

중요한 건 꺾이지 않는 마음

스마트에디터의 도전

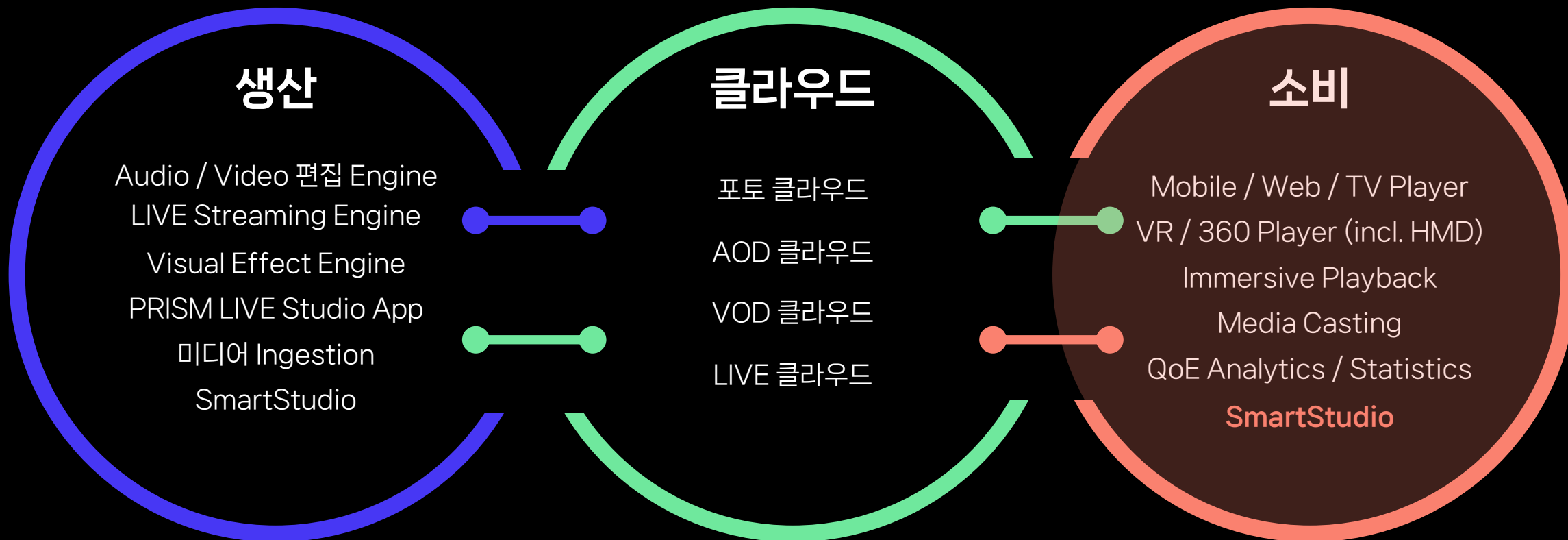
김성원, 이진

NAVER ETECH / SmartEditor

NAVER DEVIEW 2023

NAVER ETECH 를 소개합니다.

NAVER
DEVIEW
2023



CONTENTS

01 Intro

02 스마트에디터의 도전기

과거부터 현재까지

1세대~2세대 에디터

현재

3세대 에디터

많은 컴포넌트 성능 하락

긴 코드 복잡도 증가

디자인 커스텀 리소스 줄이기

Dart SASS를 이용한 운영 효율화

03 시도 정리

04 미래

동시편집, AI

다른 애플리케이션에도
적용 가능한 기술

01

Intro

세상에 완벽한 것은 없고,
초기에는 보잘 것 없다.

하지만,
시간이 지남에 따라 어떤 것은 우수해진다.

어떤 것이 우수해질까?

도자기 만들기 실험

A그룹

많이 만들수록
높은 점수를 얻음

B그룹

최고의 작품
한 점만 채점

미적, 기술적, 섬세함 면에서
최고의 작품이 속한 그룹

꿈임없이 시도하는 것이 우수해진다.

02

스마트에디터 도전기

2-1

과거부터 현재까지

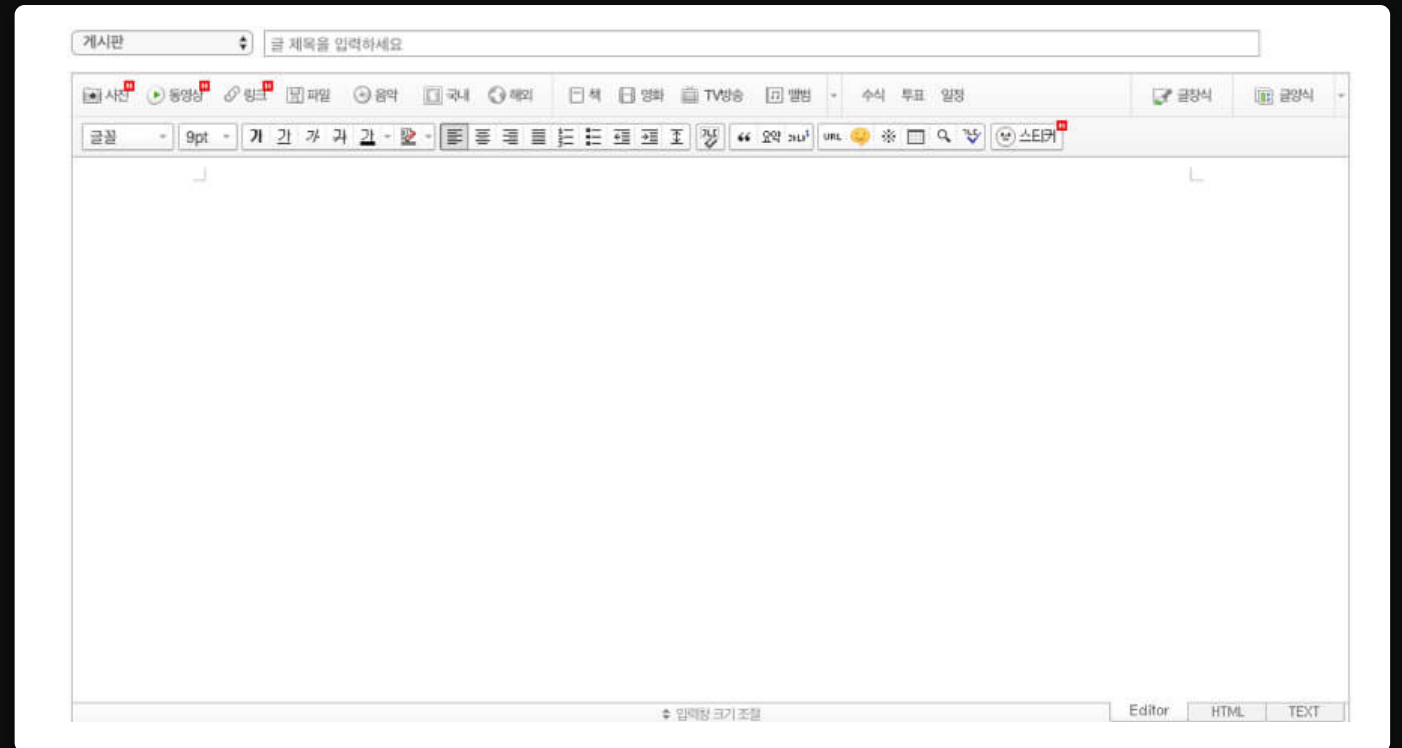
과거부터 현재까지

1세대 에디터 (SE 2.0)

NAVER
DEVIEW
2023

특징

- contenteditable 기반
- 문서를 HTML 그대로 저장
- ex. SE 1.0, SE 2.0 (구 네이버 메일)



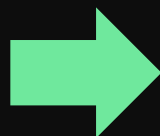
1세대 에디터 (SE 2.0)

contenteditable 영역

HTML

```
<div contenteditable="true">
```

글을 입력할 수 있어요.



```
<div contenteditable="true">글을 입력할 수 있어요.</div>
```

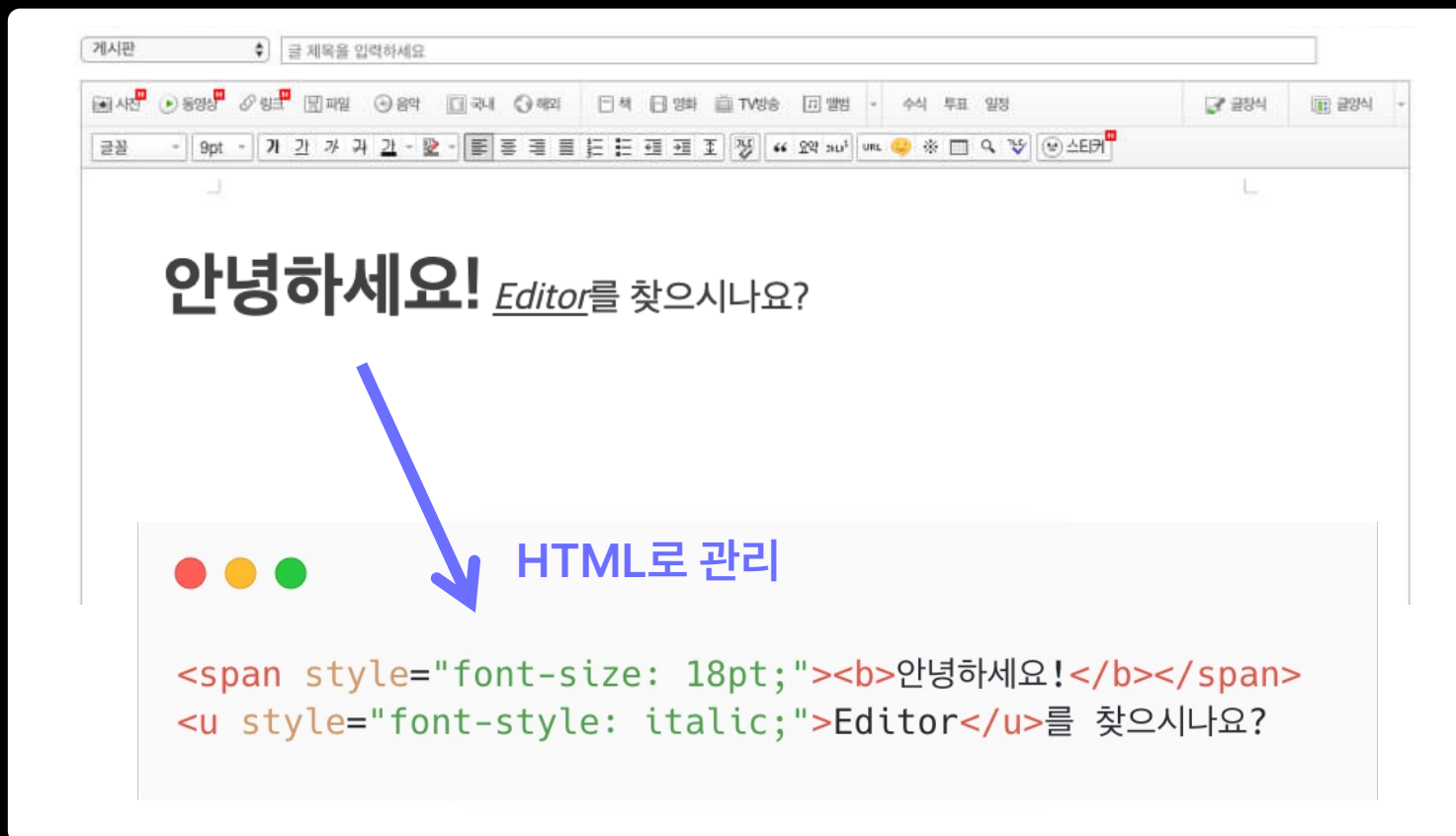
- 텍스트 입력
- 커서 이동
- execCommand 로 서식 변경 (bold, italic 등)

브라우저 대응의 어려움

- IE 버전 별 대응 필요

멀티 디바이스 대응의 어려움

- 문서를 HTML 그대로 저장
- 스타일이 inline style로 추가됨



1.5세대 에디터 (SE 3.0)

특징

- 일부 (텍스트 영역) contenteditable 사용
- 문서 모델링 (JSON), 블록별 편집

1세대

멀티 디바이스
대응 이슈

문제 해결



문서 모델링
(JSON)

1.5세대

과거부터 현재까지

1.5세대 에디터 (SE 3.0)

NAVER
DEVIEW
2023

문서 모델링

문서 데이터를
JSON으로 변환해 저장

안녕하세요! *Editor*를 찾으시나요?

