

Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with





git



GitHub

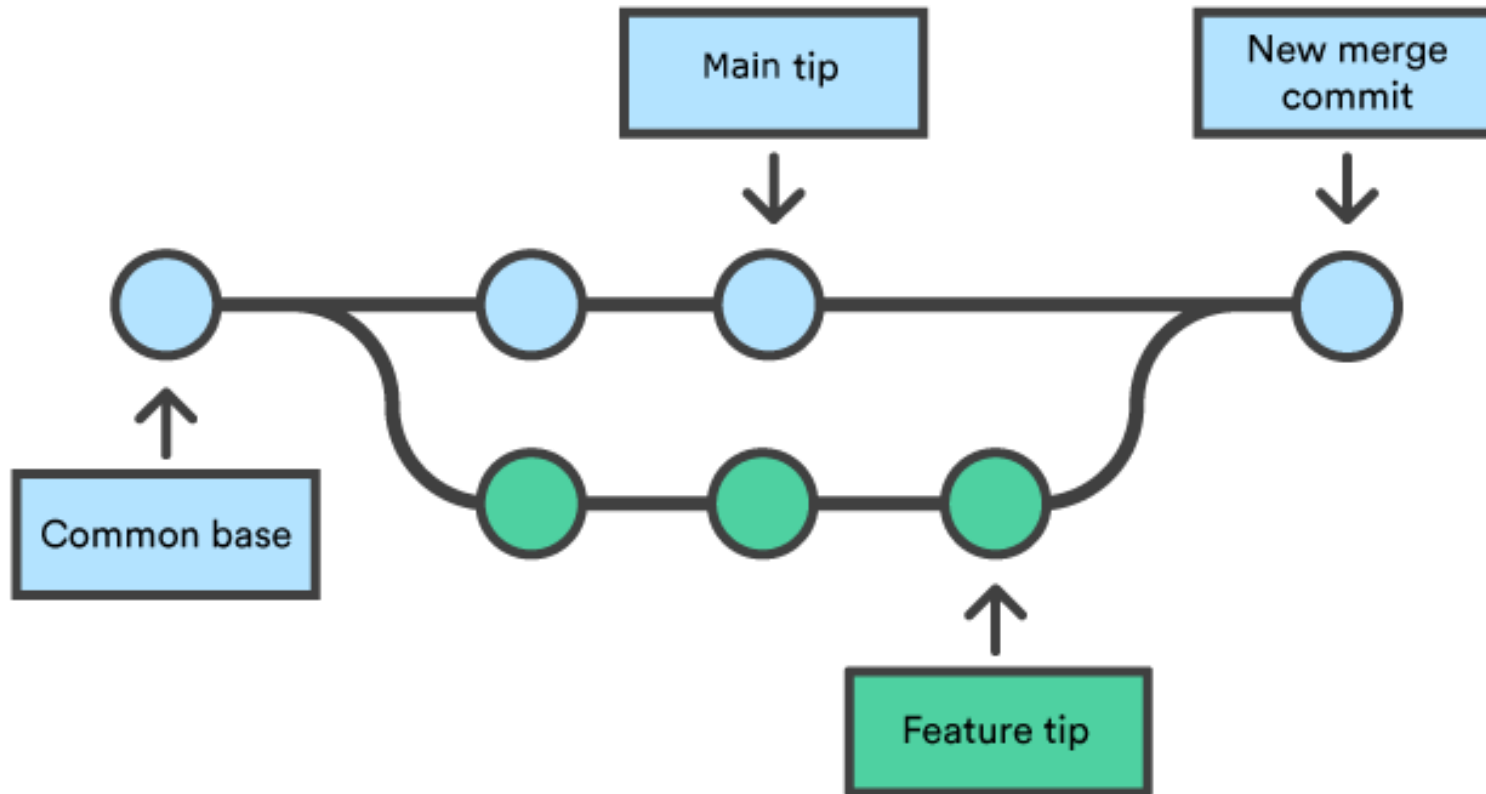
고...!!



