  
1197 }  
1198 }  
1199 }  
1200 }  
1201 }  
1202 }  
1203 }  
1204 }  
1205 }  
1206 }  
1207 }  
1208 }  
1209 }  
1210 }  
1211 }  
1212 }  
1213 }  
1214 }  
1215 }  
1216 }  
1217 }  
1218 }  
1219 }  
1220 }  
1221 }  
1222 }  
1223 }  
1224 }  
1225 }  
1226 }  
1227 }  
1228 }  
1229 }  
1230 }  
1231 }  
1232 }  
1233 }  
1234 }  
1235 }  
1236 }  
1237 }  
1238 }  
1239 }  
1240 }  
1241 }  
1242 }  
1243 }  
1244 }  
1245 }  
1246 }  
1247 }  
1248 }  
1249 }  
1250 }  
1251 }  
1252 }  
1253 }  
1254 }  
1255 }  
1256 }  
1257 }  
1258 }  
1259 }  
1260 }  
1261 }  
1262 }  
1263 }  
1264 }  
1265 }  
1266 }  
1267 }  
1268 }  
1269 }  
1270 }  
1271 }  
1272 }  
1273 }  
1274 }  
1275 }  
1276 }  
1277 }  
1278 }  
1279 }  
1280 }  
1281 }  
1282 }  
1283 }  
1284 }  
1285 }  
1286 }  
1287 }  
1288 }  
1289 }  
1290 }  
1291 }  
1292 }  
1293 }  
1294 }  
1295 }  
1296 }  
1297 }  
1298 }  
1299 }  
1300 }  
1301 }  
1302 }  
1303 }  
1304 }  
1305 }  
1306 }  
1307 }  
1308 }  
1309 }  
1310 }  
1311 }  
1312 }  
1313 }  
1314 }  
1315 }  
1316 }  
1317 }  
1318 }  
1319 }  
1320 }  
1321 }  
1322 }  
1323 }  
1324 }  
1325 }  
1326 }  
1327 }  
1328 }  
1329 }  
1330 }  
1331 }  
1332 }  
1333 }  
1334 }  
1335 }  
1336 }  
1337 }  
1338 }  
1339 }  
1340 }  
1341 }  
1342 }  
1343 }  
1344 }  
1345 }  
1346 }  
1347 }  
1348 }  
1349 }  
1350 }  
1351 }  
1352 }  
1353 }  
1354 }  
1355 }  
1356 }  
1357 }  
1358 }  
1359 }  
1360 }  
1361 }  
1362 }  
1363 }  
1364 }  
1365 }  
1366 }  
1367 }  
1368 }  
1369 }  
1370 }  
1371 }  
1372 }  
1373 }  
1374 }  
1375 }  
1376 }  
1377 }  
1378 }  
1379 }  
1380 }  
1381 }  
1382 }  
1383 }  
1384 }  
1385 }  
1386 }  
1387 }  
1388 }  
1389 }  
1390 }  
1391 }  
1392 }  
1393 }  
1394 }  
1395 }  
1396 }  
1397 }  
1398 }  
1399 }  
1400 }  
1401 }  
1402 }  
1403 }  
1404 }  
1405 }  
1406 }  
1407 }  
1408 }  
1409 }  
1410 }  
1411 }  
1412 }  
1413 }  
1414 }  
1415 }  
1416 }  
1417 }  
1418 }  
1419 }  
1420 }  
1421 }  
1422 }  
1423 }  
1424 }  
1425 }  
1426 }  
1427 }  
1428 }  
1429 }  
1430 }  
1431 }  
1432 }  
1433 }  
1434 }  
1435 }  
1436 }  
1437 }  
1438 }  
1439 }  
1440 }  
1441 }  
1442 }  
1443 }  
1444 }  
1445 }  
1446 }  
1447 }  
1448 }  
1449 }  
1450 }  
1451 }  
1452 }  
1453 }  
1454 }  
1455 }  
