인터랙션스튜디오:
웹인터랙션프로그래밍

웹인터랙션프로그래밍

odododog12@naver.com
010 9215 5512

https://youtu.be/x0_o3YHcFa8?si=DUeZZp6jEiSeUZV9

timeline


2번

```
<section>
  <h2>1번</h2>
</section>
<section>
  <h2>2번</h2>
</section>
<section>
  <h2>3번</h2>
</section>
<section>
  <h2>4번</h2>
</section>
```

```css
section {
    height:100vh;
}
section:nth-of-type(1){
    background: #2facff;
}
section:nth-of-type(2){
    background: rosybrown;
    animation: pung 5s ease;
    /*animation-timeline:scroll();*/
    animation-timeline:view();
}
section:nth-of-type(3){
    background: steelblue;
    animation: bg 5s ease;
    animation-timeline:view();
}
section:nth-of-type(4){
    background: yellowgreen;
}
```

```css
@keyframes pung {
    from {
        opacity: 0;
        transform:scale(0.5);
    }
    to {
        opacity: 1;
        transform:scale(1);
    }
}
@keyframes bg {
    from {
        background: steelblue;
    }
    to {
        background: hotpink;
    }
}
```

돌아가는 텍스트

안녕나는신원

```
<div class="txt-box">
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
    <div class="area"></div>
</div>
```

```css
body {
    display:flex;
    justify-content: center;
    align-items: center;
    perspective: 500px;
}
.txt-box {
    display: flex;
}
.area {
    position:relative;
    width:30px;
    height:60px;
    overflow:hidden;
    animation: rotate 6s var(--delay) infinite ease-in-out;
}
.area::before {
    position:absolute;
    left:var(--left);
    content:'안녕나는신원';
    font-size:50px;
    width:max-content;
    color:hsl(var(--hue), 75%, 75%);
}
```

```css
@keyframes rotate {
    0%, 10% {
        transform: rotateX(0deg);
    }
    90%, 100% {
        transform: rotateX(720deg);
    }
}
```

```css
.area:nth-child(1) {
  --left: 0px;
  --hue: 0;
  --delay: 0s;
}

.area:nth-child(2) {
  --left: -30px;
  --hue: 30;
  --delay: 0.1s;
}

.area:nth-child(3) {
  --left: -60px;
  --hue: 60;
  --delay: 0.2s;
}

.area:nth-child(4) {
  --left: -90px;
  --hue: 90;
  --delay: 0.3s;
}
.area:nth-child(5) {
  --left: -120px;
  --hue: 120;
  --delay: 0.4s;
}
```

```css
.area:nth-child(6) {
  --left: -150px;
  --hue: 150;
  --delay: 0.5s;
}
.area:nth-child(7) {
  --left: -180px;
  --hue: 180;
  --delay: 0.6s;
}
.area:nth-child(8) {
  --left: -210px;
  --hue: 210;
  --delay: 0.7s;
}
.area:nth-child(9) {
  --left: -240px;
  --hue: 240;
  --delay: 0.8s;
}
.area:nth-child(10) {
  --left: -270px;
  --hue: 270;
  --delay: 0.9s;
}
```

CSS 정리하기

1. 선택자는 상위 선택자를 포함하여 3개 이상 작성되지 않게

선택자 개수 제한

선택자를 작성할 때 키 선택자(가장 오른쪽에 위치하며, 마지막에 작성된 선택자를 지칭합니다.)를 포함하여,

상위 선택자는 3개 이상 작성되지 않는 것을 권장합니다.

```
.select-1, .select-2, .select-3 {...}
section div span {...}
```

선택자 우선순위 확인하기

```html
<div class="box"></div>
<div class="box two"></div>
<div id="three" class="box"></div>
```

```css
div {
    width:100px;
    height:100px;
    background:red;
    margin:20px;
}
.box {
    background:aquamarine;
}
.two {
    background:pink;
}
#three {
    background:violet;
}
div.box {
    background:burlywood;
}
.box.two {
    background:cadetblue;
}
div#three {
    background:palevioletred;
}
.box#three {
    background: darkseagreen;
}
```