JSON

```
"components": [{
  "value": [{
    "nodes": [
      {
        "value": "안녕하세요!",
        "style": { "fontSizeCode": "fs19", "bold": true, "@ctype": "nodeStyle" },
        "@ctype": "textNode"
      },
      {
        "value": "Editor",
        "style": { "italic": true, "underline": true, "@ctype": "nodeStyle" },
        "@ctype": "textNode"
      },
      {
        "value": "를 찾으시나요?",
        "@ctype": "textNode"
      }
    ],
    "@ctype": "paragraph"
  }],
  "@ctype": "text"
}]
```

과거부터 현재까지

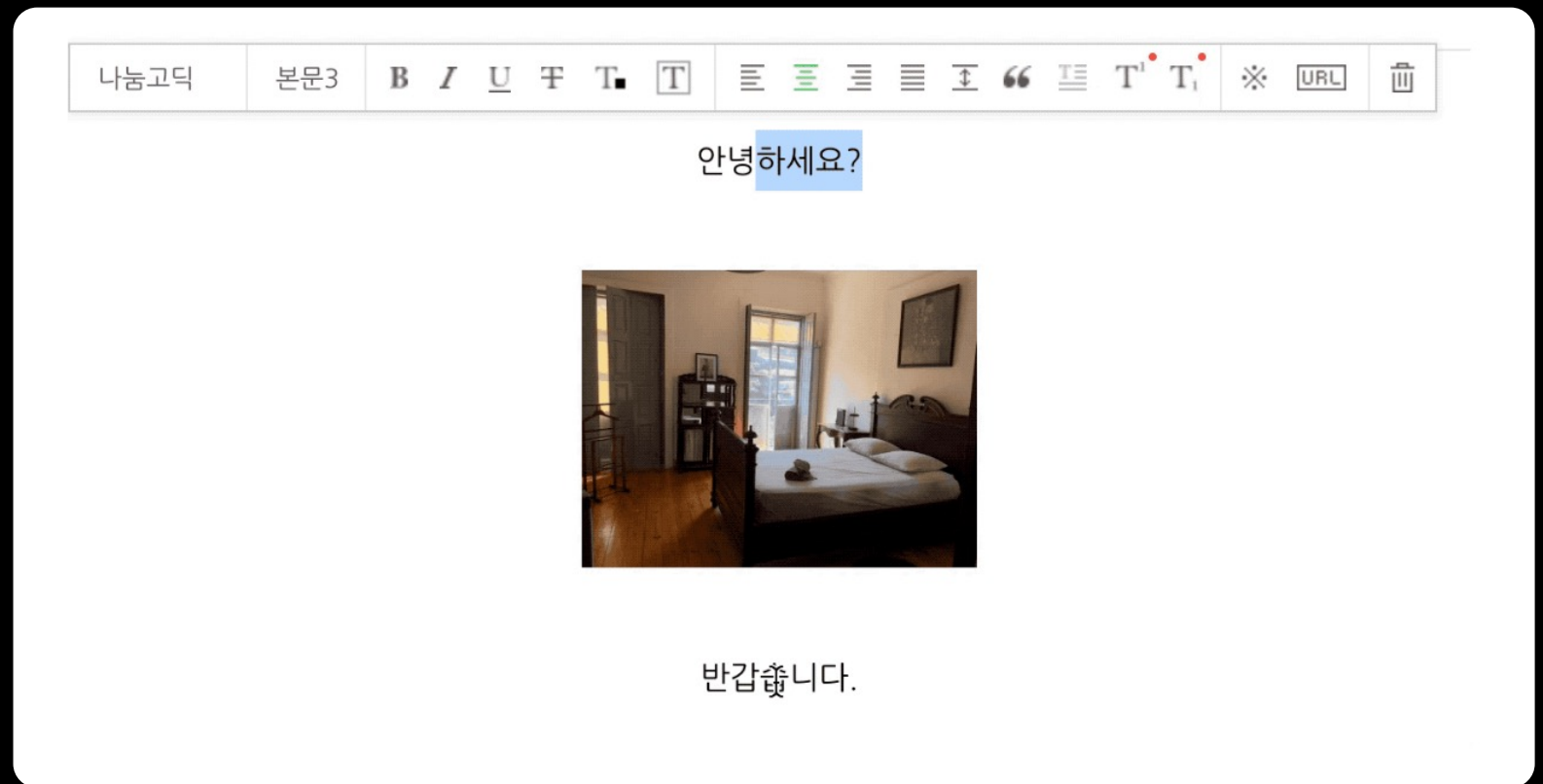
1.5세대 에디터 (SE 3.0)

NAVER
DEVIEW
2023

문제 발생

어색한 Selection 동작

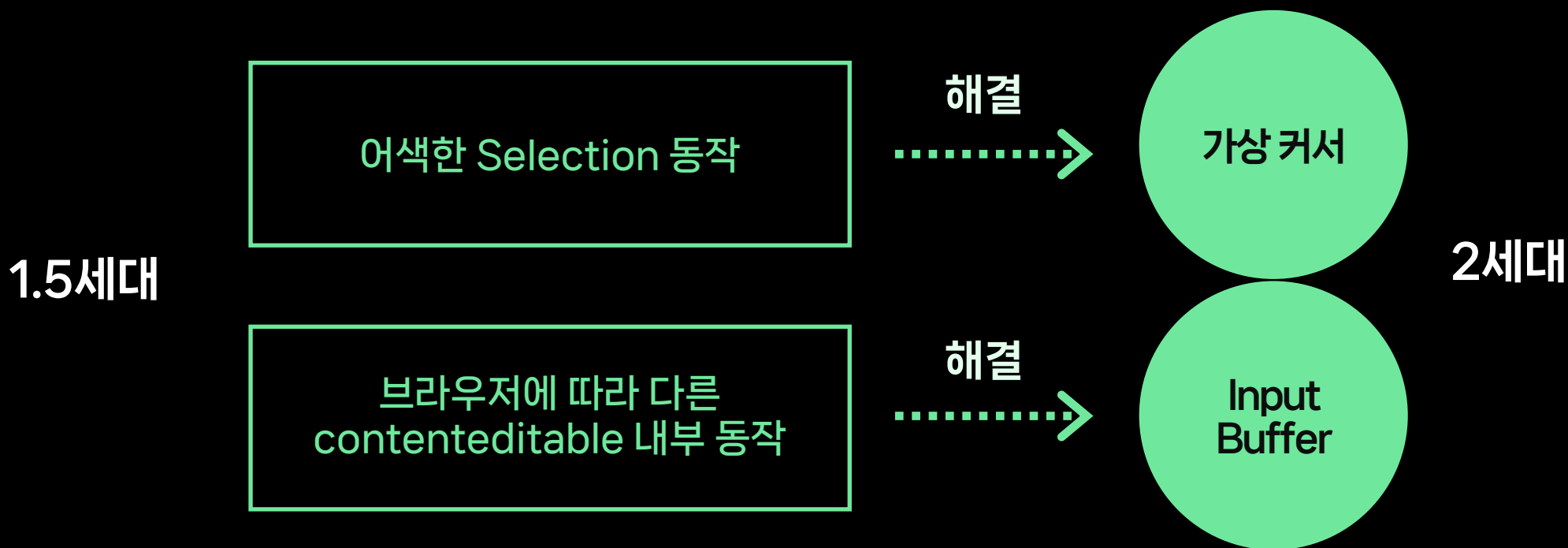
- 다른 영역의 텍스트를
함께 선택할 수 없는 문제



2세대 에디터 (SE ONE)

특징

- 가상커서 + Input Buffer



2세대 에디터 (SE ONE)

가상 커서



Input Buffer

브라우저가 지원하는 커서가 아닌 **직접 만든 커서**

안녕하세요!

HTML

```
<svg class="se-caret se-is-caret-blinking"
      style="top:217px; left:302.109px; width:1px; height:15px;"
      focusable="false">
  <rect width="100%" height="100%"></rect>
</svg>
```

CSS

```
.se-caret.se-is-caret-blinking rect {
  animation: caret-blinking 1s infinite;
}
```

과거부터 현재까지

2세대 에디터 (SE ONE)

NAVER
DEVVIEW
2023

가상 커서



Input Buffer

예시: 키보드 방향키로 커서 이동

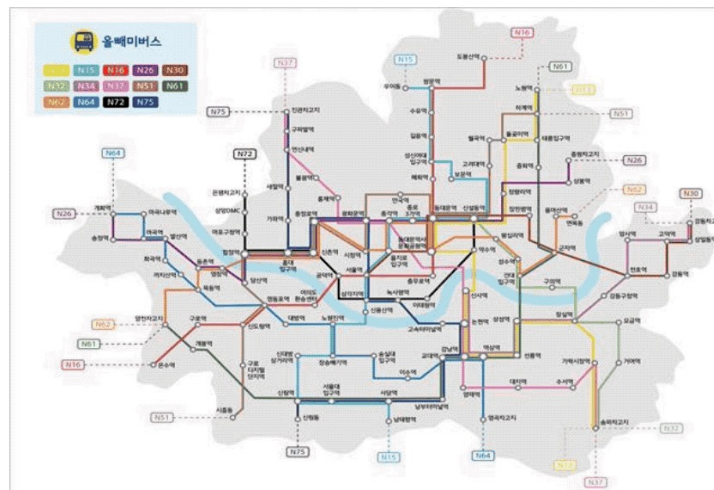
아래 방향키

커서 이동 및 저장

현재 위치 확인

이동할 위치 탐색

10~11일 대중교통 막차 연장...시립묘지 경유 버스 증회



[서울=뉴시스] 이재은 기자 = 서울시는 '추석 연휴 특별교통대책'을 실시한다고 6일 밝혔다.

2세대 에디터 (SE ONE)

가상 커서



Input Buffer

가상 커서가 위치한 곳이 아닌 **화면 밖 실제 입력 영역**

1 사용자 입력

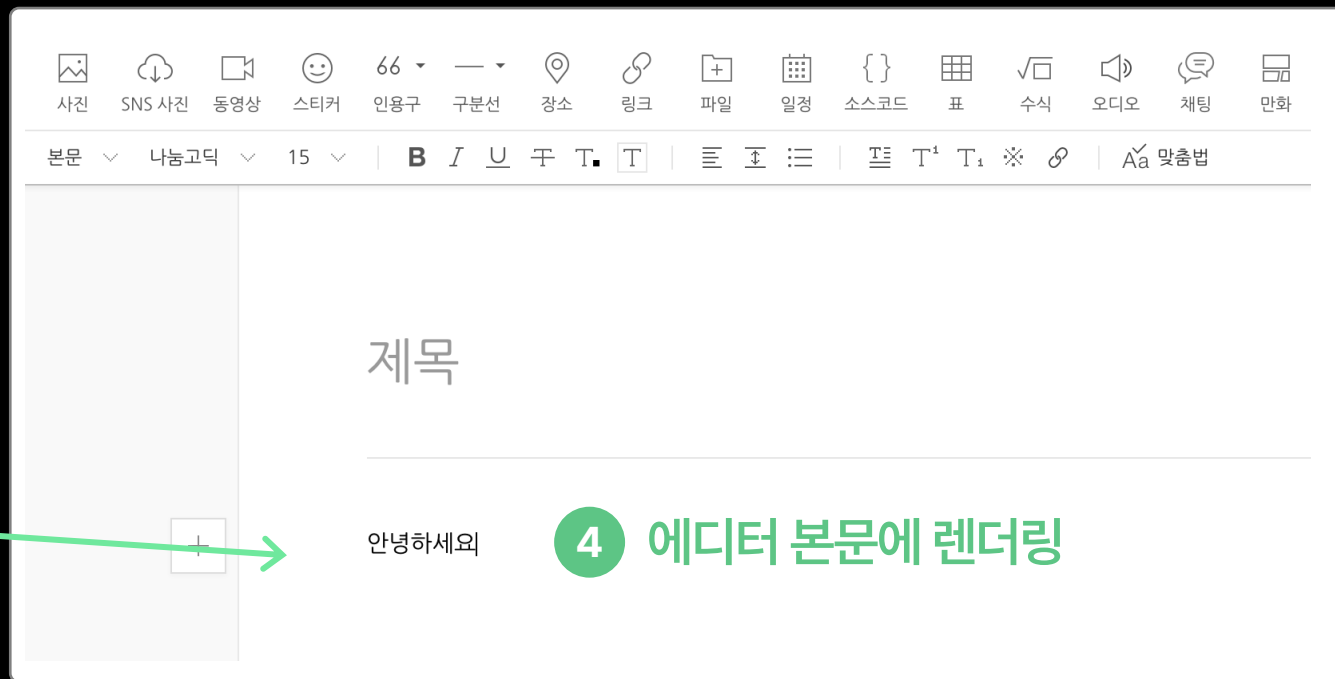


2 Input Buffer에 입력됨

안녕하세요



3 Store가 갱신됨



4 에디터 본문에 렌더링

과거부터 현재까지

2세대 에디터 (SE ONE)

NAVER
DEVVIEW
2023

가상 커서



Input Buffer

Editor

제목을 입력하세요.

5 ▾ | **B** *I* | T. 66 |

DEVVIEW 2023

안녕하세요

“
중요한 것은 꺾이지 않는 마음
”

Input Buffer

```
▼<iframe id="input_buffer1673515411343" tabindex="-1" title=" 스마트 에디터, 접근성 도움말은 alt + 0 " style="position: fixed; top: 0px; width: 50%; border: 0px; z-index: 20; left: 0px;">
  ▼#document
    ▼<html lang="ko">
      ▶<head>...</head>
      ▼<body contenteditable="true" autocapitalize="none" aria-multiline="true" spellcheck="false" aria-hidden="false" style="position: static; transform: rotateX(90deg); display: block;">
        "&#xFEFF;"
        <span data-input-buffer="INPUT_BUFFER_DATA;Mozilla%2F5.0%20(Macintosh%3B%20Intel%20Mac%20OS%20X%2010_15_7)%20AppleWebKit%2F537.36%20(KHTML%2C%20like%20Gecko)%20Chrome%2F108.0.0.0%20Safari%2F537.36;local.editor.naver.com"></span>
        "DEVVIEW 2023" == $0
      </body>
    </html>
  </iframe>
```

2세대 에디터 (SE ONE)

가상 커서



Input Buffer

문제 발생: 브라우저 네이티브 활용이 어려움

- 1 다국어 입력(IME) 처리 이슈
- 2 우 클릭 붙여넣기 이슈

과거부터 현재까지

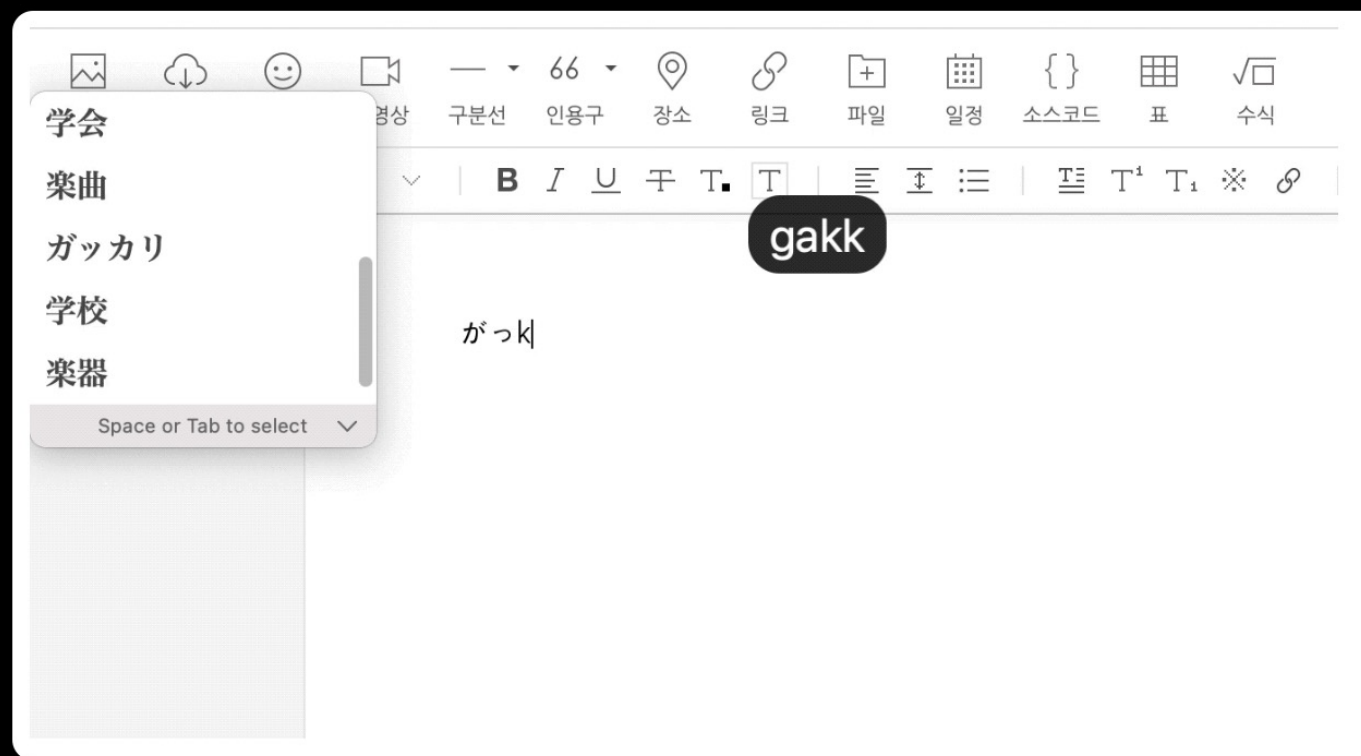
2세대 에디터 (SE ONE)