1456 }  
1457 }  
1458 }  
1459 }  
1460 }  
1461 }  
1462 }  
1463 }  
1464 }  
1465 }  
1466 }  
1467 }  
1468 }  
1469 }  
1470 }  
1471 }  
1472 }  
1473 }  
1474 }  
1475 }  
1476 }  
1477 }  
1478 }  
1479 }  
1480 }  
1481 }  
1482 }  
1483 }  
1484 }  
1485 }  
1486 }  
1487 }  
1488 }  
1489 }  
1490 }  
1491 }  
1492 }  
1493 }  
1494 }  
1495 }  
1496 }  
1497 }  
1498 }  
1499 }  
1500 }  
1501 }  
1502 }  
1503 }  
1504 }  
1505 }  
1506 }  
1507 }  
1508 }  
1509 }  
1510 }  
1511 }  
1512 }  
1513 }  
1514 }  
1515 }  
1516 }  
1517 }  
1518 }  
1519 }  
1520 }  
1521 }  
1522 }  
1523 }  
1524 }  
1525 }  
1526 }  
1527 }  
1528 }  
1529 }  
1530 }  
1531 }  
1532 }  
1533 }  
1534 }  
1535 }  
1536 }  
1537 }  
1538 }  
1539 }  
1540 }  
1541 }  
1542 }  
1543 }  
1544 }  
1545 }  
1546 }  
1547 }  
1548 }  
1549 }  
1550 }  
1551 }  
1552 }  
1553 }  
1554 }  
1555 }  
1556 }  
1557 }  
1558 }  
1559 }  
1560 }  
1561 }  
1562 }  
1563 }  
1564 }  
1565 }  
1566 }  
1567 }  
1568 }  
1569 }  
1570 }  
1571 }  
1572 }  
1573 }  
1574 }  
1575 }  
1576 }  
1577 }  
1578 }  
1579 }  
1580 }  
1581 }  
1582 }  
1583 }  
1584 }  
1585 }  
1586 }  
1587 }  
1588 }  
1589 }  
1590 }  
1591 }  
1592 }  
1593 }  
1594 }  
1595 }  
1596 }  
1597 }  
1598 }  
1599 }  
1600 }  
1601 }  
1602 }  
1603 }  
1604 }  
1605 }  
1606 }  
1607 }  
1608 }  
1609 }  
1610 }  
1611 }  
1612 }  
1613 }  
1614 }  
1615 }  
1616 }  
1617 }  
1618 }  
1619 }  
1620 }  
1621 }  
1622 }  
1623 }  
1624 }  
1625 }  
1626 }  
1627 }  
1628 }  
1629 }  
1630 }  
1631 }  
1632 }  
1633 }  
1634 }  
1635 }  
1636 }  
1637 }  
1638 }  
1639 }  
1640 }  
1641 }  
1642 }  
1643 }  
1644 }  
1645 }  
1646 }  
1647 }  
1648 }  
1649 }  
1650 }  
1651 }  
1652 }  
1653 }  
1654 }  
1655 }  
1656 }  
1657 }  
1658 }  
1659 }  
1660 }  
1661 }  
1662 }  
1663 }  
1664 }  
1665 }  
1666 }  
1667 }  
1668 }  
1669 }  
1670 }  
1671 }  
1672 }  
1673 }  
1674 }  
1675 }  
1676 }  
1677 }  
1678 }  
1679 }  
1680 }  
1681 }  
1682 }  
1683 }  
1684 }  
1685 }  
1686 }  
1687 }  
1688 }  
1689 }  
1690 }  
1691 }  
1692 }  
1693 }  
1694 }  
1695 }  
1696 }  
16
```

# 시간이 없어ㅠㅠ TODO 리스트 하자

- 아까처럼 해주자
- 데이터 정의
  - { 할 일들: [ { 이름, 했음? }, { 이름, 했음? } ] }
- 데이터 변경
  - n번째 했음, 안 했음 변경, 이름 변경, 할 일 추가

# Todo 리스트 데이터 정의, 변경