2. 전체 선택자를 사용 자제

브라우저는 선택자를 오른쪽에서 왼쪽으로 읽기 때문에 마지막에 전체 선택자를 사용한 경우 최초 문서 내의 모든 요소에 대응하고 그 이후 상위 클래스로 해당 요소를 선택하는지 확인하는 과정을 거치게 됩니다. 따라서 요소를 선택하는 CSS 규칙이 깊어지는 경우와 전체 선택자를 마지막 부분에 작성하는 경우 브라우저의 성능 저하가 발생하기 때문에 사용을 지양해야 합니다.

```
/* bad */
* {...}
.selector * {...}

/* better */
.selector * th {...}
```

3. 애트리뷰트 선택자의 사용

```
/* bad */
[type="text"] {...}
.label [type="checkbox"] {...}

/* better */
input[type="text"] {...}
.layer-pop[data-role="popup"] .header {...}
```

4. 아이디 선택자 최소한 사용

ID는 하나의 HTML 파일에 유일하게 존재해야 하는 값입니다. ID 선택자를 사용한 다는 것은 그 스타일을 한 요소에만 사용하게 된다는 것을 의미하게 됩니다. 이러한 구조는 스타일 규칙의 재 사용이 불가능하게 만들어 CSS 파일 사이즈를 크게 만들고, 유지 보수와 확장성에 불편함을 가져오게 됩니다.

5. 태그 선택자 규칙에 상위 선택자로 태그를 포함하지 말것

태그와 태그 선택자의 사용.

HTML 문서 수정에 따른 CSS 파일 수정을 최소화하기 위해서 태그 규칙에 상위 선택자로 태그 선택자를 사용하지 않습니다.

```
/* bad */
.notice-box li span {...}

/* better */
.notice-box .list .span {...}
```

6. 줄바꿈과 들여쓰기

여는 중괄호의 다음, 닫는 중괄호의 다음, 세미콜론의 다음에 줄 바꿈 하는 스타일을 권장합니다. 미디어 쿼리 및 vendor prefix 대한 속성 사용으로 속성을 한 줄에 작성을 하게 되면 코드의 가독성이 떨어져 유지 보수와 수정 작업에 불편함이 따르기 때문입니다. 또한 코드의 들여 쓰기는 중첩의 깊이에 따라 공백 2칸으로 설정합니다.

파일의 용량 증가 문제는 서비스에 반영하기 전 minify를 진행하여 배포함으로 문제가 없습니다.

```
html {
  position: relative;
  margin: 0 auto;
  ...
}

@media (width:320px) {
  .input-text.jumin {
    width: 46.1%;
  }
}
```

7. 속성 우선순위

아래의 나와 있는 순서로 작성하는 것을 권장합니다. csscomb 와 같은 빌드 스크립트를 이용하여 서비스 반영 전에 우선순위 및 코드를 정리할 수도 있습니다.

cssComb.js : https://github.com/csscomb/csscomb.js

cssComb 속성 우선 순위 설정 보기 :
https://github.com/csscomb/csscomb.js/tree/dev/config

1. display
2. list-style
3. position
4. float
5. clear
6. width / height
7. padding / margin
8. border / background
9. color / font
10. text-decoration
11. text-align / vertical-align
12. white-space
13. other text
14. content

8. CSS Reset은 필수

CSS 리셋을 하는 방법은 여러 가지가 있습니다. 용도에 따라 자신만의 CSS 리셋을 만들어 쓰기도 하지만, 기본적으로 CSS 리셋은 더 최적화되고 적은 CSS 속성을 작성할 수 있도록 도움을 주는 가장 기본적인 CSS 코드 작성입니다.
최근에는 브라우저간의 호환성 문제가 줄어들면서 이런 CSS 리셋의 중요도가 다소 불필요하게 보일 수도 있지만, 남용하지만 않으면 여전히 고전적이면서도 뛰어난 CSS 기법 중 하나입니다.

9. !important, inline style 사용 금지

특히 여러명이 공동으로 코드를 작성할 때나, 범용으로 공개를 하는 CSS에서는 기본적으로 사용하지 않는다는 생각을 가지고 CSS 코드를 작성해야 합니다.
!important는 CSS 속성을 상수로 고정하기 때문에 CSS 클래스 속성의 일부를 재정의 하거나 재활용하기 어렵게 만드는 거의 유일한 CSS의 추가 속성입니다.