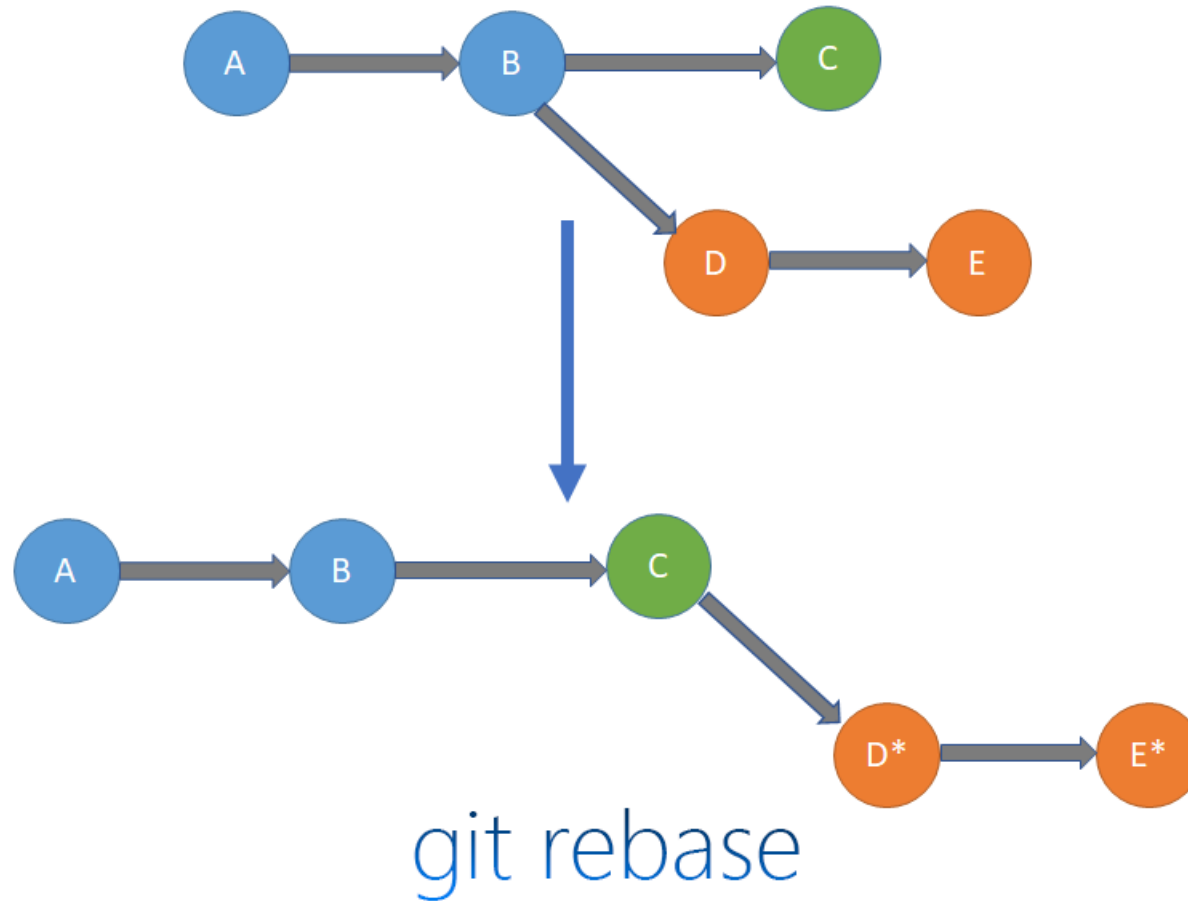
BRANCH를
유지하고
싶어요



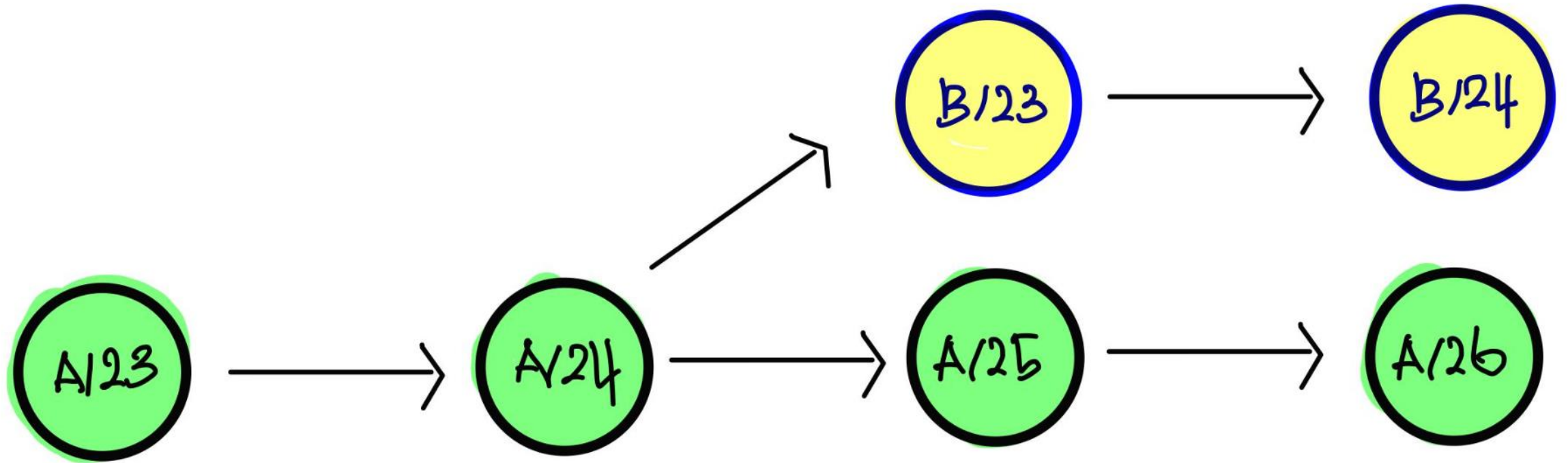
Git MERGING



Git REBASE



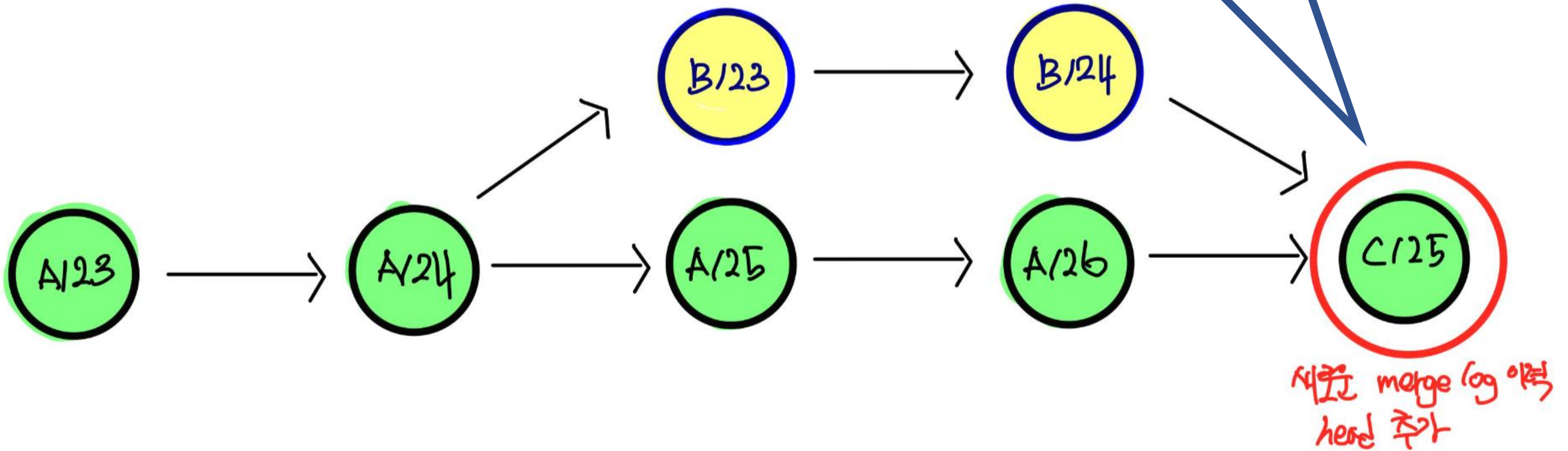
브랜치의 상황



Git MERGE – Main 기준



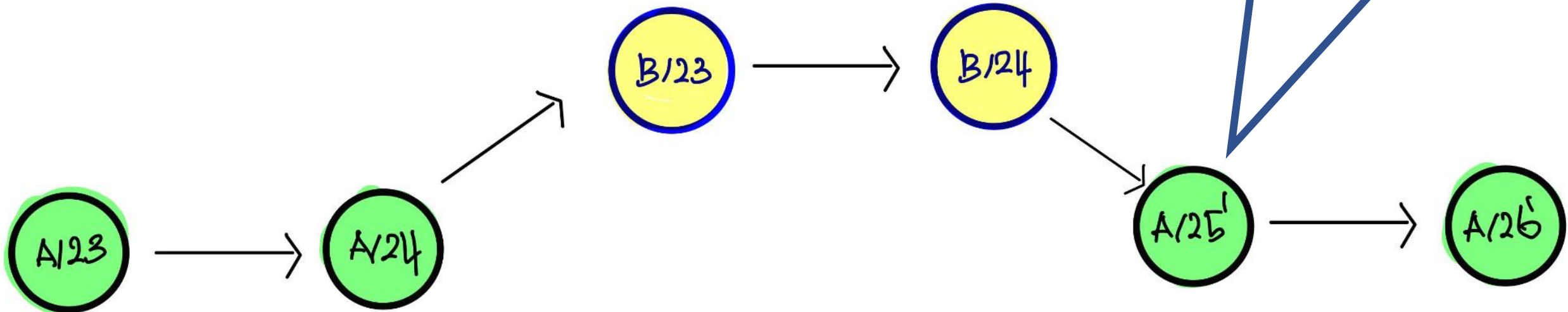
합쳐지는 **B 브랜치**의 기록이
다 남으면서 합치는 **A 브랜치**에
새로운 커밋 이력이 생깁니다!



Git REBASE – Main 기준



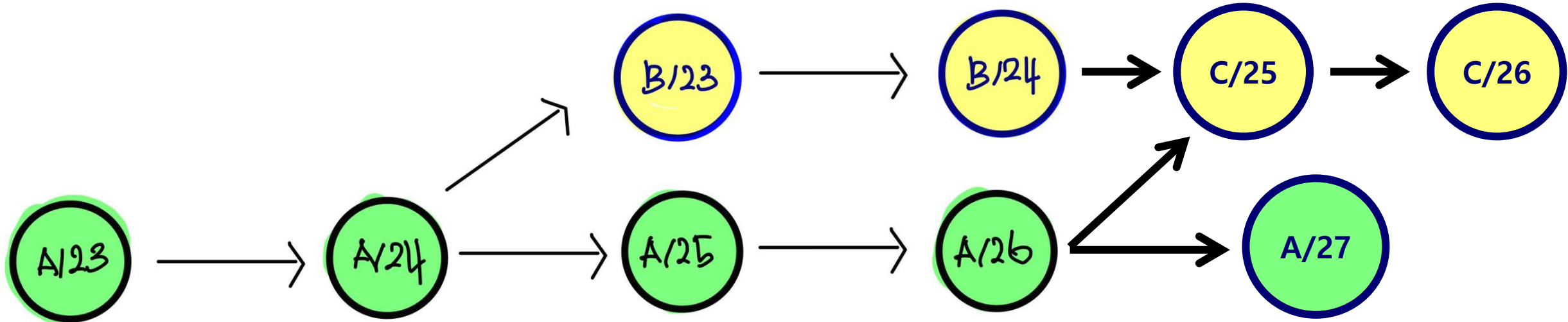
하나의 흐름으로 재정렬이
일어나면서 기존에 B/23, B/24 와
동일한 시간에 발생했던 A/25, A/26에
새로운 ID가 부여 됩니다!