NAVER
DEVIEW
2023

문제 발생: 브라우저 네이티브 활용이 어려움

1

다국어 이슈



과거부터 현재까지

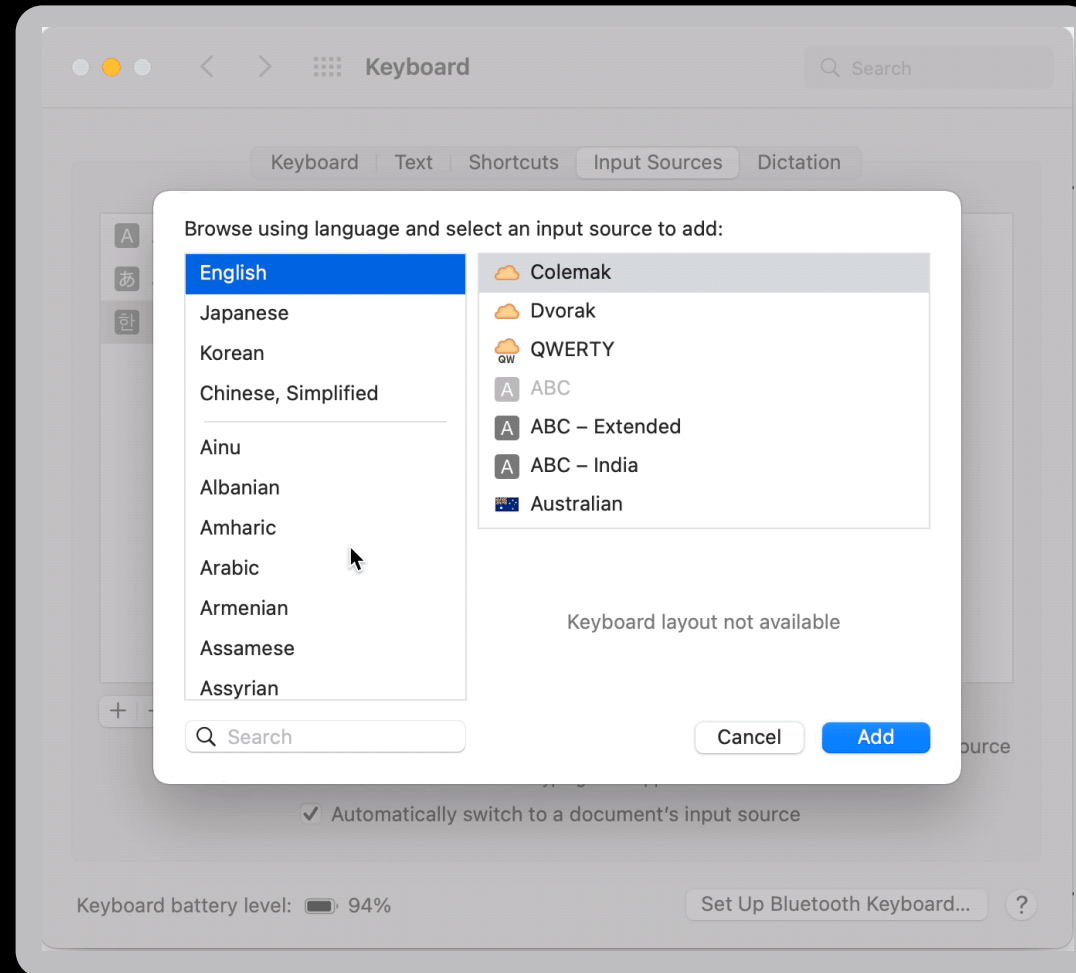
2세대 에디터 (SE ONE)

NAVER
DEVVIEW
2023

문제 발생: 브라우저 네이티브 활용이 어려움

1

다국어 이슈



과거부터 현재까지

2세대 에디터 (SE ONE)

문제 발생: 브라우저 네이티브 활용이 어려움

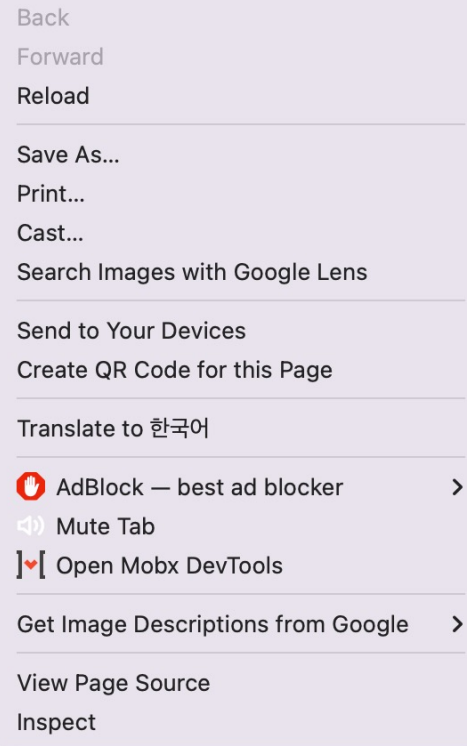
2

우 클릭
붙여넣기 이슈

우 클릭 메뉴에
Paste가 존재하지 않음

NAVER
DEVIEW
2023

지원 대상은 2021년 대학기본역량진단에서 선정된 233개대 가운데 2
학정원 감축과 학부-대학원간 정원 조정, 성인학습자 전담 과정 전환, '이
이다.]



과거부터 현재까지

2세대 에디터 (SE ONE)

NAVER
DEVIEW
2023

문제 발생: 브라우저 네이티브 활용이 어려움

2

우 클릭
붙여넣기 이슈

지원 대상은 2021년 대학기본역량진단에서 선정된 233개대 가운데 정원내 미충원 규모 대비 90% 이상으로 입학정원 감축과 학부-대학... 등 계획을 수립한 대학
이다.

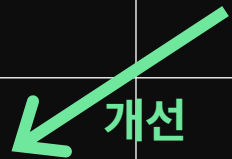
확장 프로그램이 필요합니다.

브라우저에서 클립보드에 접근할 수 없습니다.
붙여넣기를 사용하시려면 확장 프로그램을 설치하세요.

✓ 확장 프로그램 설치

스마트에디터 도전기

	특징	한계
1 세대	• contenteditable 기반 • 문서를 HTML 그대로 저장	• 멀티 디바이스 대응이 어려움
1.5 세대	• 일부 contenteditable 기반 • 문서 모델링(JSON) • 블록 단위 편집	• 어색한 Selection 동작 • 브라우저마다 다른 contenteditable 내부 동작
2 세대	• 가상커서 + input buffer	• 다국어 입력(IME) 처리 • 우 클릭 붙여넣기



2-2

스마트에디터의 현재

3세대 에디터 (SE ONE V2)

목표 ① 기존 에디터의 사용성 유지

- 가상 커서
- 2세대 에디터 기능

목표 ② 브라우저 네이티브 활용 어려움 해소

- 다국어 입력(IME) 처리
- 붙여넣기

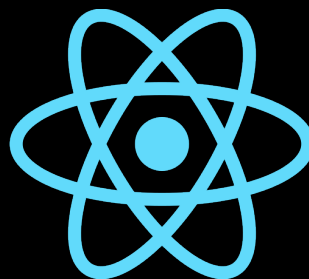
3세대 에디터 (SE ONE V2)

목표 ① 기존 에디터의 사용성 유지

SE ONE 기능 유지

.....

React

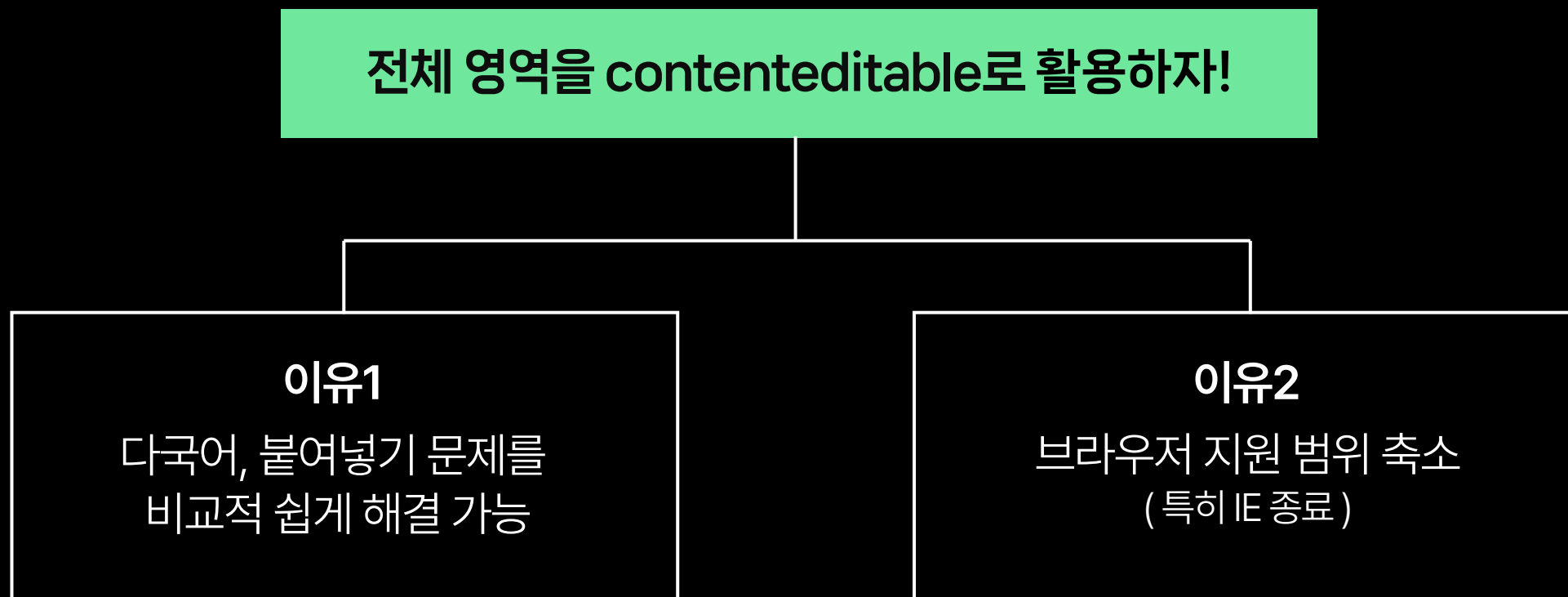


MobX



3세대 에디터 (SE ONE V2)

목표 ② 브라우저 네이티브 활용 어려움 해소



3세대 에디터: 기존 기술과의 충돌

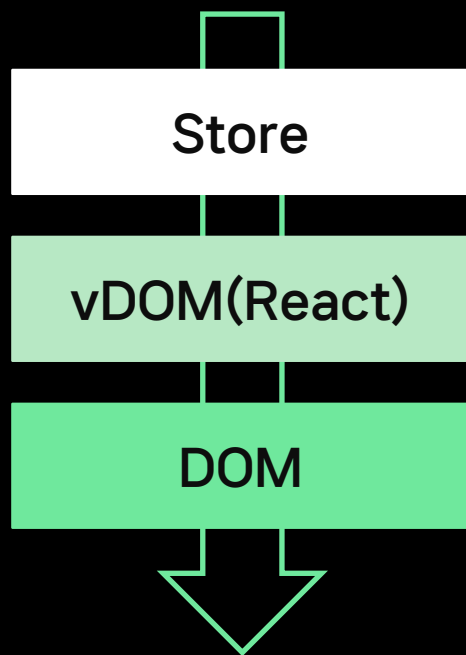
```
✖ ▶ Uncaught DOMException: Failed to execute 'insertBefore' on 'Node': The node before react-dom.development.js:12056
which the new node is to be inserted is not a child of this node.
    at insertBefore (http://localhost:3000/static/js/0.chunk.js:22207:22)
    at insertOrAppendPlacementNode (http://localhost:3000/static/js/0.chunk.js:34811:11)
    at commitPlacement (http://localhost:3000/static/js/0.chunk.js:34755:13)
    at commitReconciliationEffects (http://localhost:3000/static/js/0.chunk.js:35489:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35250:13)
    at recursivelyTraverseMutationEffects (http://localhost:3000/static/js/0.chunk.js:35176:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35196:13)
    at recursivelyTraverseMutationEffects (http://localhost:3000/static/js/0.chunk.js:35176:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35249:13)
    at recursivelyTraverseMutationEffects (http://localhost:3000/static/js/0.chunk.js:35176:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35196:13)
    at recursivelyTraverseMutationEffects (http://localhost:3000/static/js/0.chunk.js:35176:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35474:13)
    at recursivelyTraverseMutationEffects (http://localhost:3000/static/js/0.chunk.js:35176:11)
    at commitMutationEffectsOnFiber (http://localhost:3000/static/js/0.chunk.js:35334:13)
    at commitMutationEffects (http://localhost:3000/static/js/0.chunk.js:35146:7)
    at commitRootImpl (http://localhost:3000/static/js/0.chunk.js:37706:9)
    at commitRoot (http://localhost:3000/static/js/0.chunk.js:37585:9)
    at performSyncWorkOnRoot (http://localhost:3000/static/js/0.chunk.js:37017:7)
    at flushSyncCallbacks (http://localhost:3000/static/js/0.chunk.js:23192:26)
    at http://localhost:3000/static/js/0.chunk.js:36554:17
```

React, MobX와 contenteditable을 함께 사용할 때 오류 발생

3세대 에디터: 기존 기술과의 충돌

React, MobX

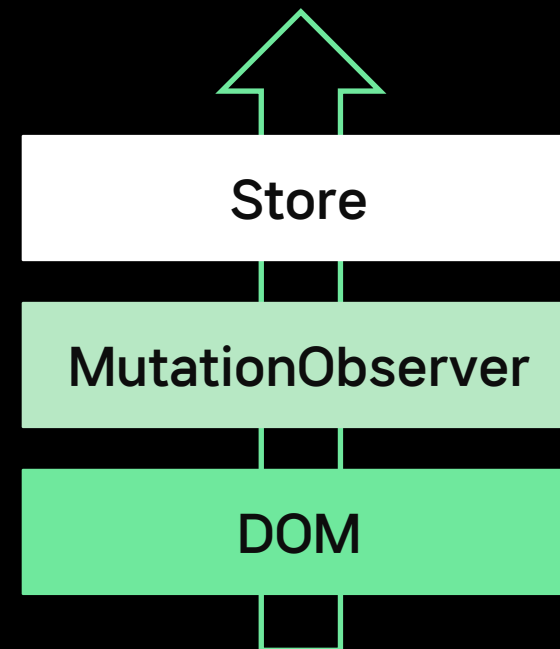
: vDOM을 별도로 관리



React vDOM과 DOM의
싱크가 깨짐

contenteditable

: DOM을 유저가 직접 변경



우리가 흐름을 제어하자!