!important를 사용하는 클래스는 되도록이면 한 군데 모아서 별도로 정의해 사용함으로써 항상 !important 로 인해 어떤 제약이 생겼는지를 쉽게 알 수 있도록 해야 합니다.
!important를 자주 사용하거나, !important가 없으면 제대로 동작하는 CSS 작성이 안된다면, 분명히 잘못된 CSS 작성 습관을 가지고 있는 것입니다.

CSS

정리하기

```
<div class="box-sort">
    <h3 style="width:100px;height:100px;background:red">저널 Journal Articles (최근)</h3>
    <ul style="text-align:center;font-size:20px;" class="journal"></ul>
</div>
```


```
.border-left {
    border-left:1px solid #ccc!important;
}
.border-left-0 {
    border-left:none!important;
}
.border-bottom-0 {
    border-bottom:0;
}
.mr-1 {
    margin-right:5px!important;
}
.mr-0 {
    margin-right:0!important;
}
.ml-1 {
    margin-left:5px!important;
}
.ml-0 {
    margin-left:0!important;
}
```

10. 상대 유닛으로 크기 값을 사용

em, rem, %, vw/vh와 같은 상대 크기 유닛을 사용해야 대상 기기의 화면 밀도, 또는 대상 기기에 무관하게 비교적 일정한 크기로 컨텐츠를 표시할 수 있고 가독성도 높일 수 있습니다.

```
body {
    font-size: 62.5%;
}
.block {
        padding: 1em;
    box-sizing: border-box;
}
p {
    font-size: 1.5em;
}
p.big {
        font-size: 3em;
}
p:first-letter {
    font-size: 3em;
}
```

제3자가 봤을때 **읽고 이해하기 쉬운** 코드

1. 위치와 방식

style head 안에

```
<head>
    <meta charset="UTF-8">
    <title>판판대로</title>

    <!--css-->
    <link href="https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@300;400;500;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/common_as_is.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/content_as_is.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/userdefined.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/connection.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/reset.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/notosans.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/common.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/jquery-ui.css?v221114" />
    <link rel="stylesheet" href="https://fanfandaero.kr/portal/css/sub.css?v221114" />
    <!-- 메인 css -->
    <link rel="stylesheet" type="text/css" href="https://fanfandaero.kr/portal/css/slick.css">
    <link rel="stylesheet" type="text/css" href="https://fanfandaero.kr/portal/css/slick-theme.css">
    <link rel="stylesheet" type="text/css" href="https://fanfandaero.kr/portal/css/main.css?v221114">
    <link rel="canonical" href="https://fanfandaero.kr/portal/main.do">
    <link rel="stylesheet" href="https://fanfandaero.kr/css/font-awesome.min.css?v201116" type="text/css"/>

    <!-- 신규 css -->
    <link rel="stylesheet" href="../assets/css/style.css">
    <link rel="stylesheet" href="../assets/css/mypage.css">

</head>
```