Git MERGE – 브랜치 기준



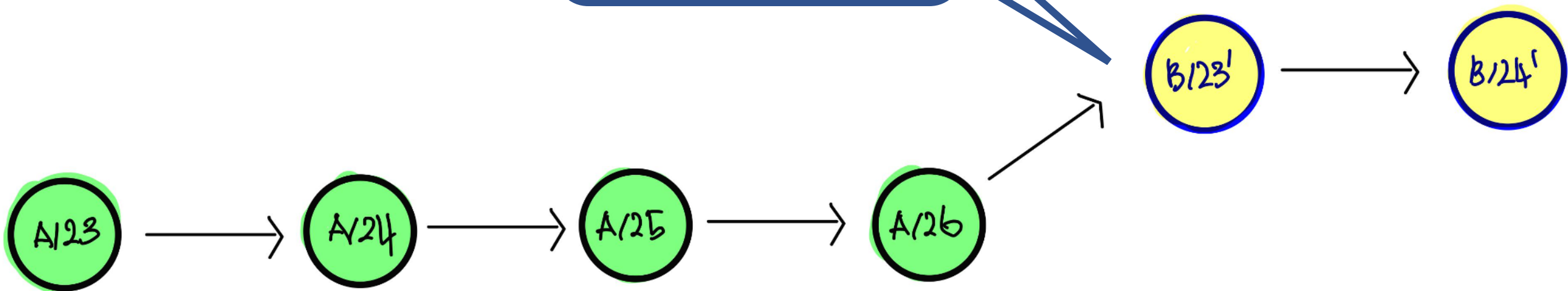
Main 과 브랜치의
모든 이력이 남으면서
관리가 됩니다



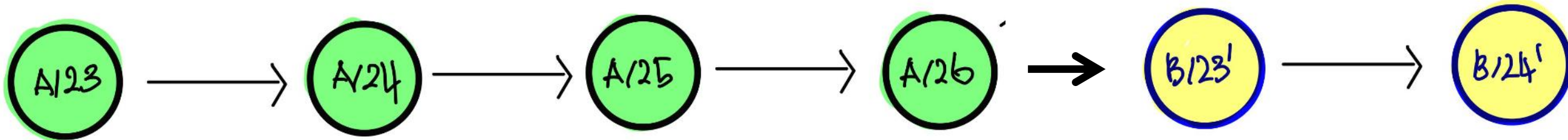
Git REBASE – 브랜치 기준



A/26 에서 브랜치를
만든 것 처럼 이력이
한 줄로 관리가 됩니다



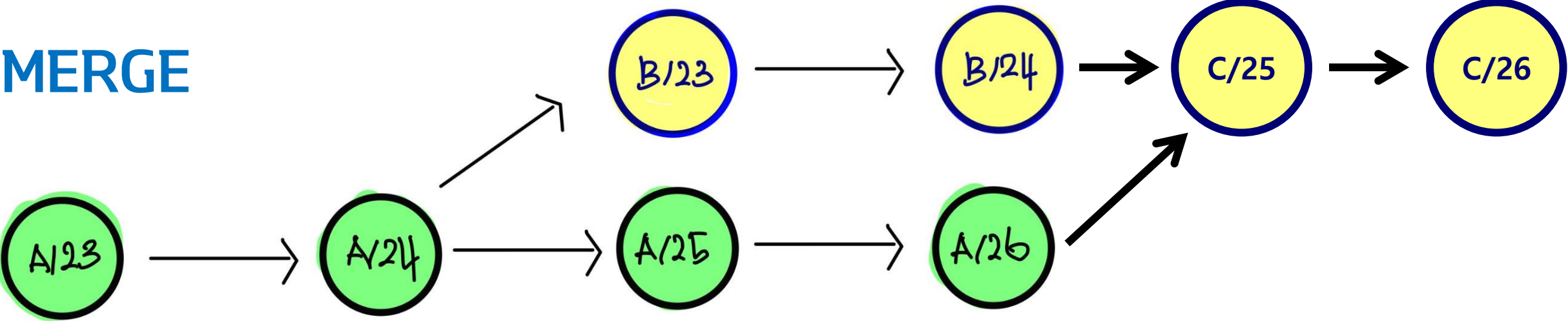
Git REBASE – 브랜치의 관점



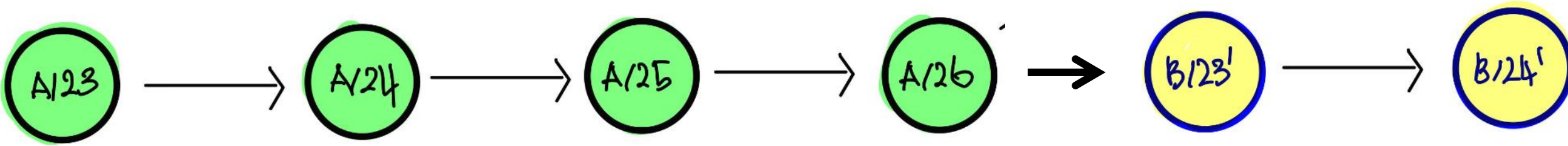
브랜치의 관점



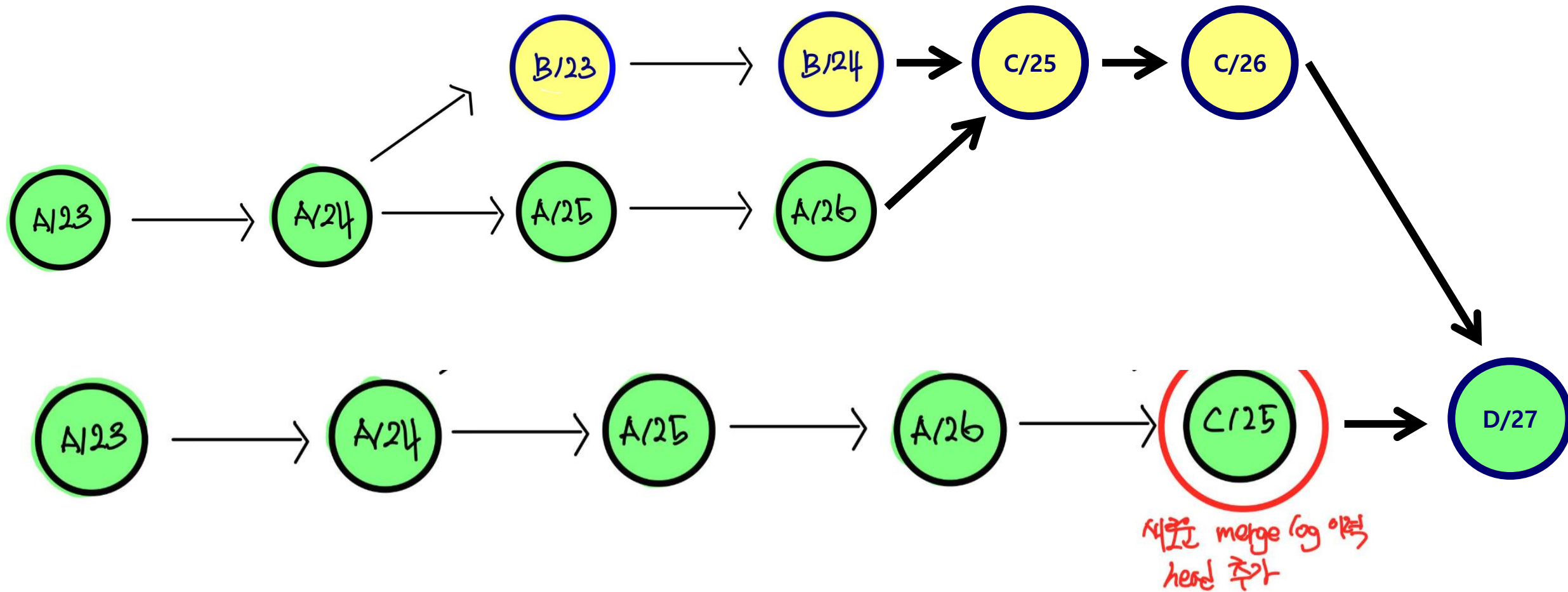
MERGE



REBASE



다시 MERGE 발생!?






실전 Part

전원 Part, 다시 Push!



- `Git push --all`
- 그리고 깃허브로 가봅시다!

 css had recent pushes less than a minute ago

Compare & pull request



전원 Part, 깃헙에서 Merge!




- Merge가 완료되어 main 브랜치에 style.css 파일이 생성

main 2 branches 0 tags Go to file Add file Code

TetzSpreatics Merge pull request #1 from xenosign/css 6e6d956 19 seconds ago 3 commits

| | | |
|------------|-----------|----------------|
| index.html | css 파일 추가 | 17 minutes ago |
| style.css | css 파일 추가 | 17 minutes ago |

Help people interested in this repository understand your project by adding a README. Add a README



- 하지만 아직 우리 Local에는 main의 변경 사항이 반영이 안되어 있습니다!

전원 Part, pull 받기!



- Vscode로 돌아 갑니다!
- Git switch main / git checkout main
- Git pull
- 변경 사항 반영 완료!



전원 Part, rebase 로 메인 코드 땡겨오기



- 자신이 작업 중인 브랜치로 갑니다
- Git switch / checkout 브랜치명
- Git rebase main

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/g-t (js)
$ git rebase main
Successfully rebased and updated refs/heads/js.
```

전원 Part, rebase 로 메인 코드 땡겨오기



- 이제 main 의 최신 사항이 자신의 브랜치에 반영이 되었습니다!
- 그런데 지금 자신의 브랜치에 main 의 코드가 반영 된 것은 Github 에 반영이 되어 있을까요? → 당연히 안되어 있습니다!
- 일단 작업할 사항을 더 작업 → 작업 마무리 → commit → push → PR → Merge 를 하면 됩니다!



영상 기반 랜딩 페이지

KDT Team Tetz 5th



Codign with fun



Coding with us

2023 KDT Tetz Team 5th

영상 기반 랜딩 페이지 <https://movie-main-page.netlify.app/>



영상부터 구해봅시다

<https://www.pexels.com/ko-kr/>

<https://pixabay.com/videos/>



Video Frame

Video Frame, HTML



- 멀티 미디어 콘텐츠의 경우 figure 태그를 사용
- Video 태그로 영상 삽입
 - 속성
 - Autoplay : 자동 재생 여부
 - Muted : 음소거 여부
 - Loop : 반복 여부
 - Controls : 컨트롤 패널 삽입 여부 (기본 속성 : X)
- 영상의 경우 muted 속성이 없으면 자동 재생이 안됩니다!



```
<!-- BACK VIDEO -->  
  <figure>  
    <video src="./video/video.mp4" autoplay muted loop></video>  
  </figure>
```



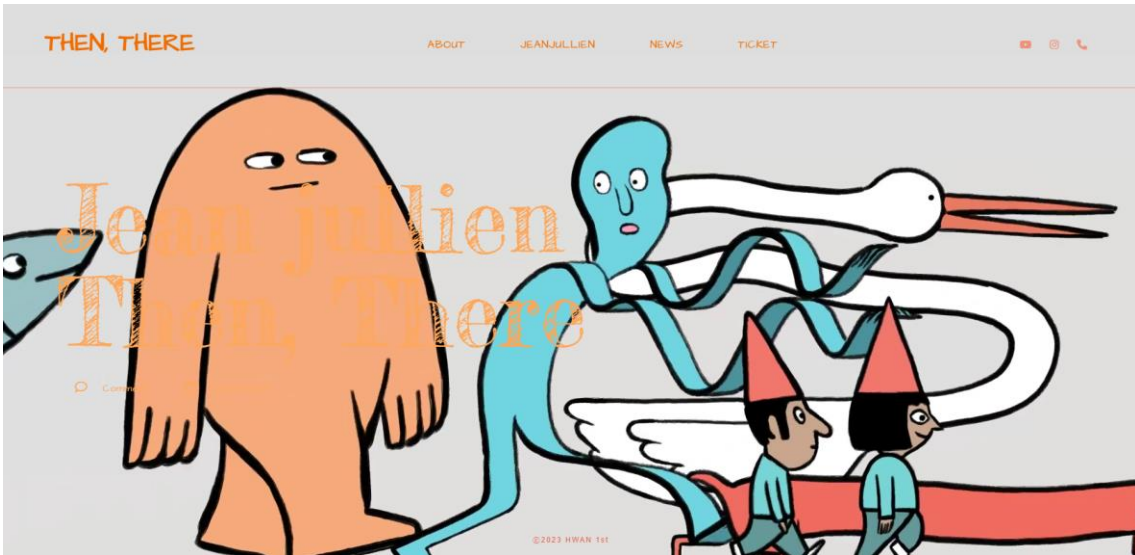
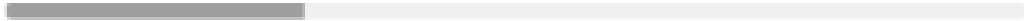
```
/* VIDEO */
figure {
  width: 100%;
  height: 100vh;
  position: absolute;
  top: 0px;
  left: 0px;
  overflow: hidden;
  z-index: 1;

  video {
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
}
```



김계환

7명







지금 시작합니다



Keyframes



@keyframes

- 지금의 animation 효과는 시작과 끝만 설정 가능
- @keyframes 를 사용하면 CSS 속성을 구간별로 정의해서 animation 효과를 줄 수 있습니다!



```
body:active div {  
  animation: animation 5s infinite linear;  
}
```

```
@keyframes animation {  
  from {  
    transform: translate(0, 0);  
  }  
  to {  
    transform: translate(500px, 0);  
  }  
}
```



```
body:active div {  
  animation2: animation 5s infinite linear;  
}
```



```
@keyframes animation2 {  
  0% {  
    transform: translate(0, 0);  
  }  
  25% {  
    transform: translate(500px, 0);  
  }  
  50% {  
    transform: translate(500px, 500px);  
  }  
  75% {  
    transform: translate(0, 500px);  
  }  
  100% {  
    transform: translate(0, 0);  
  }  
}
```



```
<style>
  div {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    background-color: orange;
    transform: translate(0, 0);
  }

  body:active div {
    animation: animation 5s infinite linear;
  }

  @keyframes animation {
    from {
      transform: translate(0, 0);
    }
    to {
      transform: translate(500px, 0);
    }
  }
}
```

```
  @keyframes animation2 {
    0% {
      transform: translate(0, 0);
    }
    25% {
      transform: translate(500px, 0);
    }
    50% {
      transform: translate(500px, 500px);
    }
    75% {
      transform: translate(0, 500px);
    }
    100% {
      transform: translate(0, 0);
    }
  }
</style>
```



LocalStorage



LocalStorage

- 브라우저에 간단한 정보를 저장하기 위한 수단 입니다!
- 라떼에는 Cookie 라는 것을 정말 불편하게 사용 했었는데요. 세상 참 좋아졌... 😊
- 프론트에서 저장할 것이 필요할 때, 브라우저에게 간단하게 정보를 저장 시키는 수단입니다!



LocalStorage 기능

- `setItem()` - key, value 추가
- `getItem()` - value 읽어 오기
- `removeItem()` - item 삭제
- `clear()` - 도메인 내의 localStorage 값 삭제
- `length` - 전체 item 갯수
- `key()` - index로 key값 찾기



LocalStorage 에 값 저장하기!

- `window.localStorage.setItem('key', 'value')` 로 값을 저장 가능

```
<script>  
  window.localStorage.setItem('name', '이효석');  
</script>
```



LocalStorage 의 값 불러오기

- `window.localStorage.getItem('key')` 로 값을 저장 가능합니다!

```
<script>  
  window.localStorage.setItem('name', '이효석');  
  console.log(window.localStorage.getItem('name'));  
</script>
```

이효석





Elements Console Sources Network Performance Memory **Application** Security Lighthouse

Application

- Manifest
- Service Workers
- Storage

Storage

- Local Storage
 - http://127.0.0.1:5599/**
- Session Storage
- IndexedDB
- Web SQL
- Cookies
 - http://127.0.0.1:5599
- Trust Tokens
- Interest Groups
- Shared Storage

Cache

- Cache Storage
- Back/forward cache

Filter

| Key | Value |
|------|-------|
| name | 이호석 |



LocalStorage 에 객체 저장하기!

- localStorage 에는 객체도 저장이 가능합니다!
- 단, 브라우저에 간단하게 저장되는 시스템이므로 문자열로만 저장이 가능하죠! → 그럼? JSON 으로 변경해서 저장!
- JSON.stringify / JSON.parse 사용

JSON!?(JS Object Notation)



- 데이터 통신을 위해 객체 형태의 데이터를 문자열로 변환한 데이터를 말합니다! → 모든 데이터를 쌍따옴표(")를 감싸서 문자열로 전달

```
const tetz = {  
  name: "이효석",  
  isOld: true,  
  hobby: ["음주", "콜라", "플스", "자전거"],  
};  
  
const jsonTetz = JSON.stringify(tetz);
```

```
[Running] node "d:\git\kdt-5th\obj.js"  
{"name":"이효석","isOld":true,"hobby":["음주","콜라","플스","자전거"]}
```



```
<script>
  const tetz = {
    name: "이효석",
    isOld: true,
    hobby: ["음주", "콜라", "플스", "자전거"],
  };

  const jsonObj = JSON.stringify(tetz);
  window.localStorage.setItem('tetz', jsonObj);

  const obj = JSON.parse(window.localStorage.getItem('tetz'));
  console.log(obj);
</script>
```

```
▼ {name: '이효석', isOld: true, hobby: Array(4)} ⓘ
  ▶ hobby: (4) ['음주', '콜라', '플스', '자전거']
    isOld: true
    name: "이효석"
  ▶ [[Prototype]]: Object
```




LocalStorage 에 다른 키 추가하기!

- localStorage 에는 여러 개의 값을 저장 가능합니다!