(특허번호: 10-2022-0073131)

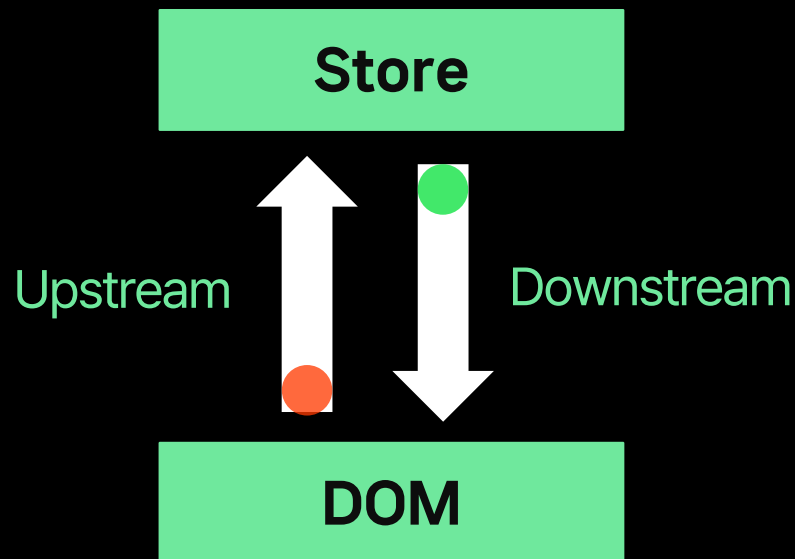
3세대 에디터: 우리가 흐름을 제어하자!

문제

contenteditable과 React를 함께 사용해
DOM과 vDOM 싱크 문제 발생

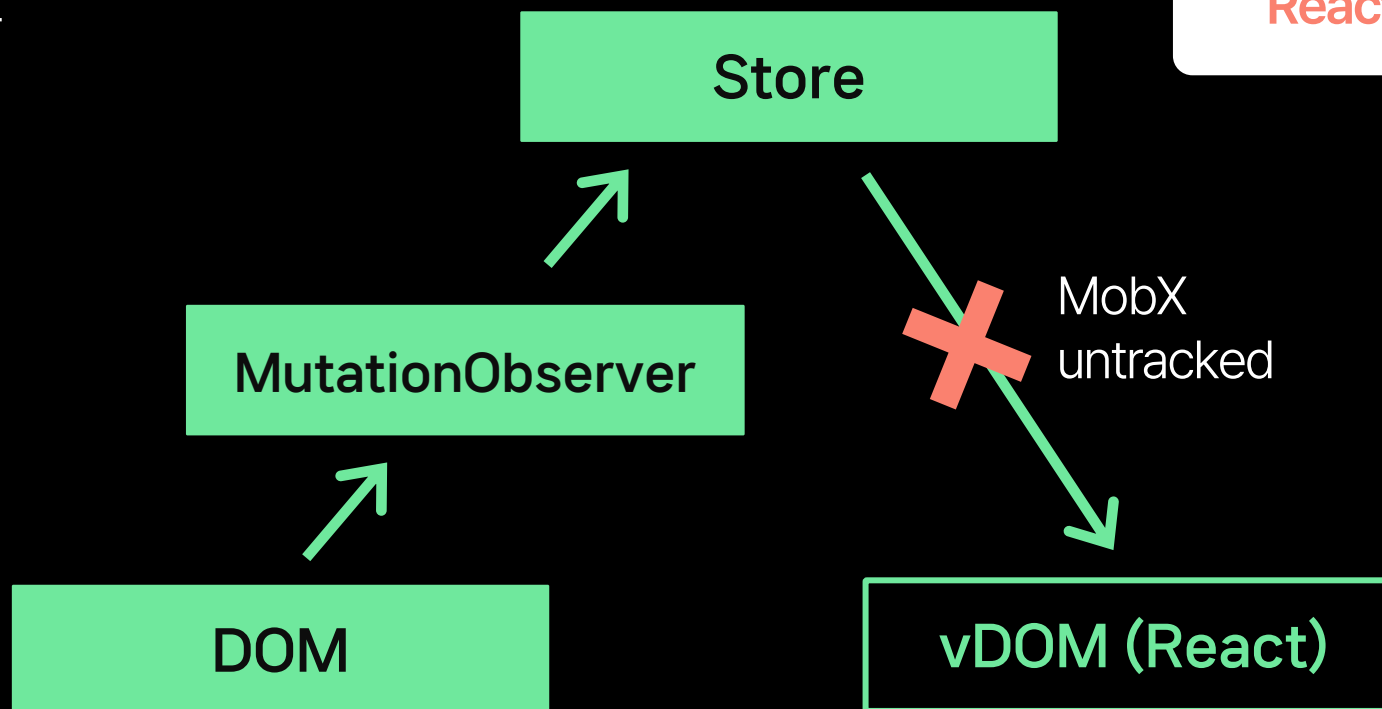
해결

DOM과 vDOM 싱크가 깨질 만한 상황은
나누어 관리하자.



3세대 에디터: Upstream

- DOM 변화에 따라 Store를 업데이트
- MobX의 **untracked API** 를 이용해 React 반응을 임시 중단
- 텍스트 입력 시 사용



contenteditable로
HTML을 수정하는 동안
React 반응을 끄음

3세대 에디터: Upstream

```
const App = observer(() => {
  const contentEditable = useRef(null);

  useEffect(() => {
    const mo = new MutationObserver((mutations) => {
      if (!store.upstreaming) {
        return;
      }

      mutations.forEach(updateStore);
    });

    mo.observe(contentEditable.current, MUTATION_OBSERVER_OPTION);

    return () => {
      mo.disconnect();
    };
  }, []);

  const handleInput = () => {
    store.upstreaming = true;
  };
  const handleKeyDown = (e) => {...};

  return (
    <>
      <Header />
      <div ref={contentEditable}
        contentEditable={true}
        suppressContentEditableWarning={true}
        onInput={handleInput}
        onKeyDown={handleKeyDown}>
        <Content />
      </div>
    </>
  );
});
```

3세대 에디터: Upstream

```
const Content = observer(() => {  
  const render = () => store.lines.map((line, index) => {  
    return (  
      <p key={index}>{line}</p>  
    );  
  });  
  
  if (store.upstreaming) {  
    return untracked(render);  
  }  
  
  return render();  
});
```

3세대 에디터: Upstream

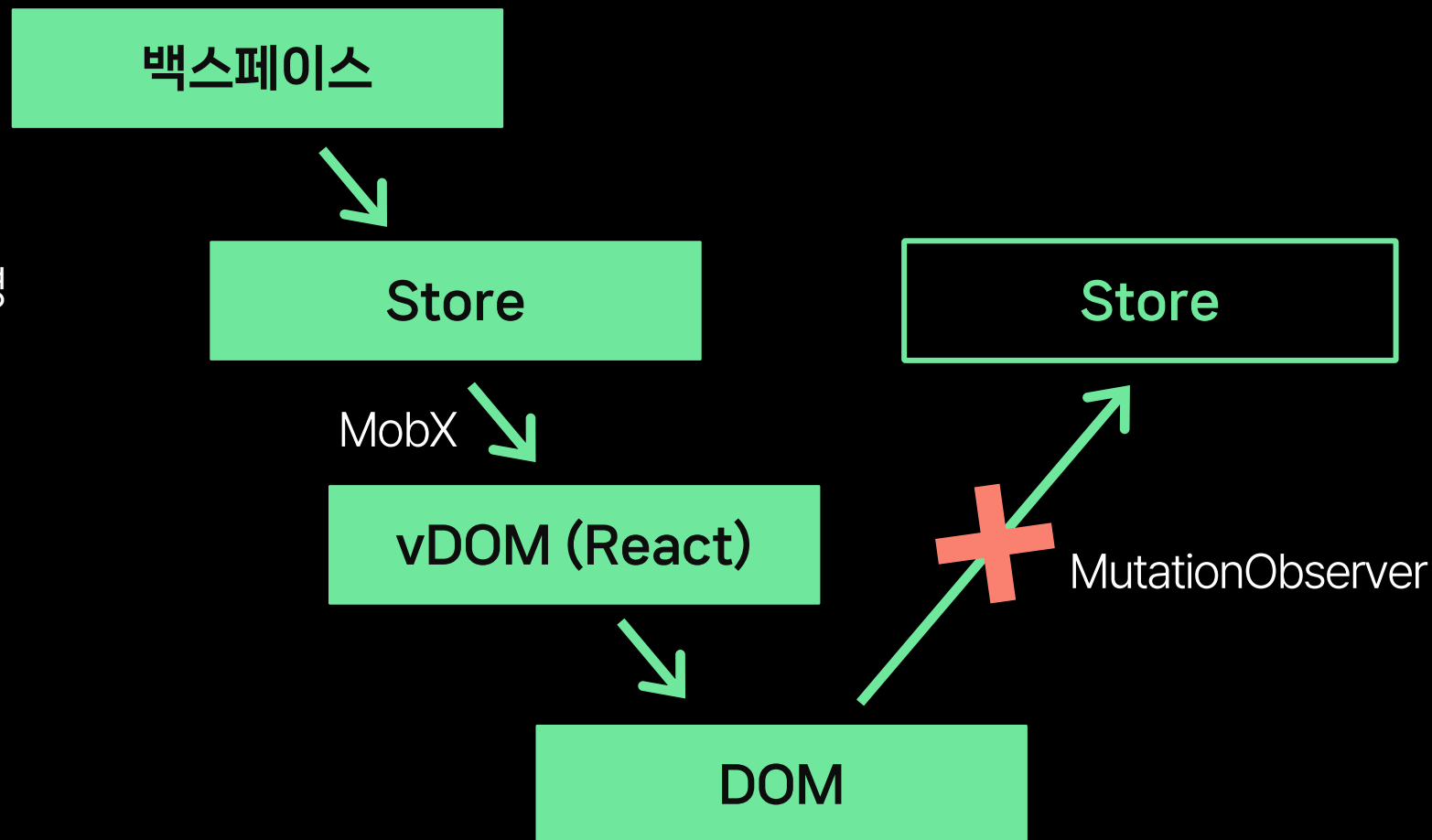
예시: 글을 작성하는 동안 rendering이 발생하지 않음

Upstreaming: **true** / React rendering: **false**

중요한 건 |

3세대 에디터: Downstream

- Store 업데이트 사항을 DOM에 반영
- DOM의 변경사항은 Store에 업데이트 되지 않음
- e.g. 백스페이스, 엔터, 컴포넌트 삽입



3세대 에디터: Downstream

```
const App = observer(() => {
  const contentEditable = useRef(null);

  useEffect(() => {...}, []);

  const handleInput = () => {...};
  const handleKeyDown = (e) => {
    if (e.key !== "Enter") {
      return;
    }

    const [index, offset] = getCurrentPosition();

    store.splitLine(index, offset);
    updateSelection(index);
    e.preventDefault();

    store.upstreaming = false;
  };

  return (
    <>
      <Header />
      <div ref={contentEditable}
        contentEditable={true}
        suppressContentEditableWarning={true}
        onInput={handleInput}
        onKeyDown={handleKeyDown}>
        <Content />
      </div>
    </>
  );
});
```

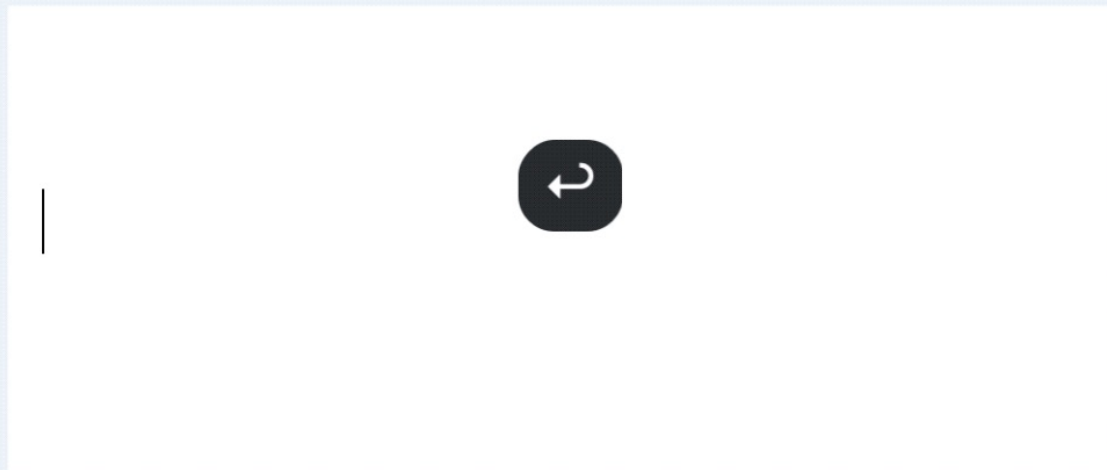
3세대 에디터: Downstream

```
const Content = observer(() => {  
  const render = () => store.lines.map((line, index) => {  
    return (  
      <p key={index}>{line}</p>  
    );  
  });  
  
  if (store.upstreaming) {  
    return untracked(render);  
  }  
  
  return render();  
});
```

3세대 에디터: Downstream

예시: Enter 입력 시 rendering 발생

Upstreaming: **false** / React rendering: **true**



3세대 에디터: Upstream & Downstream

예시: Upstream, Downstream이 복합적으로 발생

Upstreaming: **true** / React rendering: **false**

중요한 건

Upstreaming: **false** / React rendering: **true**

중요한 건 꺾이지 않는

마음

3세대 에디터의 시도

기존 에디터 문제 발생
(다국어, 붙여넣기)

↓ 해결 !