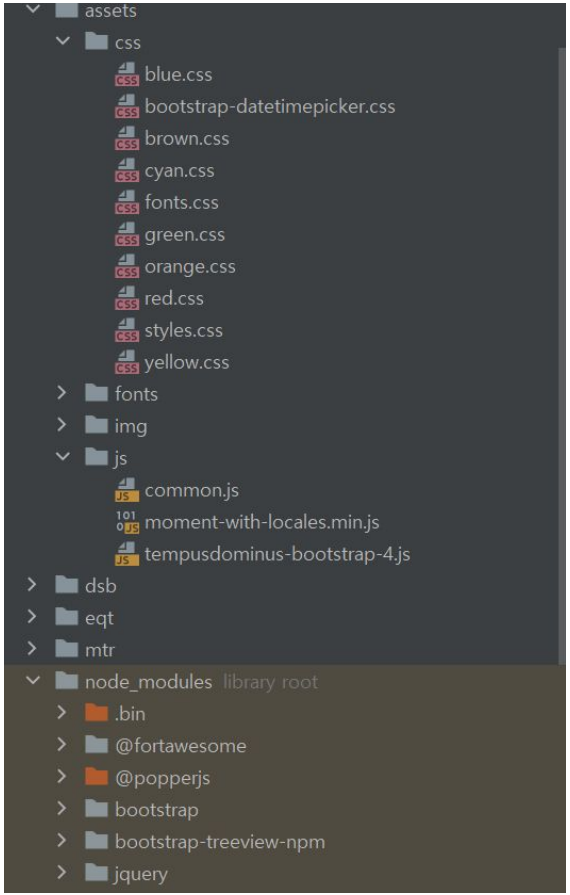
js 하단에(BUT 상황에 따라 다를 수 있다.)

```
<!--js library-->
<script src="../node_modules/jquery/dist/jquery.js"></script>
<script src="../node_modules/jquery-ui/dist/jquery-ui.js"></script>

<!--js-->
<script src="../assets/js/common.js"></script>

</body>
</html>
```

```
    <title>MACHBASE</title>

    <!-- css-->
    <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="../node_modules/jqueryui/jquery-ui.css">
    <link rel="stylesheet" href="../node_modules/bootstrap-treeview-npm/dist/bootstrap-treeview.min.
    <link rel="stylesheet" href="../assets/css/fonts.css">
    <link rel="stylesheet" href="../assets/css/styles.css">
    <link rel="stylesheet" href="dashboard.css">


    <!-- js-->
    <script src="../node_modules/jquery/dist/jquery.min.js"></script>
    <script src="../node_modules/jqueryui/jquery-ui.js"></script>
    <script src="../node_modules/bootstrap/dist/js/bootstrap.js"></script>
!--     <script src="../node_modules/bootstrap/js/dist/dropdown.js"></script>-->
    <script src="../node_modules/@fortawesome/fontawesome-free/js/all.js"></script>
    <script src="../node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="../node_modules/bootstrap-treeview-npm/dist/bootstrap-treeview.min.js"></script>
    <script src="../node_modules/@popperjs/core/dist/umd/popper.min.js"></script>

/head>
```

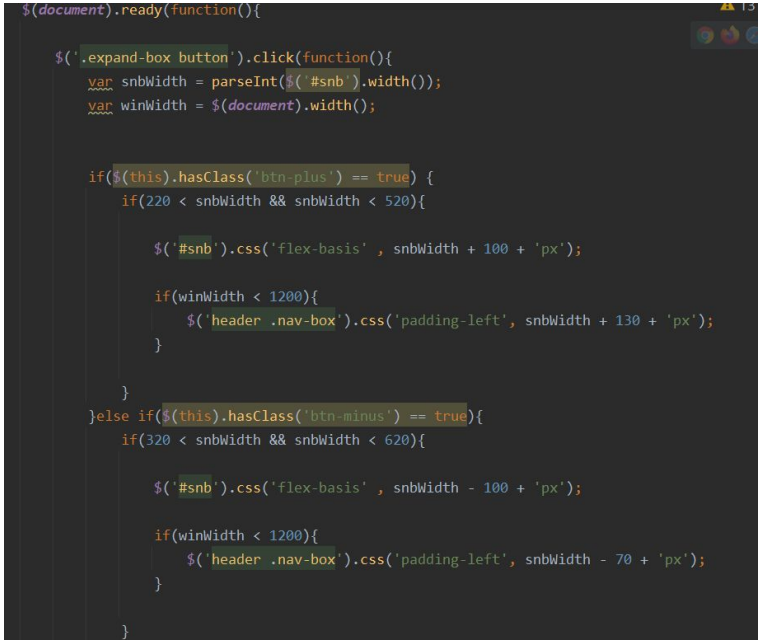
1. 위치와 방식

cdn보다는 로컬



2. 변수 선언

전역함수 vs 지역함수 잘 설정하기
알맞은 변수 선언
찾을 수 있는 범위내에 지정



```
$(document).ready(function(){

    $('.expand-box button').click(function(){
        var snbWidth = parseInt($('#snb').width());
        var winWidth = $(document).width();


        if($(this).hasClass('btn-plus') == true) {
            if(220 < snbWidth && snbWidth < 520){

                $('#snb').css('flex-basis' , snbWidth + 100 + 'px');

                if(winWidth < 1200){
                    $('header .nav-box').css('padding-left', snbWidth + 130 + 'px');
                }

            }
        }else if($(this).hasClass('btn-minus') == true){
            if(320 < snbWidth && snbWidth < 620){

                $('#snb').css('flex-basis' , snbWidth - 100 + 'px');

                if(winWidth < 1200){
                    $('header .nav-box').css('padding-left', snbWidth - 70 + 'px');
                }

            }
```