```
<script>
  window.localStorage.setItem('tetz', '테츠에요!');
  window.localStorage.setItem('pororo', '뽀로로에요!');
  console.log(window.localStorage);
</script>
```

```
▼ Storage {tetz: '테츠에요!', pororo: '뽀로로에요!', length: 2} ⓘ
  pororo: "뽀로로에요!"
  tetz: "테츠에요!"
  length: 2
  ► [[Prototype]]: Storage
```



LocalStorage 에 특정 키 삭제하기!

```
<script>
  window.localStorage.setItem('tetz', '테츠에요!');
  window.localStorage.setItem('pororo', '뽀로로에요!');
  window.localStorage.removeItem('pororo');
  console.log(window.localStorage);
</script>
```

```
▼ Storage {tetz: '테츠에요!', length: 1} ⓘ
  tetz: "테츠에요!"
  length: 1
  ▶ [[Prototype]]: Storage
```



LocalStorage 전체 삭제하기!

```
<script>
  window.localStorage.setItem('tetz', '테츠에요!');
  window.localStorage.setItem('pororo', '뽀로로에요!');
  window.localStorage.clear();
  console.log(window.localStorage);
</script>
```

```
▼ Storage {length: 0} ⓘ
  length: 0
  ► [[Prototype]]: Storage
```

LocalStorage 에 저장 된 데이터 개수 구하기



```
<script>
  window.localStorage.setItem('tetz', '테츠에요!');
  window.localStorage.setItem('pororo', '뽀로로에요!');
  console.log(window.localStorage.length);
</script>
```

2



LocalStorage 에 저장 된 키 이름 찾기!

```
<script>  
  window.localStorage.setItem('tetz', '테츠에요!');  
  window.localStorage.setItem('pororo', '뽀로로에요!');  
  console.log(window.localStorage.key(0));  
  console.log(window.localStorage.key(1));  
</script>
```

tetz

pororo



LocalStorage 만료일 정하기!

```
<script>
  window.localStorage.setItem('tetz', '테츠에요!');
  window.localStorage.setItem('pororo', '뽀로로에요!');
  console.log(window.localStorage.key(0));
  console.log(window.localStorage.key(1));
</script>
```

tetz

pororo



LocalStorage & SessionStorage



SessionStorage 는 뭐죠?

- LocalStorage 가 영구적으로 보존 되는 문제를 해결하기 위해 나왔습니다
- SessionStorage 는 브라우저가 닫히면 저장 된 데이터가 삭제 되기 때문에, 보안이 필요한 데이터를 저장 할 때 사용합니다!

둘의 차이 확인



```
<script>
  const tetz = {
    name: "이효석",
    isOld: true,
    hobby: ["음주", "콜라", "플스", "자전거"],
  };

  const jsonObj = JSON.stringify(tetz);
  window.sessionStorage.setItem("tetz", jsonObj);

  const obj = JSON.parse(window.sessionStorage.getItem("tetz"));
  console.log(obj);
</script>
```



실습, LocalStorage 사용하기

- 로컬 스토리지에 5개 이상의 값을 입력해 주세요! (객체로 저장 X)
- 오늘 배운 localStorage 기능을 이용해서 저장 된 5개의 값의 key 와 value 를 console.log 에 전부 찍어주는 코드를 작성하시면 됩니다!
- For 문 써서 찍어 주세요 😊



```
<script>
  window.localStorage.setItem('tetz', '테츠입니다!');
  window.localStorage.setItem('pororo', '뽀로로예요!');
  window.localStorage.setItem('luppy', '루피예요!');
  window.localStorage.setItem('crong', '크롱이 입니다!');
  window.localStorage.setItem('poby', '포비예요!!');

  for (let i = 0; i < window.localStorage.length; i++) {
    const key = window.localStorage.key(i);
    const value = window.localStorage.getItem(key);
    console.log(`${key} 의 값은 ${value} 입니다!`);
  }
</script>
```



Bootstrap



New in v5.3

Color mode support, expanded color palette, and more!



Build fast, responsive sites with Bootstrap

Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.

```
$ npm i bootstrap@5.3.0-alpha1
```



 **Read the docs**

<https://getbootstrap.com/>



Bootstrap

- **프론트 프레임 워크입니다!**
- **이미 작성 되어있는 프론트 요소들을 가져다 쓰거나, 반응형을 쉽게 구현할 수 있습니다!**
- **즉, Bootstrap 이 구현한 것을 잘 가져다만 쓰면 편리하게 프론트 구성이 가능해 집니다~!**



기본 세팅!



Bootstrap

- <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min."
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bund
  </body>
</html>
```




```
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-GLh1TQ8iRABdZL1603oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
  crossorigin="anonymous"
/>
<script
  defer
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"
  integrity="sha384-mQ93GR66B00ZXjt0Y05KlohRA5SY2XofN4zfuZxLkoj1gXtW8ANNCe9d5Y3eG5eD"
  crossorigin="anonymous"
></script>
```

- Defer 를 쓰면 되므로 <head> 태그 내부에 삽입!



요소 가져다 쓰기

Columns

Gutters

Utilities

Z-index

CSS Grid

Content

Reboot

Typography

Images

Tables

Figures

Forms

Overview

Form controls

Select

Checks & radios

Range

Input group

Floating labels

Layout

Validation

Overview

Bootstrap's form controls expand on [our Rebooted form styles](#) with classes. Use these classes to opt into their customized displays for a more consistent rendering across browsers and devices.

Be sure to use an appropriate `type` attribute on all inputs (e.g., `email` for email address or `number` for numerical information) to take advantage of newer input controls like email verification, number selection, and more.

Here's a quick example to demonstrate Bootstrap's form styles. Keep reading for documentation on required classes, form layout, and more.

Email address

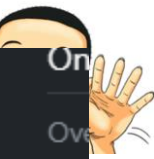
We'll never share your email with anyone else.

Password

☐ Check me out

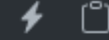
Submit

HTML





HTML



```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHel
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```





```
<body>
  <form>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Email address</label>
      <input
        type="email"
        class="form-control"
        id="exampleInputEmail1"
        aria-describedby="emailHelp"
      />
      <div id="emailHelp" class="form-text">
        We'll never share your email with anyone else.
      </div>
    </div>
    <div class="mb-3">
      <label for="exampleInputPassword1" class="form-label">Password</label>
      <input
        type="password"
        class="form-control"
        id="exampleInputPassword1"
      />
    </div>
    <div class="mb-3 form-check">
      <input type="checkbox" class="form-check-input" id="exampleCheck1" />
      <label class="form-check-label" for="exampleCheck1">Check me out</label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
</body>
```



| |
|--|
| Email address |
| |
| We'll never share your email with anyone else. |
| Password |
| |
| <input type="checkbox"/> Check me out |
| <input type="submit" value="Submit"/> |
| |



Input group
Floating labels
Layout
Validation

Components

Accordion
Alerts
Badge
Breadcrumb
Buttons
Button group
Card
Carousel
Close button
Collapse
Dropdowns
List group
Modal
Navbar

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

Primary

Secondary

Success

Danger

Warning

Info

Light

Dark

[Link](#)

HTML



```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```



Primary

Secondary

Success

Danger

Warning

Info

Light

Dark

[Link](#)



클래스로 컨트롤 하기



```
<div class="parent">
  <div class="child">1</div>
  <div class="child">2</div>
  <div class="child">3</div>
  <div class="child">4</div>
</div>
```

```
<style>
  .parent {
    padding: 50px 50px;
  }
  .child {
    border: 1px solid black;
    width: 100px;
  }
</style>
```

| |
|---|
| 1 |
| 2 |
| 3 |
| 4 |



```
<div class="parent">
  <div class="child">1</div>
  <div class="child">2</div>
  <div class="child">3</div>
  <div class="child">4</div>
</div>
```

```
<style>
  .parent {
    padding: 50px 50px;
    display: flex;
    justify-content: center;
  }
  .child {
    border: 1px solid black;
    width: 100px;
  }
</style>
```

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
|---|---|---|---|



```
<div class="d-flex justify-content-center parent">
  <div class="child">1</div>
  <div class="child">2</div>
  <div class="child">3</div>
  <div class="child">4</div>
</div>
```

```
<style>
  .parent {
    padding: 50px 50px;
  }
  .child {
    border: 1px solid black;
    width: 100px;
  }
</style>
```

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
|---|---|---|---|



```
<div class="d-flex flex-column align-items-center parent">  
  <div class="child">1</div>  
  <div class="child">2</div>  
  <div class="child">3</div>  
  <div class="child">4</div>  
</div>
```

| |
|---|
| 1 |
| 2 |
| 3 |
| 4 |

```
<style>  
  .parent {  
    padding: 50px 50px;  
  }  
  .child {  
    border: 1px solid black;  
    width: 100px;  
  }  
</style>
```



```
<div class="d-flex flex-column align-items-center parent">  
  <div class="child">1</div>  
  <div class="child">2</div>  
  <div class="child">3</div>  
  <div class="child">4</div>  
</div>
```

1

2

3

4

```
<style>  
  .parent {  
    padding: 50px 50px;  
  }  
  .child {  
    border: 1px solid black;  
    width: 100px;  
    margin-top: 30px;  
  }  
</style>
```



```
<div class="d-flex flex-column align-items-center parent">  
  <div class="mt-4 child">1</div>  
  <div class="mt-4 child">2</div>  
  <div class="mt-4 child">3</div>  
  <div class="mt-4 child">4</div>  
</div>
```

1

2

3

4

```
<style>  
  .parent {  
    padding: 50px 50px;  
  }  
  .child {  
    border: 1px solid black;  
    width: 100px;  
  }  
</style>
```



```
<div class="d-flex flex-column align-items-center parent">  
  <div class="mt-4 p-3 child">1</div>  
  <div class="mt-4 child">2</div>  
  <div class="mt-4 child">3</div>  
  <div class="mt-4 child">4</div>  
</div>
```