브라우저 네이티브 활용
(contenteditable)

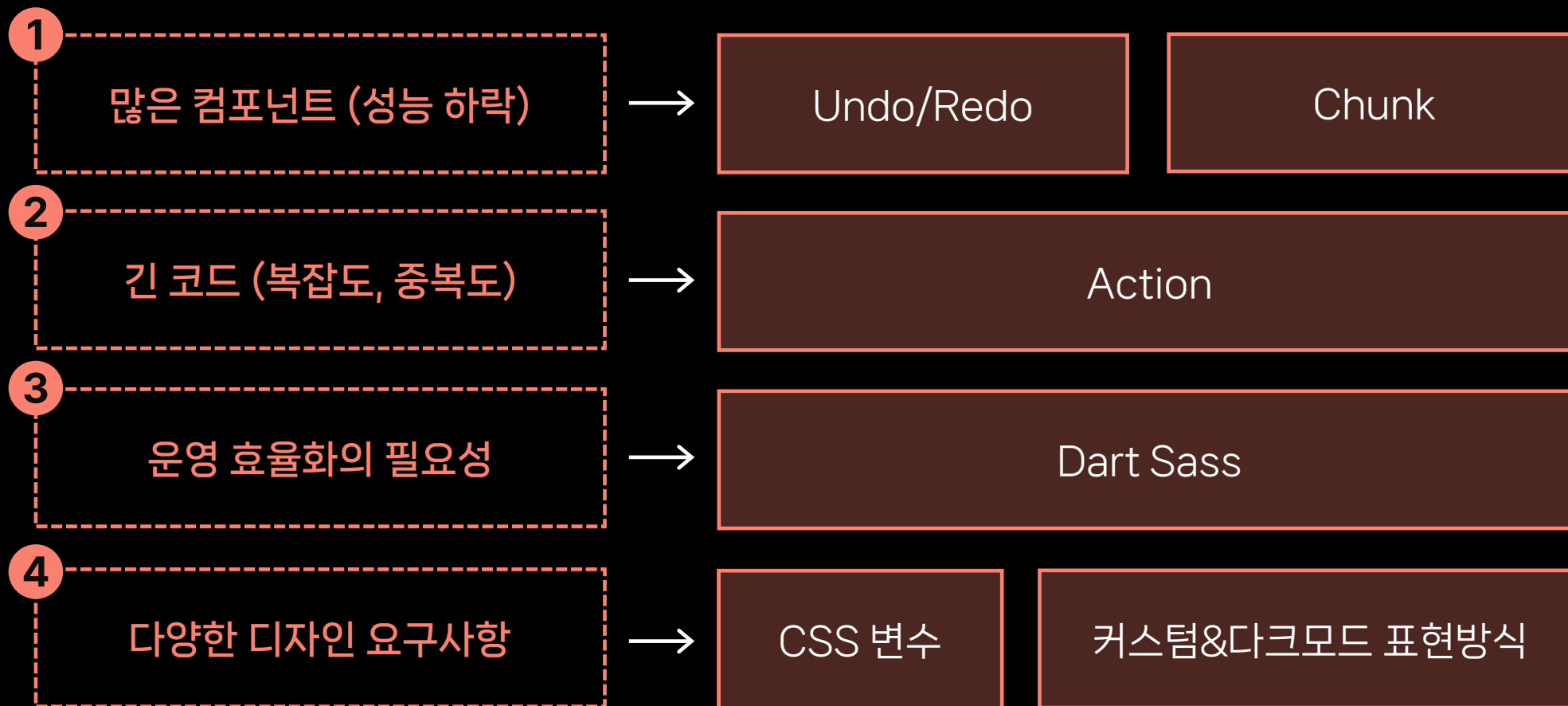
다른 문제 발생

기술 충돌
(React vs contenteditable)

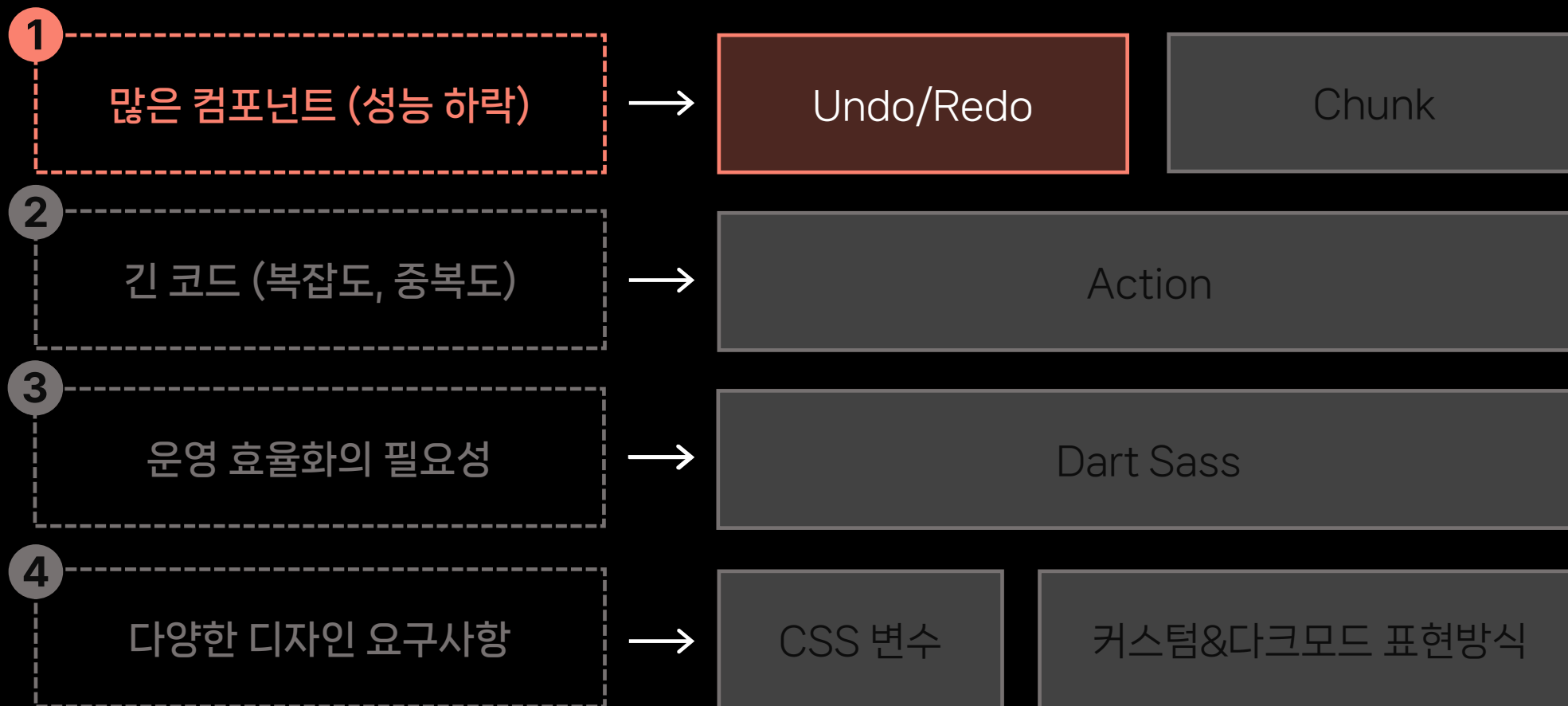
↓ 해결 !

upstream &
downstream

다른 앱에도 적용 가능한 기술



다른 앱에도 적용 가능한 기술



많은 컴포넌트 성능 하락 이슈 해결하기 1

Undo/Redo: 전체 렌더링

NAVER
DEVVIEW
2023

redo 시 3초 소요됨

CPU 6x slowdown

2015년 3월, 혼자 [redacted] (단 한명!!) 한 달 네팔여행을 다녀왔다.

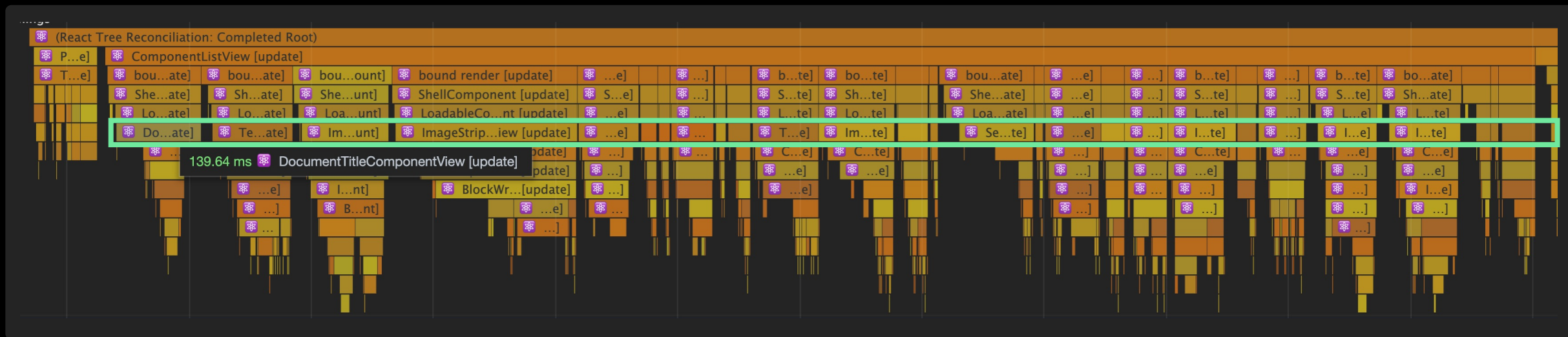
10박 11일 동안 안나푸르나 베이스캠프에 올랐다. (어떤 날은 할 일 없이 골목을 서성였다.) 바쁘게 다니는 여행 대신 느리게 쉬는 여행을 택했다. 짐을 얻고 돌아온 여행이었지만, 그 끝은 슬펐다. 한국에 돌아오고 2주 뒤 네팔은 지진의 슬픔에 잠겼다. 그래도 네팔이 살면서 한 번쯤 가봐야 할 곳임에는 변함이 없다. 30일간의 이야기를 전한다.



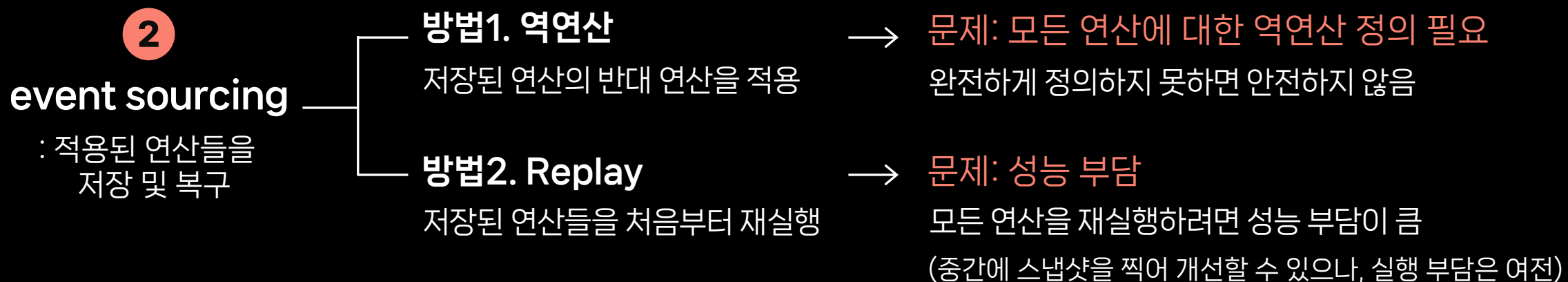
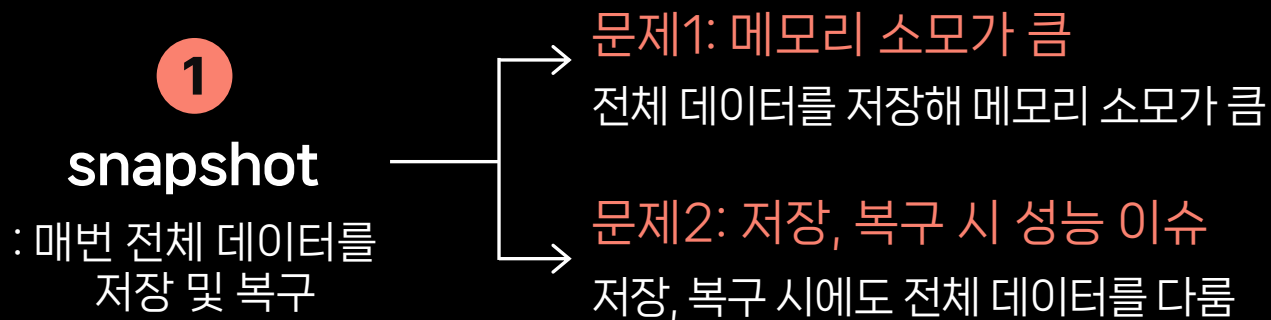
사진 설명을 입력하세요.

Undo/Redo: 전체 렌더링

Snapshot 방식의 경우, undo/redo 시 모든 컴포넌트 렌더링 발생



기존 방식: History 관리



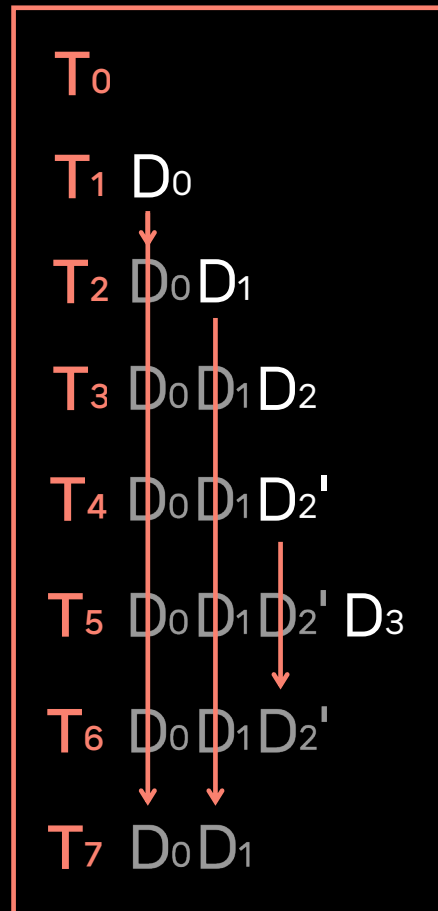
변경된 부분만 **undo, redo**하자!