3. 변수와 함수를 만들 때 의도가 드러나도록 이름을 사용

```
let chk = 0;
const buttonTag = document.querySelector('button');
const pTag = document.querySelector('p');

buttonTag.addEventListener('click', function(){
    chk = chk + 1;
    pTag.innerText = chk;
});
```

의미없는 이름 사용하지 말것

```
let chk = 0;
const a = document.querySelector('button');
const b = document.querySelector('p');

a.addEventListener('click', function(){
    chk = chk + 1;
    b.innerText = chk;
});
```

태그를 그대로 사용하지 말것

```
let chk = 0;
const object = document.querySelector('button');
const p = document.querySelector('p');

object.addEventListener('click', function(){
    chk = chk + 1;
    p.innerText = chk;
});
```

4. 공통 코드

재사용 되는 코드는 쪼개고 공통 코드 사용하기

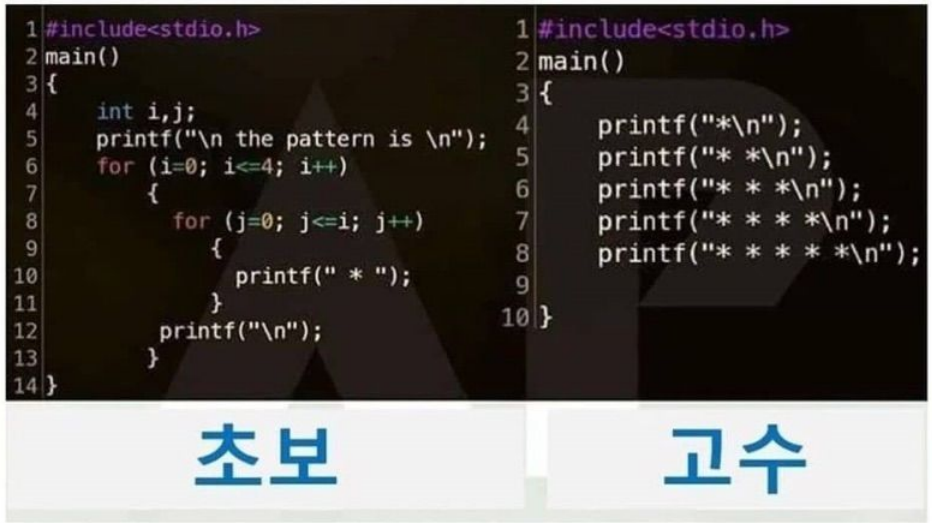
```
// POPUP REGISTER MODAL
$('.register-pop').click(function(){
    $('.popup.register').show();
    $('body').css('overflow', 'hidden');
});

$('.popup .btn-close').click(function(){
    $(this).parents('.popup').hide();
    $('body').css('overflow', 'auto');
});

// POPUP RELOAD
$('.reload-pop').click(function(){
    $('.popup.reload').show();
    $('body').css('overflow', 'hidden');
});

// POPUP LINK
$('.link-pop').click(function(){
    $('.popup.link').show();
    $('body').css('overflow', 'hidden');
});
```

```
$('.btn-pop').click(function(){

    if($(this).hasClass('register')){
        $('.popup.register').show();
    }else if($(this).hasClass('reload')){
        $('.popup.reload').show();
    }else if($(this).hasClass('link')){
        $('.popup.link').show();
    }
    $('body').css('overflow', 'hidden');
});

$('.popup .btn-close').click(function(){
    $(this).parents('.popup').hide();
    $('body').css('overflow', 'auto');
});
```

```
$('.btn-pop').click(function(){

    let btnClass = $(this).attr('class');
    btnClass = btnClass.split(' ')[1];

    $('.popup.' + btnClass).show();

});


$('.popup .btn-close').click(function(){
    $(this).parents('.popup').hide();
    $('body').css('overflow', 'auto');
});
```