1

2

3

4

```
<style>  
  .parent {  
    padding: 50px 50px;  
  }  
  .child {  
    border: 1px solid black;  
    width: 100px;  
  }  
</style>
```




Grid



Container

- Bootstrap 의 내용을 담는 용기 역할을 하는 DIV 요소
- Bootstrap 의 그리드를 위한 필수 요소이며, wrapping 요소
- Container 는 화면 넓이에 따라 고정폭의 넓이 제공
- Container-fluid 는 화면 넓이를 최대한 가져가는 넓이 제공
- Container 중첩은 사용 X



```
<div class="container" style="background-color: orange">  
  <div class="fixed">fixed width (.container)</div>  
</div>  
<br />  
<div class="container-fluid" style="background-color: skyblue">  
  <div class="fluid">full width (.container-fluid)</div>  
</div>
```

fixed width (.container)

full width (.container-fluid)



Container

- 화면 넓이 1400px 이상 → 넓이 1320px
- 화면 넓이 1400 ~ 1200px → 넓이 1140px
- 화면 넓이 1200 ~ 992px → 넓이 960px

| | Extra small <576px | Small ≥576px | Medium ≥768px | Large ≥992px | X-Large ≥1200px | XX-Large ≥1400px |
|-------------------------|-----------------------|-----------------|------------------|-----------------|--------------------|---------------------|
| <code>.container</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |



| | Extra small <576px | Small ≥576px | Medium ≥768px | Large ≥992px | X-Large ≥1200px | XX-Large ≥1400px |
|-------------------------------|-----------------------|-----------------|------------------|-----------------|--------------------|---------------------|
| <code>.container</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-sm</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-md</code> | 100% | 100% | 720px | 960px | 1140px | 1320px |
| <code>.container-lg</code> | 100% | 100% | 100% | 960px | 1140px | 1320px |
| <code>.container-xl</code> | 100% | 100% | 100% | 100% | 1140px | 1320px |
| <code>.container-xxl</code> | 100% | 100% | 100% | 100% | 100% | 1320px |
| <code>.container-fluid</code> | 100% | 100% | 100% | 100% | 100% | 100% |



Getting started

An overview of Bootstrap, how to download and use, basic templates and examples, and more.

Demystify performance metrics. [Play freely](#)

Processes, Apps, Hosts. Monitor it All via Carbon

Download

Bootstrap (currently v3.3.4) has a few easy ways to quickly get started, each one appealing to a different skill level and use case. Read through to see what suits your particular needs.

Bootstrap

Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included.

[Download Bootstrap](#)

Source code

Source Less, JavaScript, and font files, along with our docs. **Requires a Less compiler and some setup.**

[Download source](#)

Sass

Bootstrap ported from Less to Sass for easy inclusion in Rails, Compass, or Sass-only projects.

[Download Sass](#)

Bootstrap CDN

The folks over at [MaxCDN](#) graciously provide CDN support for Bootstrap's CSS and JavaScript. Just use these [Bootstrap CDN](#) links.

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->
```

[Copy](#)

- Download
- What's included
- Compiling CSS and JavaScript
- Basic template
- Examples
- Tools
- Community
- Creating responsiveness
- Migrating from 2.x to 3.0
- Browser and device support
- Third party support
- Accessibility
- License FAQs
- Translations
- Back to top

Bootstrap Grid System

그리드 레이아웃을 구성 시에는 반드시 **.row** (행)를 먼저 배치하고 행 안에 **.col-**-*** (열)을 필요한 갯수만큼 배치한다. 즉, container 내에 .row(행)을 먼저 배치하고 그 안에 .col-**-*(열)을 배치한다. 그리고 콘텐츠는 .col-**-* 내에 배치한다.

Row



- 컨테이너 내부에 배치하는 행의 개념



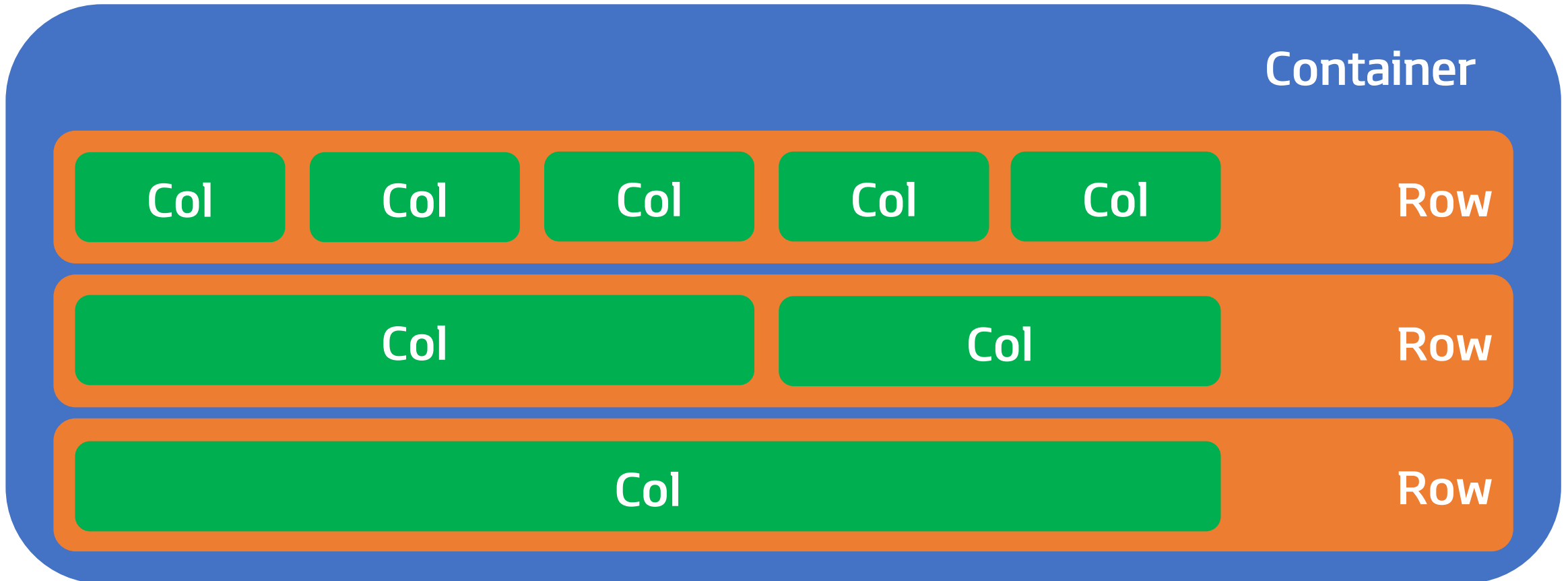


```
<div class="container">  
  <div class="row">  
    <!-- ... -->  
  </div>  
  <div class="row">  
    <!-- ... -->  
  </div>  
</div>
```




Col(umn)

- 컨테이너 내부에 위치 한 Row 안에 위치하는 열의 개념





```
<div class="container">
  <div class="row">
    <div class="col-xs-8">xs-8</div>
    <div class="col-xs-4">xs-4</div>
  </div>
  <div class="row">
    <div class="col-xs-5">xs-5</div>
    <div class="col-xs-5">xs-5</div>
  </div>
</div>
```

```
<style>
  .row {
    margin-bottom: 20px;
  }

  [class|="col"] {
    border: 1px solid black;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="row">
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
      <div class="col-1">col-1</div>
    </div>
    <div class="row">
      <div class="col-4">col-4</div>
      <div class="col-4">col-4</div>
      <div class="col-4">col-4</div>
    </div>
    <div class="row">
      <div class="col-6">col-6</div>
      <div class="col-6">col-6</div>
    </div>
    <div class="row">
      <div class="col-12">col-12</div>
    </div>
    <div class="row">
      <div class="col-6">col-6</div>
      <div class="col-6">col-6</div>
      <div class="col-6">col-6</div>
    </div>
  </div>
</body>
```





Col(umn)의 구성

- Col은 12 그리드를 Col-숫자 만큼의 영역으로 나누어 사용합니다!

| | | | | | | | | | | | |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 |
| col-4 | | | | col-4 | | | | col-4 | | | |
| col-6 | | | | | | col-6 | | | | | |
| col-12 | | | | | | | | | | | |
| col-6 | | | | | | col-6 | | | | | |
| col-6 | | | | | | | | | | | |

- 숫자의 총합이 12를 넘어가면 수직으로 쌓입니다!



반응형



| | Extra small <576px | Small ≥576px | Medium ≥768px | Large ≥992px | X-Large ≥1200px | XX-Large ≥1400px |
|-------------------------------|-----------------------|-----------------|------------------|-----------------|--------------------|---------------------|
| <code>.container</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-sm</code> | 100% | 540px | 720px | 960px | 1140px | 1320px |
| <code>.container-md</code> | 100% | 100% | 720px | 960px | 1140px | 1320px |
| <code>.container-lg</code> | 100% | 100% | 100% | 960px | 1140px | 1320px |
| <code>.container-xl</code> | 100% | 100% | 100% | 100% | 1140px | 1320px |
| <code>.container-xxl</code> | 100% | 100% | 100% | 100% | 100% | 1320px |
| <code>.container-fluid</code> | 100% | 100% | 100% | 100% | 100% | 100% |



복합 사용



Desktop / Tablet / Mobile

- 보통 모바일의 경우 768px 이하 → col-sm 적용
- 태블릿은 768 ~ 992px → col-md 적용
- 데스크탑은 992px 이상 → col-lg 적용