(특허번호 10-2022-0136713)

Undo/Redo

- 부분 snapshot 방식
- 데이터의 **변경점만** 기록

History



D_0 데이터 추가

D_1 데이터 추가 D_0 데이터는 기록 X
(메모리 주소만 기록)

D_2 데이터 추가

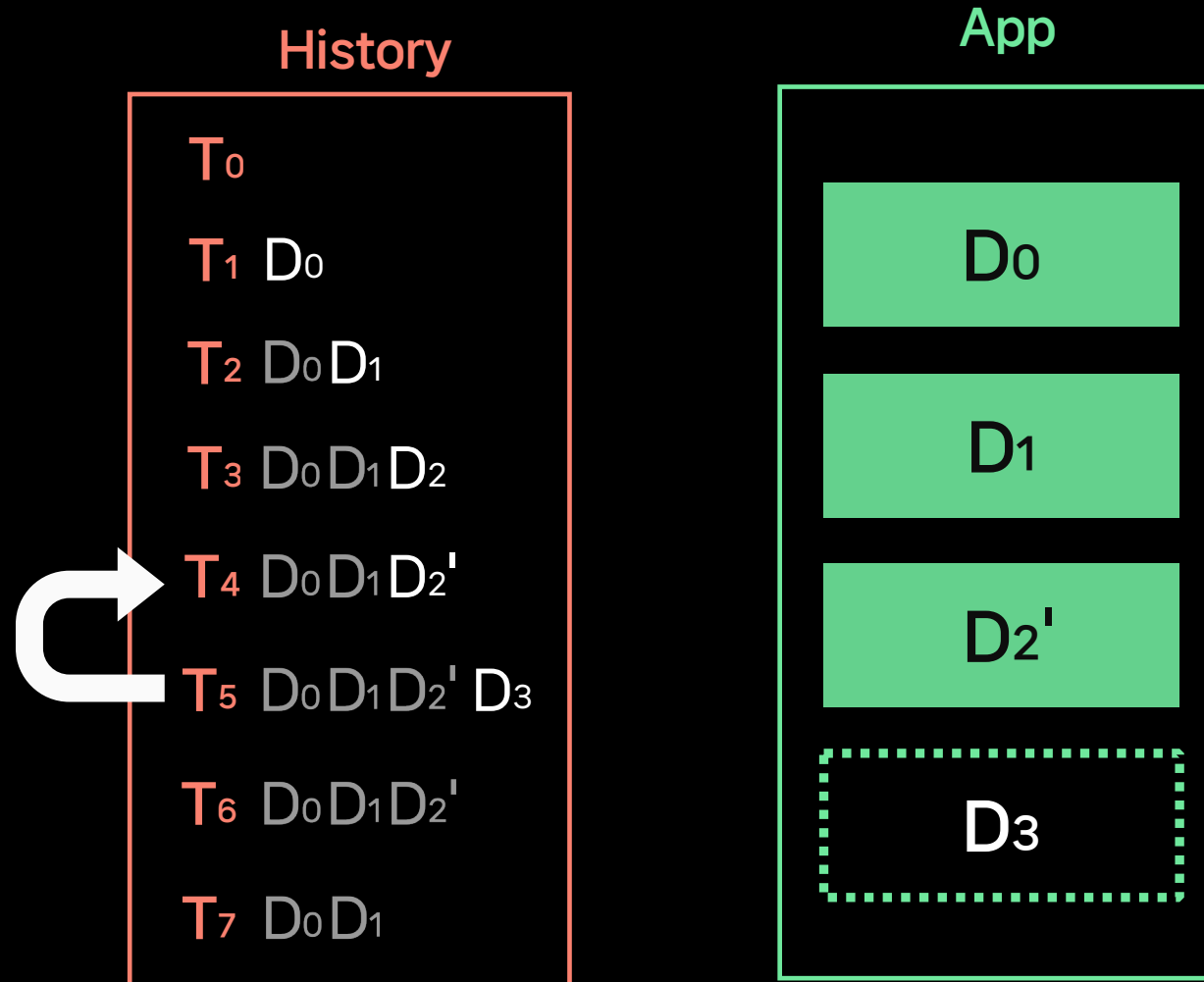
D_2 데이터 **변경** D_2 대신 D_2' 기록

D_3 데이터 추가

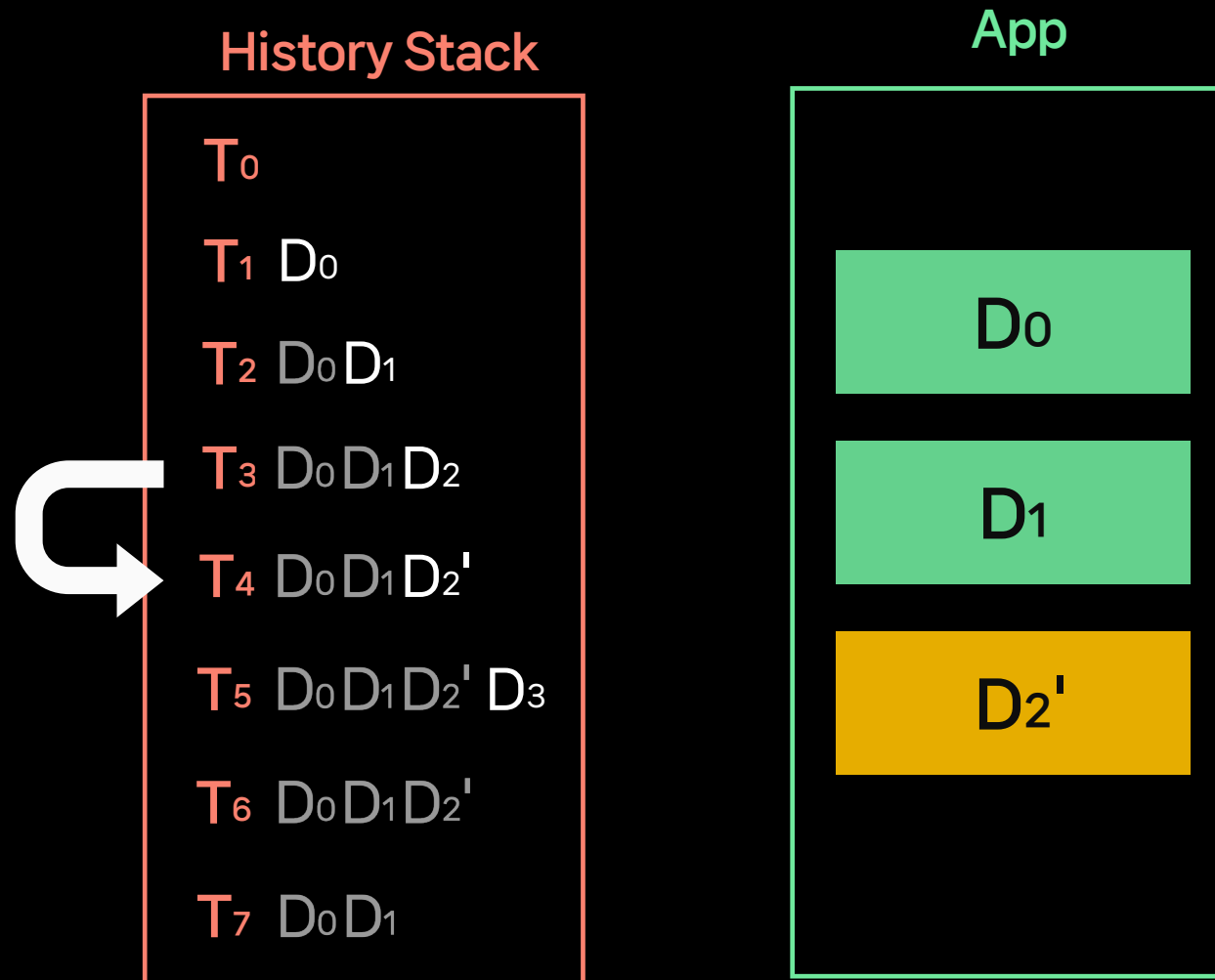
D_3 데이터 **삭제** 메모리 주소값들로 구성된
이력만 남아있음

D_2' 데이터 **삭제**

Undo



Redo



많은 컴포넌트 성능 하락 이슈 해결하기 1

Undo/Redo: 변경사항만 렌더링

NAVER
DEVVIEW
2023

redo 시 0.2초 소요됨
(기존 3초 소요)

CPU 6x slowdown

2015년 3월, 혼자  (단 한명!!) 한 달 네팔여행을 다녀왔다.

10박 11일 동안 안나푸르나 베이스캠프에 올랐다. (어떤 날은 할 일 없이 골목을 서성였다.) 바쁘게 다니는 여행 대신 느리게 쉬는 여행을 택했다. 짐을 얻고 돌아온 여행이었지만, 그 끝은 슬펐다. 한국에 돌아오고 2주 뒤 네팔은 지진의 슬픔에 잠겼다. 그래도 네팔이 살면서 한 번쯤 가봐야 할 곳임에는 변함이 없다. 30일간의 이야기를 전한다.



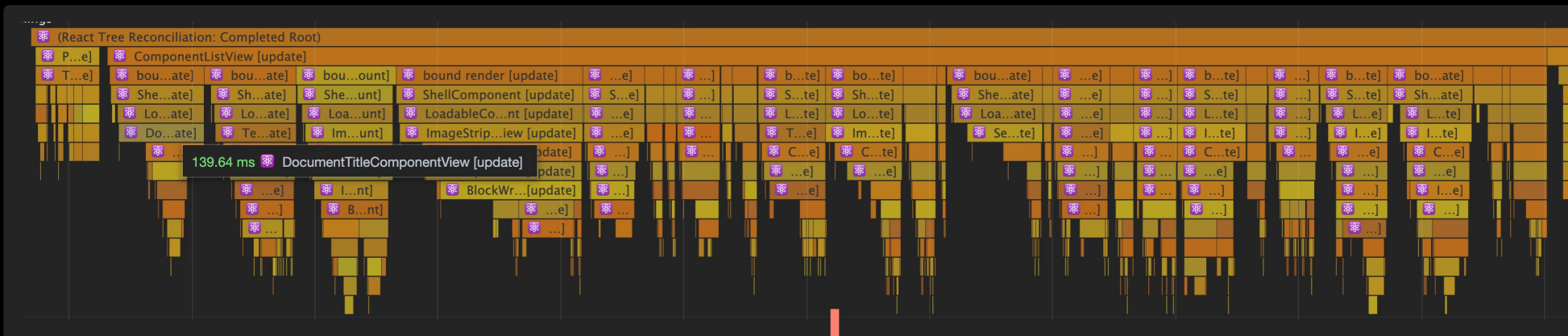
사진 설명을 입력하세요.