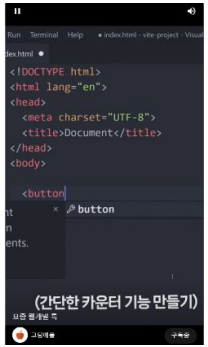
5. 단순하게 작성

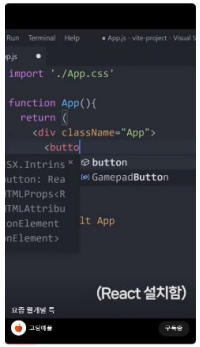
쉬운 내용을 어렵게 짜지 말자
css로 해결할 수 있는 부분을 최대한 활용

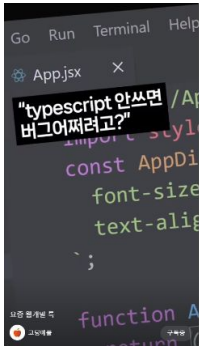


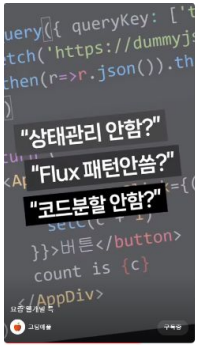
javascript

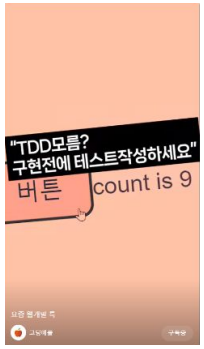
코드정리











댓글 92

Silienne 5일 전
ㄹㅇ 핵공감인 내용이네요ㅋㅋㅋ
4 답글

Ssol 5일 전
상남자 특) 제이쿼리로 화면 단 모든걸 해결함
7 답글

답글 2개

대충 헛소리하는 채널 5일 전
쌍남자 특)바닐라로만 만듬
6 답글

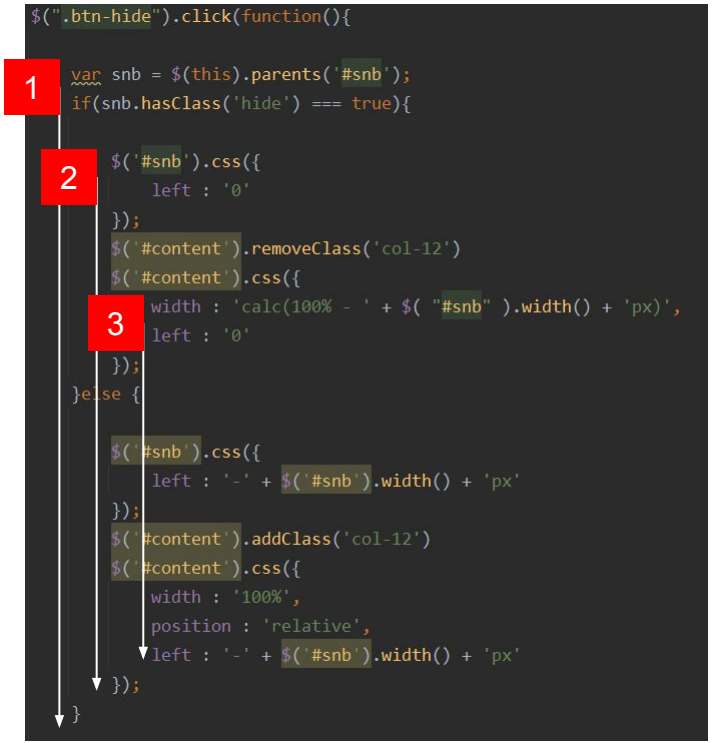
123 4일 전
제이쿼리 이제퇴물되가자낭
답글

Hust 3일 전
그냥 디자이너 할게요
답글

bob 2일 전
아 tdd 진짜 암걸림 ㅋㅋㅋㅋㅋ
답글

6. 들여쓰기, 줄바꿈

최소한의 정리를 하자
뎁스가 4개를 넘어가지 말자



본인 작업 START

다음시간 중간 점검

코드 정리해서 진행한 만큼 가져오면 됩니다.