```
<div class="container">
  <div class="row">
    <div class="col-lg-4 col-md-6 col-sm-12">1</div>
    <div class="col-lg-4 col-md-6 col-sm-12">2</div>
    <div class="col-lg-4 col-md-6 col-sm-12">3</div>
    <div class="col-lg-4 col-md-6 col-sm-12">4</div>
    <div class="col-lg-4 col-md-6 col-sm-12">5</div>
    <div class="col-lg-4 col-md-6 col-sm-12">6</div>
  </div>
</div>
```



| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |

463px × 929px

1

2

3

4

5

6



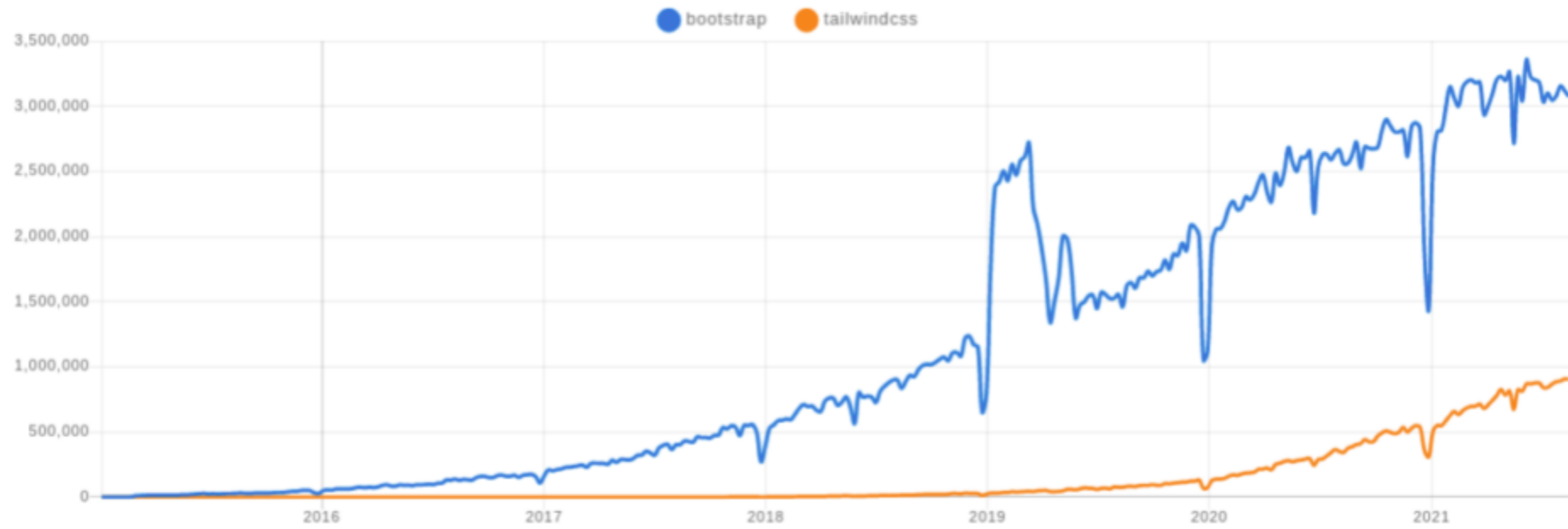
tailwindcss

vs





Downloads in past All time ▾



Enter an npm package...

bootstrap x

tailwindcss x

+ semantic-ui

+ react-bootstrap

+ reactstrap

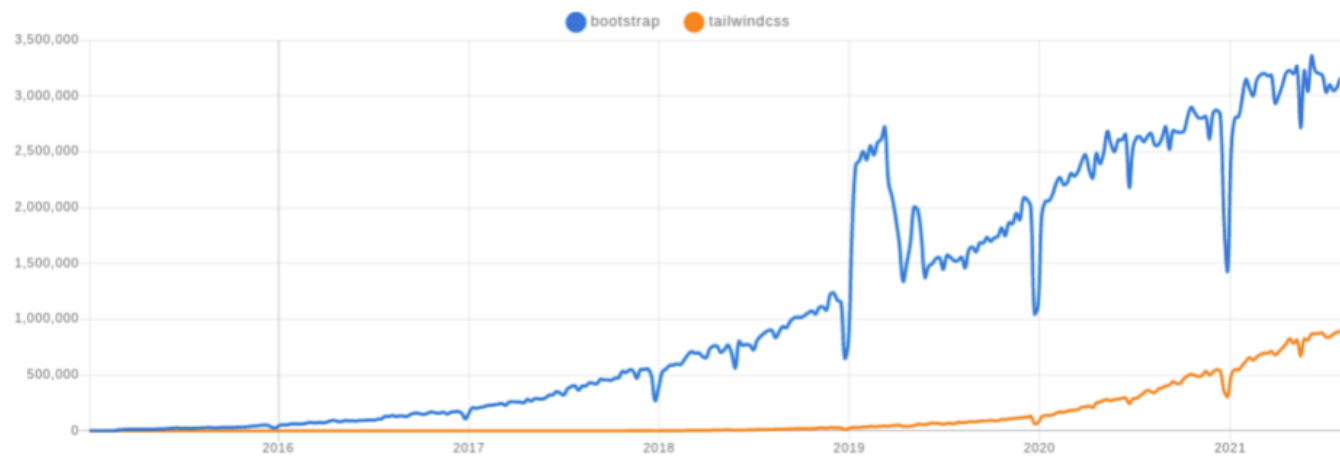
+ material

+ ngx-bootstrap

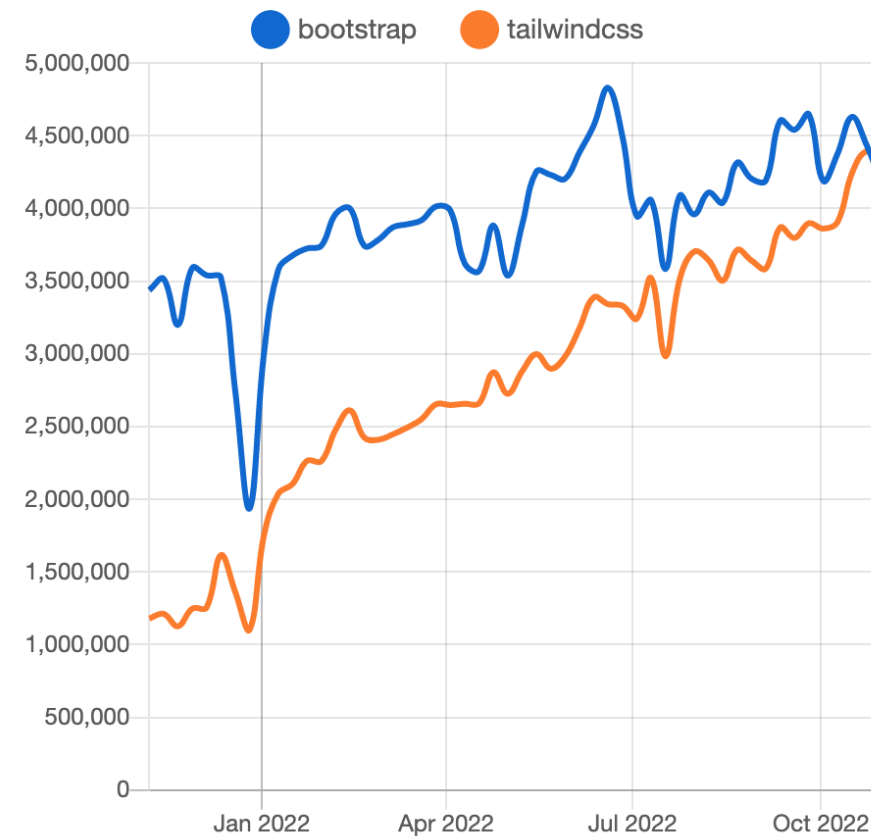
+ @angular/material

+ ng-bootstrap

Downloads in past All time ▾



Downloads in past 1 Year ▾





뮤직 플레이어 제작!

PREV
Music

NEXT
Music



groove

Lorem ipsum dolor, sit amet consectetur
adipiscing elit.



2022 KDT CLASS



cardio

Lorem ipsum dolor, sit amet consectetur
adipiscing elit.



happy

Lorem ipsum dolor, sit amet consectetur
adipiscing elit.



<https://tetz-music-player.netlify.app/>



기본 세팅하기~!



기본 세팅하기!

- 음악 및 이미지 파일 다운로드!
- 폰트 어썸 연결하기!
- JS 파일 연결하기!(Defer)
- Scss 파일 컴파일 → CSS 파일 링크!
- 폰트 가져오기
 - Oswald / VT323 (각자 원하는 것으로 가져오기!)
- _reset 설정



```
<script
  src="https://kit.fontawesome.com/d15cbbf71d.js"
  crossorigin="anonymous"
></script>
<link rel="stylesheet" href="./css/style.css" />
<script defer src="./js/main.js"></script>
```

Index.html

```
* {
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
```

```
ul,
ol {
  list-style: none;
}
```

```
a {
  text-decoration: none;
}
```

_reset.scss



```
@charset "UTF-8";
@import "reset";

@import url("https://fonts.googleapis.com/css2?family=VT323&display=swap");
// font-family: 'VT323', monospace;
@import url("https://fonts.googleapis.com/css2?family=Oswald&display=swap");
// font-family: "Oswald", sans-serif;
```

style.scss



기본 UI 설정하기!

기본 UI, HTML 작업



- Figure 태그 하나 사용!
- H1 으로 로고 작업!
- A 태그로 bars 메뉴 설정(실제로 작동은 X)
- P 태그로 푸터 설정!



```
<body>
  <figure>
    <h1>
      <strong>TetzKDT</strong><br>
      <span>Coding with fun</span>
    </h1>

    <a href="#" class="menu">
      <i class="fas fa-bars"></i>
    </a>

    <p>2022 KDT CLASS</p>
  </figure>
```

기본 UI, CSS 작업!



- Body

- 전체 폰트 설정

- Figure

- 뷰포트 만큼 크기 주기
 - Overflow: hidden
 - Position: relative
 - Background: linear-gradient(25deg, dodgerblue, rgb(71, 255, 255));



```
body {  
  font: 16px/1 "Oswald";  
}  
  
figure {  
  width: 100%;  
  height: 100vh;  
  overflow: hidden;  
  position: relative;  
  background: linear-gradient(25deg, dodgerblue, rgb(71, 255, 255));  
}
```


기본 UI, CSS 작업!



- H1
 - Position: absolute;
 - Top: 7vh / left: 4vw
 - 줄 간격을 없애기 위하여 → font-size: 0;
- Strong
 - Font: bold 40px/1.4 "Oswald"
 - Color: #fff / letter-spacing: 1px;
- Span
 - Font: 16px/1 "Oswald" / color: #fff / opacity: 0.8 / letter-spacing: 1px



```
figure {  
  h1 {  
    position: absolute;  
    top: 7vh;  
    left: 4vw;  
    font-size: 0;  
  
    strong {  
      font: bold 40px/1.4 "Oswald";  
      color: #fff;  
      letter-spacing: 1px;  
    }  
  
    span {  
      font: 16px/1 "Oswald";  
      color: #fff;  
      opacity: 0.8;  
      letter-spacing: 1px;  
    }  
  }  
}
```

기본 UI, CSS 작업!



- `.menu`
 - Position: absolute / top: 8vh / right: 4vw
 - font-size: 30px / color: #fff
- `> p {`
 - Position: absolute
 - bottom: 7vh / left: 50% / transform: translate(-50%, 0);
 - color: #fff



```
figure {  
  
  .menu {  
    position: absolute;  
    top: 8vh;  
    right: 4vw;  
    font-size: 30px;  
    color: #fff;  
  }  
  
  > p {  
    position: absolute;  
    bottom: 7vh;  
    left: 50%;  
    transform: translate(-50%, 0);  
    font: 16px/1 "Oswald";  
    color: #fff;  
  }  
}
```



음악 패널 작성하기!

패널, HTML 작업



- Section 태그 사용!
- 각각의 음악 패널은 article 로 작업
 - 각각 패널마다 transform 효과를 주어야 하므로 div.inner 를 배치하여 transform 이 중복되어 효과가 안걸리는 문제를 회피!