많은 컴포넌트 성능 하락 이슈 해결하기 1

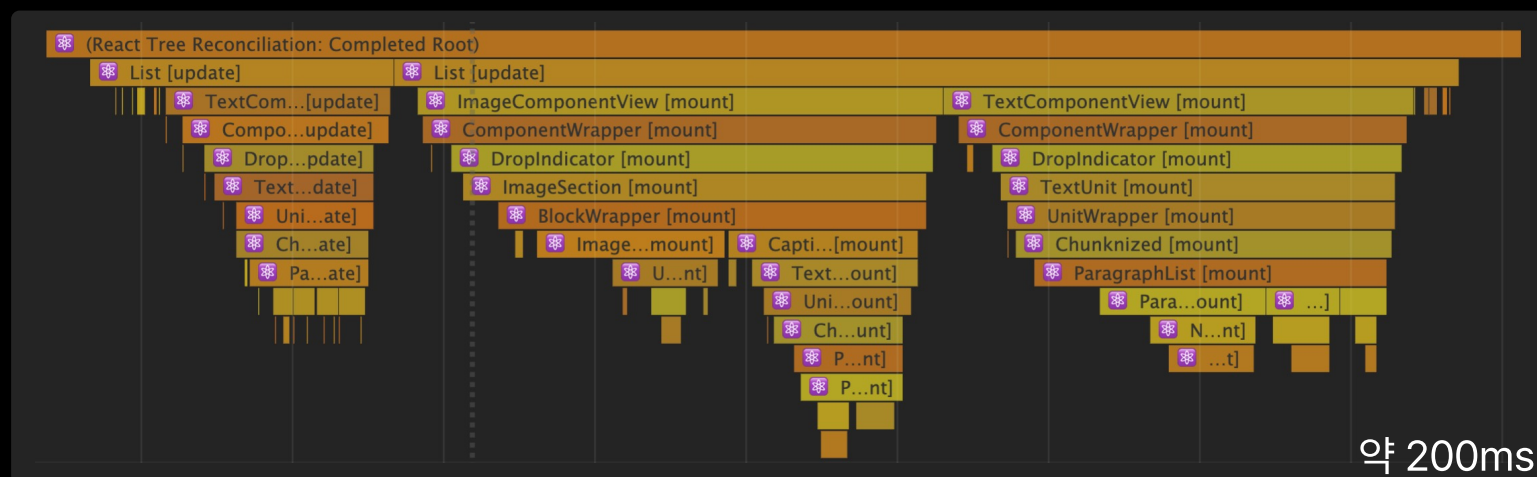
Undo/Redo: 변경사항만 렌더링

NAVER
DEVVIEW
2023

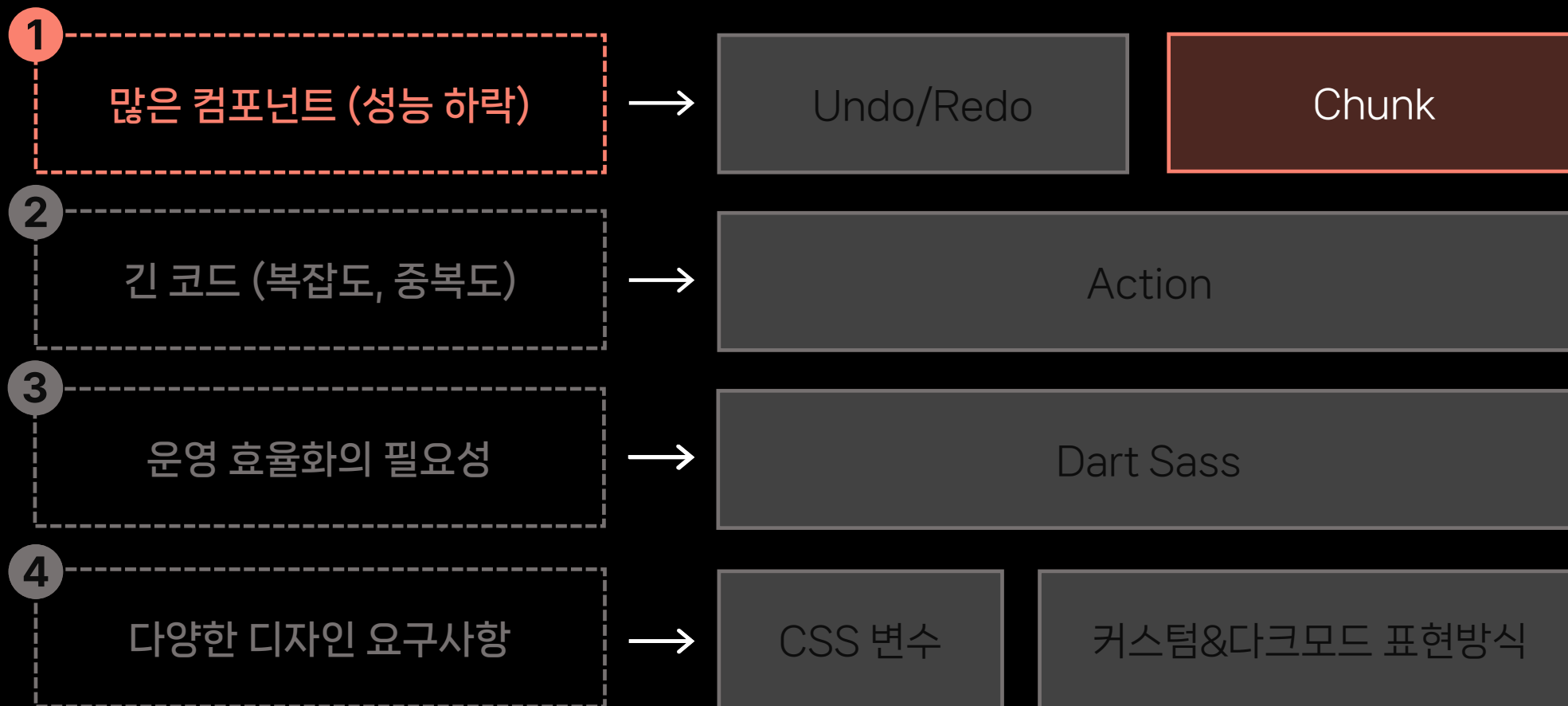
CPU 6x slowdown



부분 Snapshot 적용 ↓



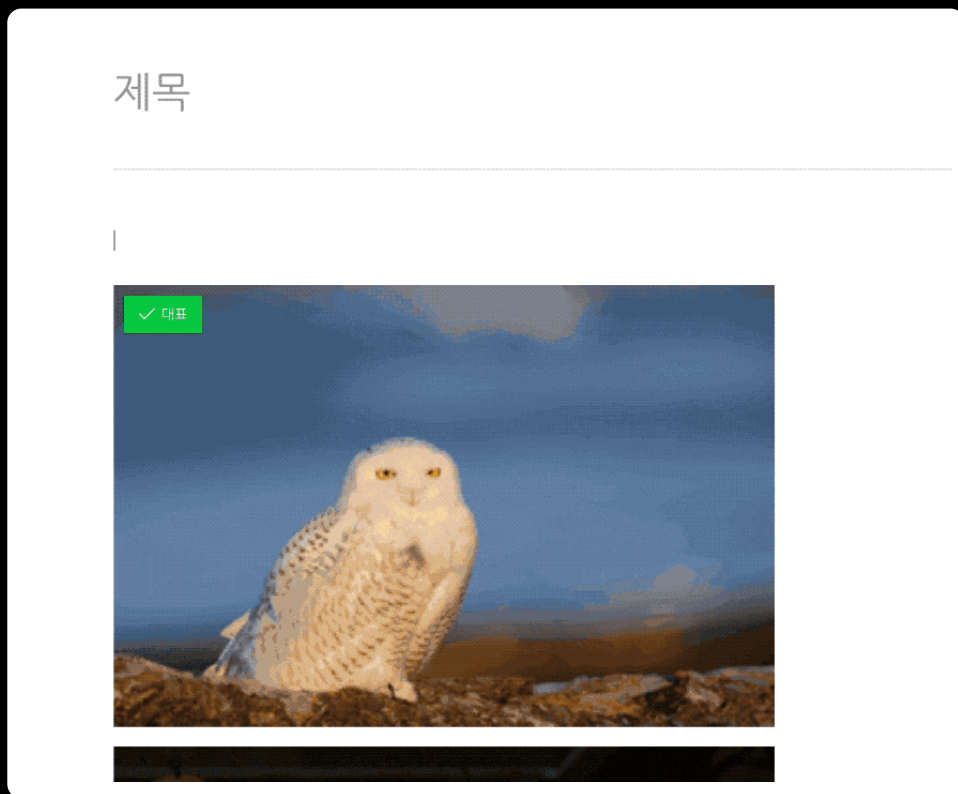
다른 앱에도 적용 가능한 기술



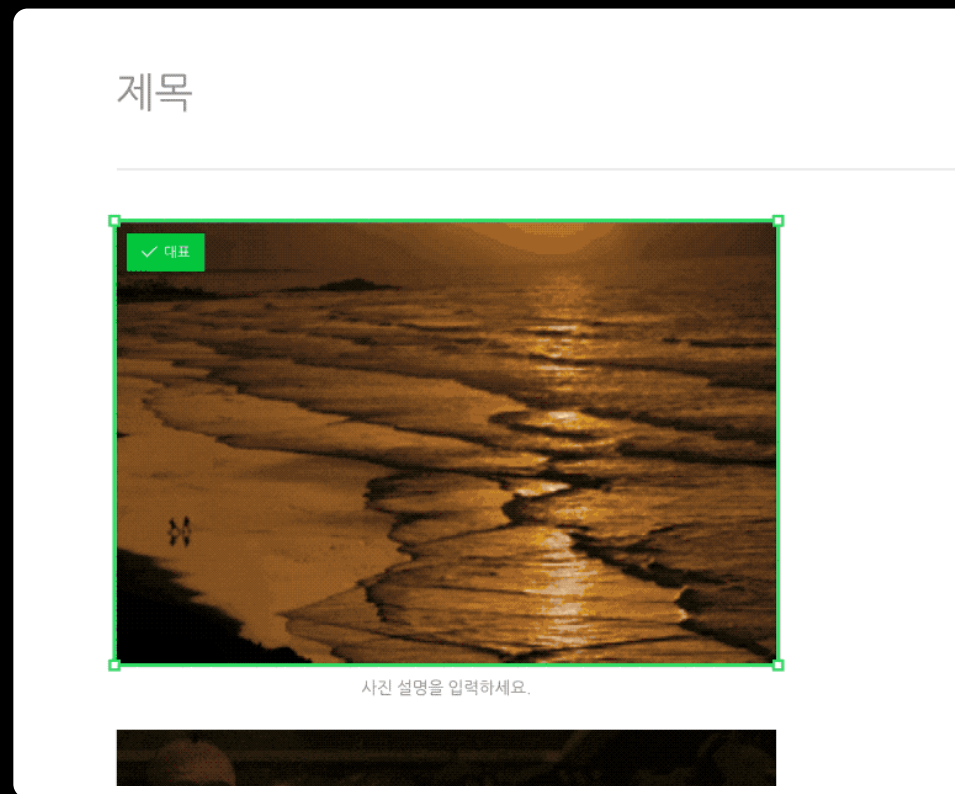


문제2. 컴포넌트 추가/삭제 시 성능이슈

글 입력 2000ms 소요



컴포넌트 추가 3000ms 소요



1000 gif Components Rendering
CPU 6x slowdown



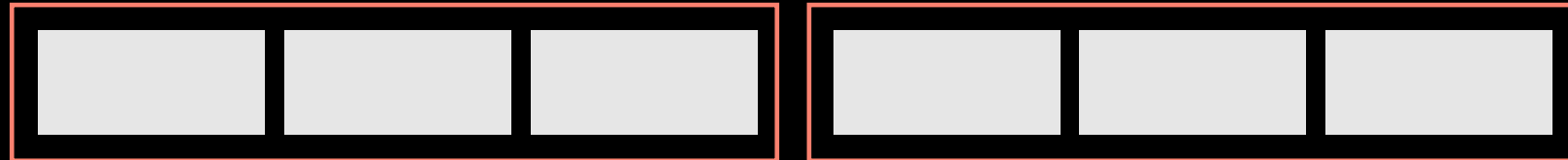
1000 gif components Rendering
CPU 6x slowdown

Chunk

1차 Array



2차 Array
정적 Chunk

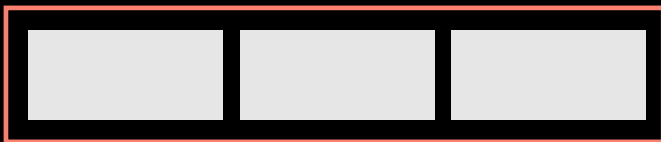


정적 Chunk

초기 로딩 성능 개선



```
class LazyChunk extends React.PureComponent {  
  componentDidUpdate(prevProps) {  
    const {isShow, onRequireNextItem} = this.props;  
  
    if (!prevProps.isShow && isShow) {  
      onRequireNextItem();  
    }  
  }  
  
  render() {  
    const {isShow, _onRequireNextItem, ...props} = this.props;  
  
    return isShow ? <Component {...props} /> : null;  
  }  
}
```



100 ms



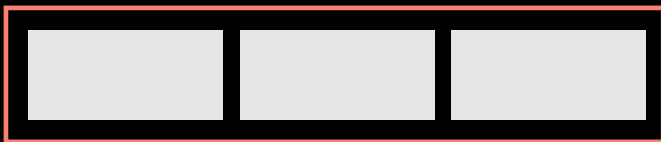
isShow = false

정적 Chunk

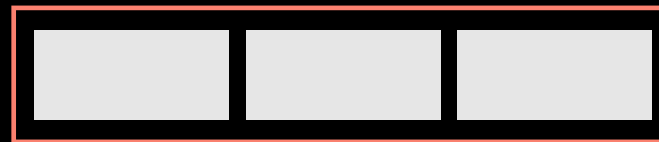
초기 로딩 성능 개선



```
class LazyChunk extends React.PureComponent {  
  componentDidUpdate(prevProps) {  
    const {isShow, onRequireNextItem} = this.props;  
  
    if (!prevProps.isShow && isShow) {  
      onRequireNextItem();  
    }  
  }  
  
  render() {  
    const {isShow, _onRequireNextItem, ...props} = this.props;  
  
    return isShow ? <Component {...props} /> : null;  
  }  
}
```



100 ms



isShow = true

정적 Chunk

컴포넌트 추가/수정/삭제 성능 개선

```
class AutoReplace extends React.Component {
  render() {
    return (
      <div style={this._outerStyle} ref={this._ref}>
        <div style={this._innerStyle}>{this.props.children}</div>
      </div>
    );
  }
  _ref = React.createRef();

  get _innerStyle() {
    return {
      display: this._shown ? "" : "none",
    };
  }

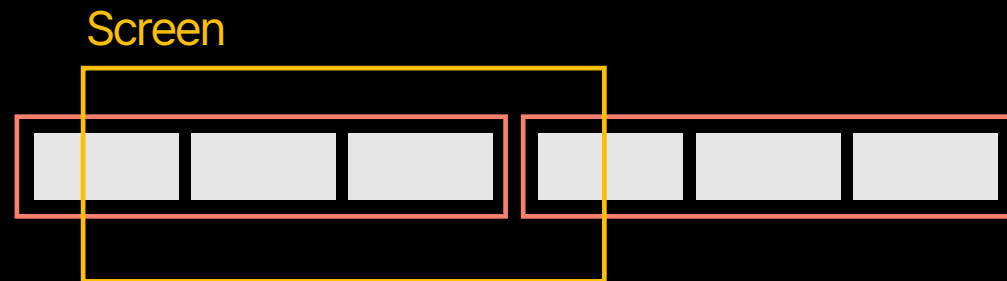
  get _outerStyle() {
    return this._shown ? {} : {height: this._height};
  }

  get _shown() {
    const {alwaysShow, isIntersection} = this.props;

    return alwaysShow || isIntersection;
  }
  _height = 0;
}
```

HOC (Chunk, Lazy, Autoreplace 적용)

```
const ChunknizedObservedPage = withChunknize(
  withLazyPaginize(
    AutoReplaceList
  )
);
```