```
class Todo {
  constructor(name) {
    this.name = name;
    this.done = false;
  }
  isDone() {
    return this.done;
  }
  toggleDone() {
    this.done = !this.done;
  }
  changeName(name) {
    this.name = name;
  }
}

class TodoList {
  constructor() {
    this.todos = [];
  }
  addTodo(name) {
    this.todos.push(new Todo(name));
  }
  removeTodo(index) {
    this.todos.splice(index, 1);
  }
}
```

```
var todoList = new TodoList();
function add(todoList, name) {
  todoList.addTodo(name);
}
function remove(todoList, index) {
  todoList.remove(index);
}
function done(todoList, index) {
  todoList.getTodo(index).toggleDone();
}
```

# 검증

```
var todoList = new TodoList();  
add(todoList, '발표자료 만들기');  
done(todoList, 0);  
remove(todoList, 0);
```

- 잘 됨

# 이제야!! UI를 만들 수 있어요

- 사실 UI는 만들었죠?
  - 함수 3개가 UI
  - 콘솔로 조작하면 됨
  - 일반 사람들이 쓰기 어려울 뿐
- UI는 껍데기일 뿐이다!
  - 반복 3번 하세요
- 불편하니까 GUI로 제공해주자

# RE:DOM으로 만들자

- RE:DOM



Develop web applications with 100% JavaScript and web standards. 🚀



**RE:DOM** is a tiny (2 KB) UI library by [Juha Lindstedt](#) and [contributors](#), which adds some useful helpers to create DOM elements and keeping them in sync with the data.

Because RE:DOM is so close to the metal and **doesn't use virtual dom**, it's actually **faster** and uses **less memory** than almost all virtual dom based libraries, including React ([benchmark](#)).

It's also easy to create **reusable components** with RE:DOM.

Another great benefit is, that you can use just **pure JavaScript**, so no complicated templating languages to learn and hassle with.

# RE:DOM - 기본적인 method

- method el
- el(tag이름, attr-object, 자식들) 온갖 것 넣기 가능

```
el('div#id',  
  el('span.class', '안뇽'),  
  el('input', { type: 'checkbox' } )  
); // 의 결과는 아래와 같이
```

```
<div id="id">  
  <span class="class">안뇽</span>  
  <input type="checkbox">  
</div>
```

# RE:DOM - component

- Node instance 를 el 프로퍼티로 가지는 Object
- el을 프로퍼티로 가지는 클래스를 만들어주자

```
function TodoView() {  
  this.el = el('span', '아');  
}  
var todoView = new TodoView();  
document.body.appendChild(todoView);
```

# RE:DOM - mount

- 리돔에도 약간의 프레임워크 끼가 있어서 라이프사이클이 존재 하긴 함
- append 를 직접하는 것보단 mount method를 사용해서 해주 면 좋다

```
mount(document.body, todoView);
```

# RE:DOM - list

- TodoList 처럼 데이터에 따라 동적으로 생성되는 것들을 위해 헬퍼로 list를 제공함. list 클래스는 update() 를 통해 data를 집어넣을 수 있으며 자세한 건 [도큐먼트 참조](#)
- 이 때 javascript의 assignment expression을 이용하여 재밌는 일을 할 수 있다

```
function TodoApp() {  
  this.el = el('div',  
    this.list = list('ul', TodoView), // this.list에 대입하면서 넘어감  
  )  
}
```

# RE:DOM - update

- list 두번째 인자로 넘겨준 컴포넌트 클래스에 update method 를 적당히 추가해주면 데이터를 갖고 업데이트 가능
- 이벤트는 dom을 알고 있으니까 직접 핸들러 추가

```
const { el, list, mount } = redom;

function draw() {
  app.todoListView.update(todoList.getTodos());
}

function TodoView() {
  this.el = el('li',
    el('label',
      this.todoNameView = el('span.name'),
      this.checkbox = el('input', { type: 'checkbox' })),
    ),
  this.removeButton = el('button', '삭제 ');
};

this.update = (todo, index) => {
  this.todo = todo;
  this.index = index;

  this.todoNameView.textContent = todo.name;
};

this.checkbox.onchange = event => {
  done(todoList, this.index);
};

this.removeButton.onclick = event => {
  remove(todoList, this.index);
  draw();
};
}
```

```
const todoList = new TodoList();
function TodoApp() {
  this.el = el('div',
    el('h1', 'Todo List'),
    this.form = el('form',
      this.todoNameInput = el('input'),
      this.makeTodoButton = el('button', '추가 ')
    ),
    this.todoListView = list('ul', TodoView)
  );

  this.form.onsubmit = event => {
    event.preventDefault();
    if (this.todoNameInput.value) {
      add(todoList, this.todoNameInput.value);
      this.todoNameInput.value = '';
      draw();
    }
  };
}

const app = new TodoApp();
mount(document.body, app);
```

# 마무리

- UI는 껍데기이다
- 프론트엔드에서도 데이터 구조를 잡는 것이 중요하다!
- 아까 만들었던 데이터 구조들은 이미 프레임워크들에 녹아있다
- Redux
  - Store -- 데이터 정의
  - Action & Reducer -- 데이터 조작 및 처리
- 처럼...
- 그래서... RE:DOM 써보실래요?