```
section>article*8>.inner
```

```
<section>  
  <article>  
    <div class="inner"></div>  
  </article>
```

패널, CSS 작업



- 패널은 CSS 를 컴포넌트로 만들어 작업!
- _panel.scss 파일 작성
- Style.scss 파일에 panel 임포트

```
@import "panel";
```

패널, CSS 작업



- `_panel.scss` 파일에서 작업 시작!
- Section
 - Width: 20vw / height: 65vh
 - Position: absolute / left: 50% / top : 50% / margin-left: -10vw / margin-top: 32.5vh
- Article
 - Width: 100% / height: 100%
 - Position : absolute / top : 0px / left: 0px


```
section {  
  width: 20vw;  
  height: 65vh;  
  position: absolute;  
  left: 50%;  
  top: 150%;  
  margin-left: -10vw;  
  margin-top: -32.5vh;  
  
  article {  
    width: 100%;  
    height: 100%;  
    position: absolute;  
    top: 0px;  
    left: 0px;
```



패널, CSS 작업



- **.inner**
 - Width: 100% / height: 100% / background-color: #fff
 - Padiing: 5vh 2.5vw 8vh / border-radius: 10px;
 - Box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.1);
 - Opacity: 0.6

```
section {  
  article {  
    .inner {  
      width: 100%;  
      height: 100%;  
      background-color: #fff;  
      padding: 5vh 2.5vw 8vh;  
      border-radius: 10px;  
      box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.1);  
      opacity: 0.6;  
    }  
  }  
}
```



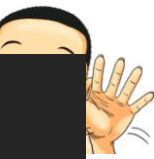
패널, 회전 주기



- 각각의 패널에 회전 값을 JS 를 통해서 각각의 회전 값을 한번에 처리해 봅시다!
- Section 요소 가져오기
- Section 내부의 article 요소 전부 가져오기
- Article 개수 선언 및 각도 구하기

```
const frame = document.querySelector("section");
const list = frame.querySelectorAll("article");
const len = list.length;
const deg = 360 / len;

for (let i = 0; i < len; i++) {
  list[i].style.transform = `rotate(${deg * i}deg`;
}
```



패널, 회전 주기



- 패널이 지금은 자기 자신의 중앙을 기준으로 회전 되어 있으므로 특정 기준 점을 기준으로 퍼져야 하므로 translateY 값 부여

```
for (let i = 0; i < len; i++) {  
  list[i].style.transform = `rotate(${deg * i}deg) translateY(-100vh)`;
```

- 패널이 너무 위로 가있으므로!
- Section 의 top → 150%

```
section {  
  width: 20vw;  
  height: 65vh;  
  position: absolute;  
  left: 50%;  
  top: 150%;  
  margin-left: -10vw;  
  margin-top: -32.5vh;  
  transition: 1s;
```



좌우 이동 버튼 만들기!

버튼, HTML 작업



- Section 태그 아래에 nav 태그로 작업!
- 이전, 이후 버튼이 있어야 하므로! 2개 작업

```
<nav class="btnPrev">  
  <span>PREV Music</span>  
</nav>  
  
<nav class="btnNext">  
  <span>NEXT Music</span>  
</nav>
```


버튼, CSS 작업



- 버튼도 컴포넌트로 작업!
- _btns.scss 파일 생성
- Style.scss 파일에 임포트!

```
@import "panel";  
@import "btns";
```

버튼, CSS 작업



- **.btnPrev**
 - Width: 60px / height: 60px
 - Position: absolute / top: 50% / left: 50%
 - Transform(-20vw, -50%)
 - Display: flex / justify-content: flex-start / align-items: center
 - Font-size: 0 / padding-left: 20px / cursor: pointer
- **Span**
 - Font-size: 18px / color: #fff / transition: 0.5s

```
.btnPrev {  
  width: 60px;  
  height: 60px;  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  transform: translate(-20vw, -50%);  
  display: flex;  
  align-items: center;  
  justify-content: flex-start;  
  cursor: pointer;  
  font-size: 0;  
  padding-left: 20px;  
  
  span {  
    font-size: 18px;  
    color: #fff;  
    transition: 0.5s;  
  }  
}
```



버튼, 움직이는 화살표 만들기



- 요런거는 보통 가상 요소 선택자로 만듭니다!
- ::Before / ::after 둘 다 선언해서 만들고 Ani 효과 주기
- ::Before
 - Content: ""
 - Display: block / width: 100% / height: 2px / background-color: #fff
 - Position: absolute / left: 0px
 - Transform-origin: left center / transform: rotate(-180deg)
 - Transition: 0.5s

```
.btnPrev {  
  &::before,  
  &::after {  
    content: "";  
    display: block;  
    width: 100%;  
    height: 2px;  
    background-color: #fff;  
    position: absolute;  
    left: 0px;  
    transform-origin: left center;  
    transform: rotate(-180deg);  
    transition: 0.5s;  
  }  
}
```

```
&::after {  
  transform: rotate(180deg);  
}
```



버튼, 움직이는 화살표 만들기



- Hover 효과 주기
 - 글자는 속 사라지게!
 - ::Before / ::After 는 각도를 변경해서 화살표처럼 보이도록
 - 30도를 기준으로 위 아래로 위치 하도록 설정!



```
.btnPrev {  
  &:hover {  
    span {  
      transform: translate(100%, 0);  
      opacity: 0;  
    }  
  
    &::before {  
      transform: rotate(-30deg);  
    }  
  
    &::after {  
      transform: rotate(30deg);  
    }  
  }  
}
```

Next 버튼도 작업해 봅시다!



- Prev 버튼과 거의 유사한 Next 버튼 작업하기!


```
.btnNext {
  width: 60px;
  height: 60px;
  position: absolute;
  top: 50%;
  right: 50%;
  transform: translate(20vw, -50%);
  display: flex;
  cursor: pointer;
  font-size: 0;
  align-items: center;
  justify-content: flex-start;
  padding-right: 20px;

  span {
    font-size: 18px;
    color: #fff;
    opacity: 1;
    transition: 0.5s;
  }
}
```

```
&::before, &::after {
  content: "";
  display: block;
  width: 100%;
  height: 2px;
  background-color: #fff;
  position: absolute;
  top: 50%;
  left: 0px;
  transform-origin: right center;
  transform: rotate(-180deg);
  transition: 0.5s;
}
&::after {
  transform: rotate(180deg);
}
&:hover {
  span {
    transform: translate(-100%, 0);
    opacity: 0;
  }
  &::before {
    transform: rotate(-30deg);
  }
  &::after {
    transform: rotate(30deg);
  }
}}
```



좌우 이동 버튼

클릭 액션 처리

버튼, JS 처리!



- 버튼을 각각 불러옵시다!
- 해당 패널의 위치 값을 저장할 변수 만들기!
- 각각 버튼에 이벤트 붙이기!
- 0을 기준으로 next 를 누르면 $+1 * 45\text{deg}$ 씩 돌아가고
- Prev 를 누르면 $-1 * 45\text{deg}$ 씩 돌아가면 됩니다!
- 다만, 해당 속성 값은 누적이 아니므로 next 가 2번 눌리면 45deg 에서 90deg 로 돌아야 합니다!
- 즉, 해당 패널의 위치 값을 0 을 기준으로 정해서 곱하면 됩니다!



```
const prev = document.querySelector(".btnPrev");
const next = document.querySelector(".btnNext");
let num = 0;

prev.addEventListener("click", function (e) {
  frame.style.transform = `rotate(${deg * ++num}deg)`;
});

next.addEventListener("click", function (e) {
  frame.style.transform = `rotate(${deg * --num}deg)`;
});
```

- 돌아가는 애니메이션을 줘야 하는데, 돌아가는 UI는 Section 이므로 Section 에 transition: 1s; 값 추가

```
section {
  transition: 1s;
```



가운데 패널에 활성화 주기

활성화 패널, on 클래스 부여



- 첫번째 패널인 Article 에 on 클래스를 부여

```
<section>
  <article class="on">
    <div class="inner">
```

- On 클래스를 가진 패널은
 - Opacity: 1 / transform: scale(1) 부여

```
&.on {
  .inner {
    opacity: 1;
    transform: scale(1);
  }
}
```

활성화 패널, JS 처리하기



- 전역 변수로 활성화 패널의 위치를 저장 → `let active = 0;`
- Prev 버튼 처리
 - 초창기 위치가 0
 - 0에서 prev 가 눌리면? → 제일 마지막(article 의 개수 - 1)으로 이동
 - 그 외에는 자신의 값에서 -1 만 해주면 됩니다!
 - 그리고 현재 모든 article 에서 on 이라는 클래스를 제거 후 → active 위치에
만 on 클래스 추가!



```
let active = 0;

prev.addEventListener("click", function (e) {
  frame.style.transform = `rotate(${deg * ++num}deg)`;

  if (active === 0) {
    active = len - 1;
  } else {
    active--;
  }

  for (let el of list) {
    el.classList.remove("on");
  }
  list[active].classList.add("on");
});
```


활성화 패널, JS 처리하기



- Next 버튼 처리

- 이 친구는 반대로, 마지막 위치에서 next 가 눌리면 0으로 변경!
- 나머지는 클릭 되면 +1 해주면 되겠죠?
- 역시 모든 article 에서 on 클래스 제거 후 → active 위치에 on 클래스 부여



```
next.addEventListener("click", function (e) {  
    frame.style.transform = `rotate(${deg * --num}deg)`;  
  
    if (active === len - 1) {  
        active = 0;  
    } else {  
        active++;  
    }  
  
    for (let el of list) {  
        el.classList.remove("on");  
    }  
    list[active].classList.add("on");  
});
```



패널 꾸미기

앨범 사진, HTML 작업



- 먼저 앨범 사진을 넣어 보아요!
- 사진을 넣을 div 요소 pic 삽입!
- CD 처럼 보이기 위한 가운데 구멍 dot 도 삽입!

```
<section>
  <article class="on">
    <div class="inner">
      <div class="pic">
        <div class="dot"></div>
      </div>
    </div>
  </article>
```

앨범 사진, CSS 작업



- .pic
 - Width: 15vw / height: 15vw;
- 실제 사진은 ::before 와 ::after 를 사용해서 처리 합니다!
 - Content: ""
 - Display: block / width: 100% / height: 100%
 - position: absolute / top: 0px / left: 0px / border-radius: 50%
 - background-repeat: no-repeat / background-position: center / background-size: cover
 - 이미지 주소로 이미지 삽입!

```
section {
  article {
    .inner {
      .pic {
        height: 15vw;
        width: 15vw;
        position: relative;

        &::before,
        &::after {
          content: "";
          display: block;
          width: 100%;
          height: 100%;
          position: absolute;
          top: 0px;
          left: 0px;
          border-radius: 50%;
          background-repeat: no-repeat;
          background-position: center;
          background-size: cover;
          background-image: "url()";
        }
      }
    }
  }
}
```



앨범 사진, CSS 작업



- 지금은 사진 2장이 겹쳐진 상태이죠?
- ::before 만 따로 이미지를 처리해서 자연스러운 그림자 효과를 줍시다
 - Transform: translste(0, 10%) / filter: blur(20px) brightness(130%);
- .dot 가운데 구멍도 처리해 보죠
 - Width: 2.5vw / height: 2.5vw / postion: absolute / top: 50% / left: 50% / transform: translate(-50%, -50%) / background-color: #fff / border-radius: 50% / box-shadow: inset 5px 5px 10px rgba(0, 0, 0, 0.2); / z-index: 3



```
section {
  article {
    .inner {
      .pic {

        ::before {
          transform: translate(0, 10%);
          filter: blur(20px) brightness(130%);
        }

        .dot {
          width: 2.5vw;
          height: 2.5vw;
          position: absolute;
          top: 50%;
          left: 50%;
          transform: translate(-50%, -50%);
          background-color: #fff;
          border-radius: 50%;
          box-shadow: inset 5px 5px 10px rgba(0, 0, 0, 0.2);
          z-index: 3;
        }
      }
    }
  }
}
```


앨범 사진, JS로 일괄 삽입을 해봅시다!



- 현재 앨범 사진과 음악 mp3 파일명은 동일한 상태이죠?
- JS에서 배열을 사용해서 일괄적으로 이미지를 삽입해 BoA요!
 - 각각의 파일명이 담긴 배열 만들기
 - 이미 사용한 반복문이 있으니, 해당 반복문에 처리를 해봅시다!



```
const names = ["cardio", "groove", "happy", "light", "lily", "limes", "pop", "swing"];

for (let i = 0; i < len; i++) {
  list[i].style.transform = `rotate(${deg * i}deg) translateY(-100vh)`;

  const pic = list[i].querySelector(".pic");
  pic.style.backgroundImage = `url(../img/${names[i]}.jpg)`;
}
```

앨범 사진, before / after 에 상속 시키기



- 지금은 .pic 에 사진이 들어간 상태입니다!
- .pic 사진을 저 멀리 먼저 치웁시다
 - Background-repeat: no-repeat / background-position: -9999px - 9999px;
- 그리고 ::before ::after 에 .pic 의 사진 위치를 상속 시킵니다!



```
.pic {  
  height: 15vw;  
  width: 15vw;  
  position: relative;  
  background-repeat: no-repeat;  
  background-position: -9999px -9999px;  
  
  &::before,  
  &::after {  
    content: "";  
    display: block;  
    width: 100%;  
    height: 100%;  
    position: absolute;  
    top: 0px;  
    left: 0px;  
    border-radius: 50%;  
    background-repeat: no-repeat;  
    background-position: center;  
    background-size: cover;  
    background-image: inherit;  
  }  
}
```

앨범 제목, HTML 영역



- .text 영역 만들기

- H2

- P

```
<section>
  <article class="on">
    <div class="inner">
      <div class="pic">
        <div class="dot"></div>
      </div>
      <div class="text">
        <h2></h2>
        <p>Lorem ipsum dolor, sit amet consectetur.</p>
      </div>
    </div>
  </article>
```

앨범 제목, JS로 제목도 일괄 적용



- 이미 선언한 이름 배열을 또 쓰면 되겠죠?

```
const names = ["cardio", "groove", "happy", "light", "lily", "limes", "pop", "swing"];

for (let i = 0; i < len; i++) {
  list[i].style.transform = `rotate(${deg * i}deg) translateY(-100vh)`;

  const pic = list[i].querySelector(".pic");
  pic.style.backgroundImage = `url(../img/${names[i]}.jpg)`;

  const title = list[i].querySelector(".text>h2");
  title.innerHTML = `${names[i]}`;
}
```

앨범 제목, text 영역 CSS 작업



- **.text**
 - Width: 15vw / text-align: center
 - Position: absolute / margin-top: 60px / letter-spacing: 1px;
- **H2**
 - Margin-bottom: 20px;
- **P**
 - Color: #777;



```
section {  
  article {  
    .inner {  
  
      .text {  
        width: 15vw;  
        text-align: center;  
        position: absolute;  
        margin-top: 60px;  
        letter-spacing: 1px;  
  
        h2 {  
          margin-bottom: 20px;  
        }  
  
        p {  
          color: #777;  
        }  
      }  
    }  
  }  
}
```




음악 플레이어, HTML 작업

- UI 태그 + icon 으로 음악 플레이어를 만듭시다!
- UI > li * 3
 - 각각 li 에 icon 으로 버튼 주기

```
<div class="text">
  <h2></h2>
  <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>
  <ul class="control">
    <li class="pause"><i class="fas fa-pause"></i></li>
    <li class="play"><i class="fas fa-play"></i></li>
    <li class="reload"><i class="fas fa-redo-alt"></i></li>
  </ul>
</div>
```

음악 플레이어, CSS 작업



- UI
 - 가로 배치죠? → `display: flex`
 - 버튼이 적절히 퍼지도록 → `justify-content: space-around`
- UI li
 - `Cursor: pointer;`
 - `Opacity: 0.6 / transition: 0.5s`
- `&.play`
 - 플레이 버튼은 더 크고 선명하게 보이도록 → `transform: scale(1.5) / opacity: 0.8;`

```
section {  
  article {  
    .inner {  
      .text {  
  
        .control {  
          display: flex;  
          justify-content: space-around;  
          margin-top: 60px;  
  
          li {  
            cursor: pointer;  
            opacity: 0.6;  
            transition: 0.5s;  
  
            &.play {  
              transform: scale(1.5);  
              opacity: 0.8;  
            }  
          }  
        }  
      }  
    }  
  }  
}
```



음악 플레이어, CSS 작업



- Hover 효과 주기
 - 각각의 버튼 → 1.5배 / opacity: 0.8
 - Play 버튼 → 1.8배 / opacity: 1

```
li {  
  
    &:hover {  
        transform: scale(1.5);  
        opacity: 0.8;  
  
        &.play {  
            transform: scale(1.8);  
            opacity: 1;  
        }  
    }  
}
```



앨범 이미지 회전

앨범 회전, Keyframes 작업



- 앨범은 ::before 와 ::after 2개의 이미지가 있으므로 2개의 keyframes 작업 필요
- 하나는 자신의 위치에서 360deg 돌고
- 하나는 자신의 위치 10% 아래에서 360deg 돌면 되므로!



```
@keyframes rotation {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

@keyframes rotation2 {
  0% {
    transform: translateY(10%) rotate(0deg);
  }

  100% {
    transform: translateY(10%) rotate(360deg);
  }
}
```

앨범 회전, CSS 작업



- 사진 요소에 on 이라는 클래스가 부여되면 회전 하도록 설정!

```
.pic {  
  &.on {  
    &::before {  
      animation: rotation2 4s linear infinite;  
    }  
  
    &::after {  
      animation: rotation 4s linear infinite;  
    }  
  }  
}
```


앨범 회전, JS 작업



- Play 버튼을 누르면 on 클래스가 부여 되도록 처리!
- Pause 버튼을 누르면 on 클래스가 제거 되도록 처리!



```
for (let el of list) {  
  const play = el.querySelector(".play");  
  const pause = el.querySelector(".pause");  
  const load = el.querySelector(".reload");  
  
  play.addEventListener("click", function (e) {  
    e.currentTarget.closest("article").querySelector(".pic").classList.add("on");  
  });  
  
  pause.addEventListener("click", function (e) {  
    e.currentTarget.closest("article").querySelector(".pic").classList.remove("on");  
  });  
}
```



음악 파일

삽입 및 재생

음악 파일, JS로 일괄 삽입을 해봅시다!



- 아까 쓰던 반복문이 있네요!?
- createElement 로 audio 엘리먼트 만들기
- setAttribute로 위치(src) 속성 입력하기
- 재생이 무한히 반복해야 하므로 loop 속성도 주기
- 각각의 list 요소에 audio 붙여넣기!



```
for (let i = 0; i < len; i++) {  
  list[i].style.transform = `rotate(${deg * i}deg) translateY(-100vh)`;  
  
  const pic = list[i].querySelector(".pic");  
  pic.style.backgroundImage = `url(../img/${names[i]}.jpg)`;  
  
  const title = list[i].querySelector(".text>h2");  
  title.innerHTML = `${names[i]}`;  
  
  const audio = document.createElement("audio");  
  audio.setAttribute("src", `../music/${names[i]}.mp3`);  
  audio.setAttribute("loop", "loop");  
  
  list[i].append(audio);  
}
```

음악 파일, JS로 재생 처리!



- 아까 play 랑 pause 버튼 작업 했었죠!? 거기로 고고싱
- 재생 버튼을 클릭하면 audio 태그를 선택 후 play() 메소드로 재생
- 멈춤 버튼을 클릭하면 audio 태그를 선택 후 pause() 메소드로 멈춤
- 이번에는 reload 버튼도 작업 합시다!
 - 재생과 완전히 동일하지만 처음부터 틀어야 하므로
 - Load() 로 음악을 다시 로드 시킨 후 → play()

```
for (let el of list) {
  const play = el.querySelector(".play");
  const pause = el.querySelector(".pause");
  const load = el.querySelector(".reload");

  play.addEventListener("click", function (e) {
    e.currentTarget.closest("article").querySelector(".pic").classList.add("on");
    e.currentTarget.closest("article").querySelector("audio").play();
  });

  pause.addEventListener("click", function (e) {
    e.currentTarget.closest("article").querySelector(".pic").classList.remove("on");
    e.currentTarget.closest("article").querySelector("audio").pause();
  });

  load.addEventListener("click", function (e) {
    e.currentTarget.closest("article").querySelector(".pic").classList.add("on");

    e.currentTarget.closest("article").querySelector("audio").load();
    e.currentTarget.closest("article").querySelector("audio").play();
  });
}
```



음악 파일, 여러 개가 한꺼번에 재생?



- 다른 노래가 재생 되면 다른 노래가 멈추도록 작업해 봅시다!!
- 이전 노래의 위치를 저장할 before 라는 전역 변수 생성!
- Before 의 초기 값을 0으로 세팅하고 before 가 0일 경우 현재 패널의 위치를 저장
- 다른 패널에서 재생 또는 Reload 버튼을 눌렀을 경우 이전 위치의 노래를 정지
 - 현재 클릭이 된 패널의 위치와 before 를 비교하여 서로 다를 경우 before 의 노래를 정지 → 현 위치를 before 로 등록!



```
play.addEventListener("click", function (e) {  
  if (before === 0) {  
    before = e.currentTarget;  
  } else if (before !== e.currentTarget) {  
    before.closest("article").querySelector("audio").pause();  
    before.closest("article").querySelector(".pic").classList.remove("on");  
    before = e.currentTarget;  
  }  
  
  e.currentTarget.closest("article").querySelector(".pic").classList.add("on");  
  
  e.currentTarget.closest("article").querySelector("audio").play();  
});
```





수고하셨습니다!