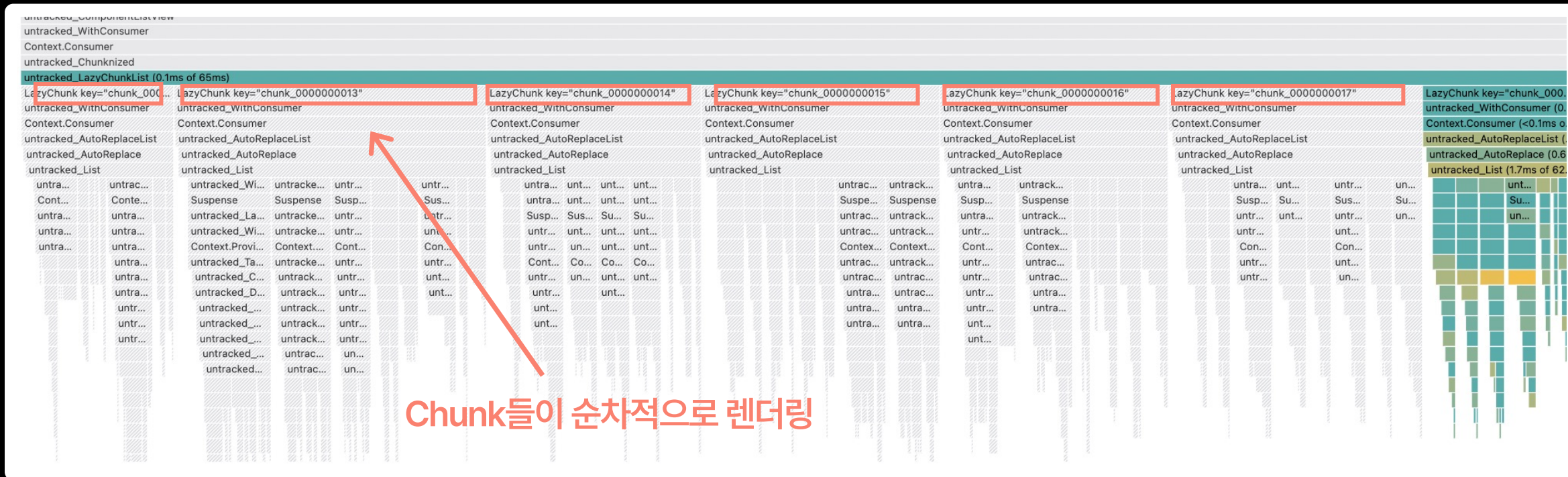


많은 컴포넌트 성능 하락 이슈 해결하기 2

정적 Chunk

NAVER
DEVVIEW
2023

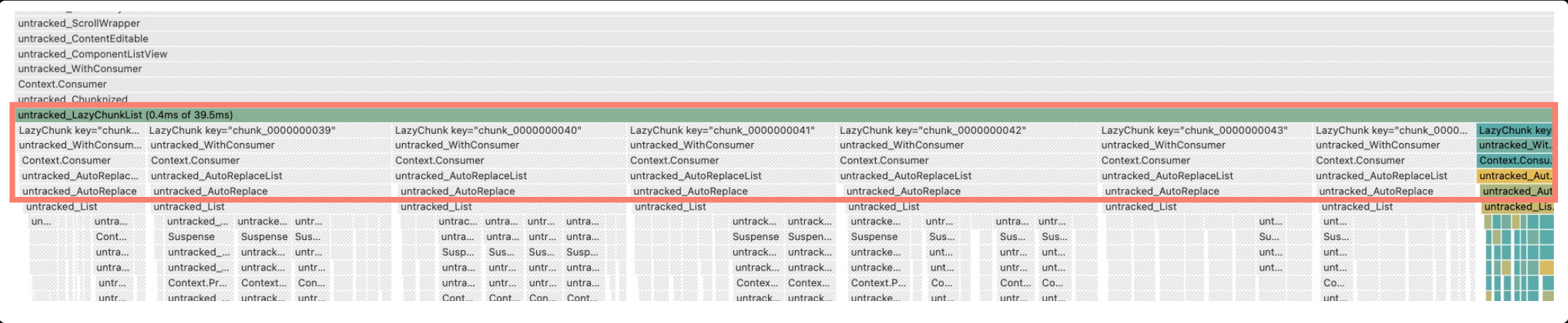
정적 Chunk 도입 이후



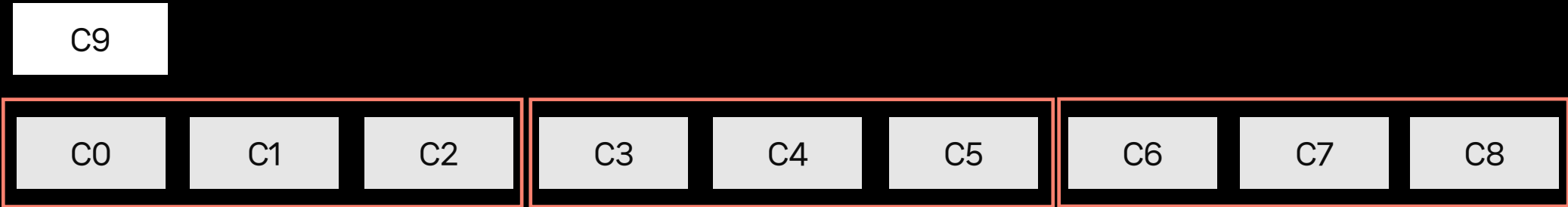
65 Components Rendering
CPU 6x slowdown

정적 Chunk

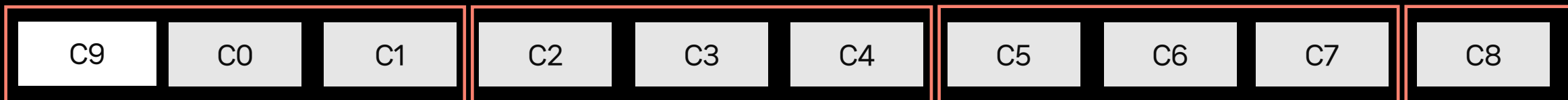
문제: 컴포넌트 추가 시, 전체 컴포넌트 렌더링



CPU 6x slowdown



CPU 6x slowdown

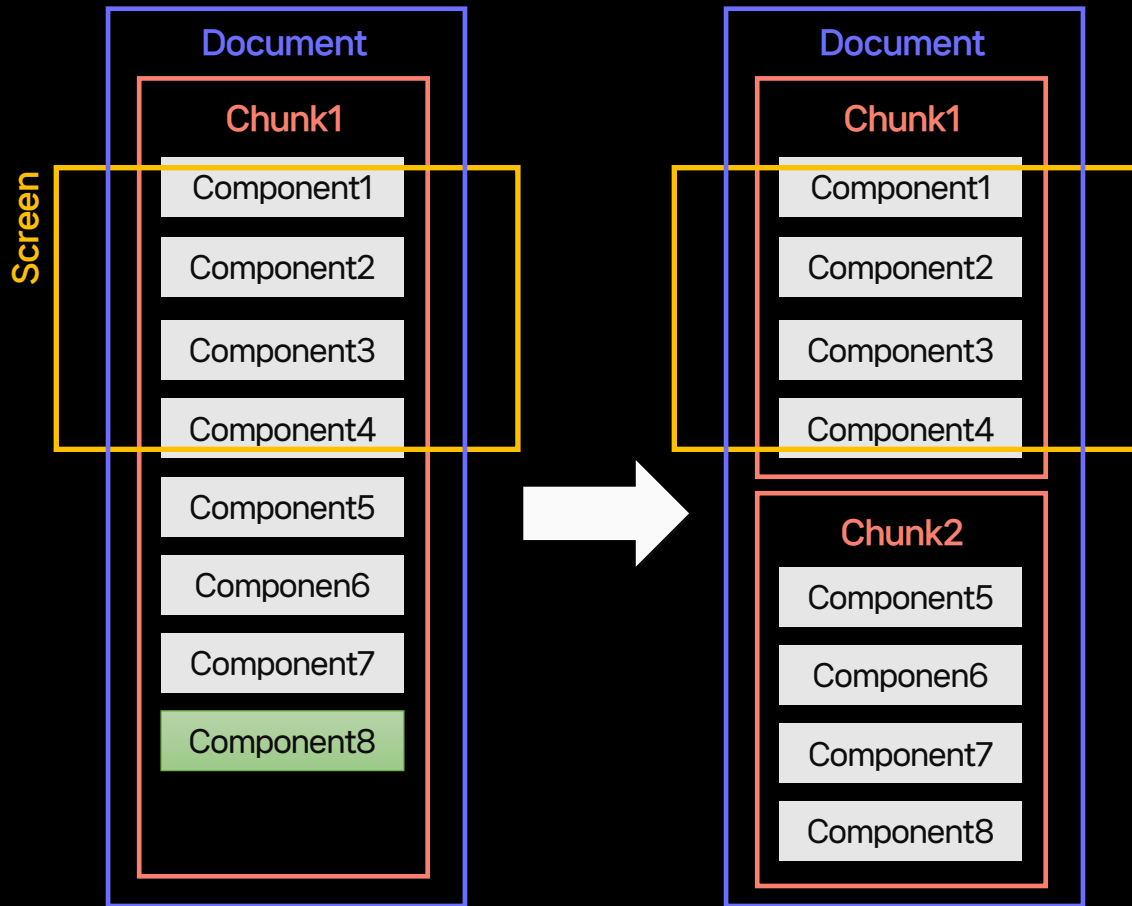


Chunk 크기를 동적으로 하자!

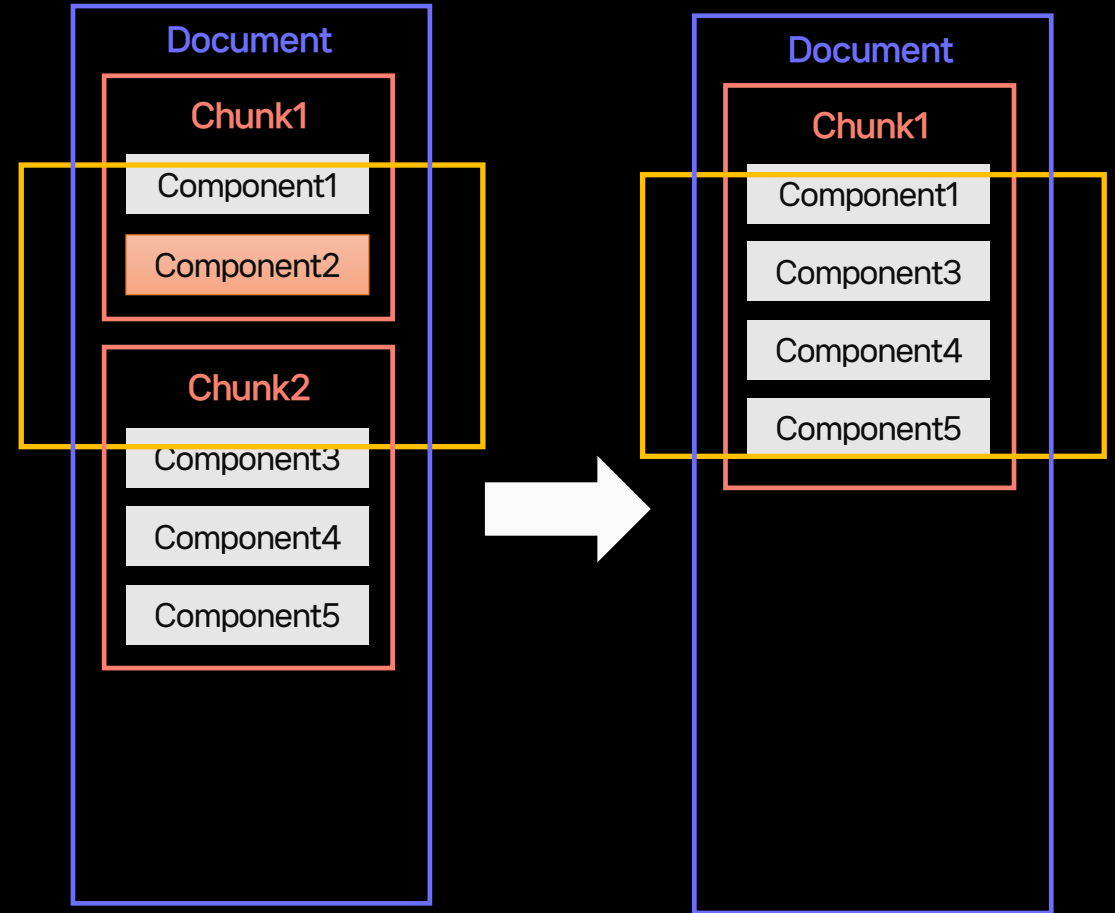
(특허번호: 10-2022-0103939)

동적 Chunk

<특정개수 이상인 경우 나누어짐>

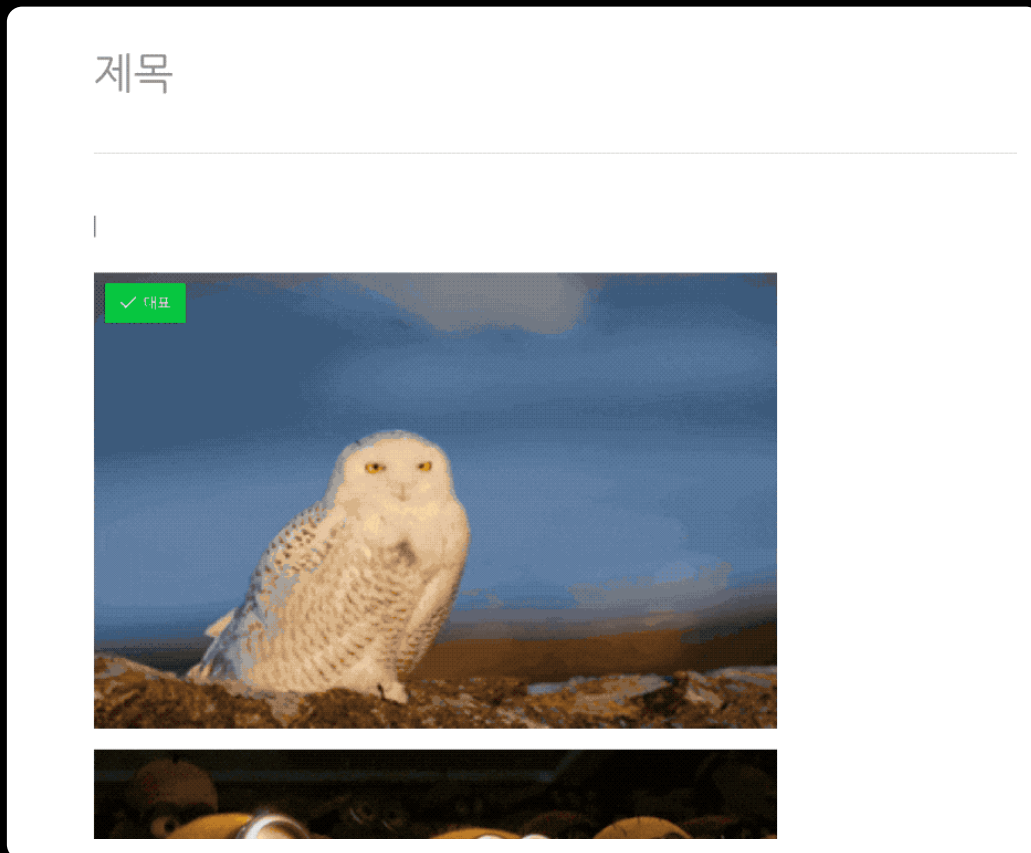


<특정개수 미만인 경우 합쳐짐>

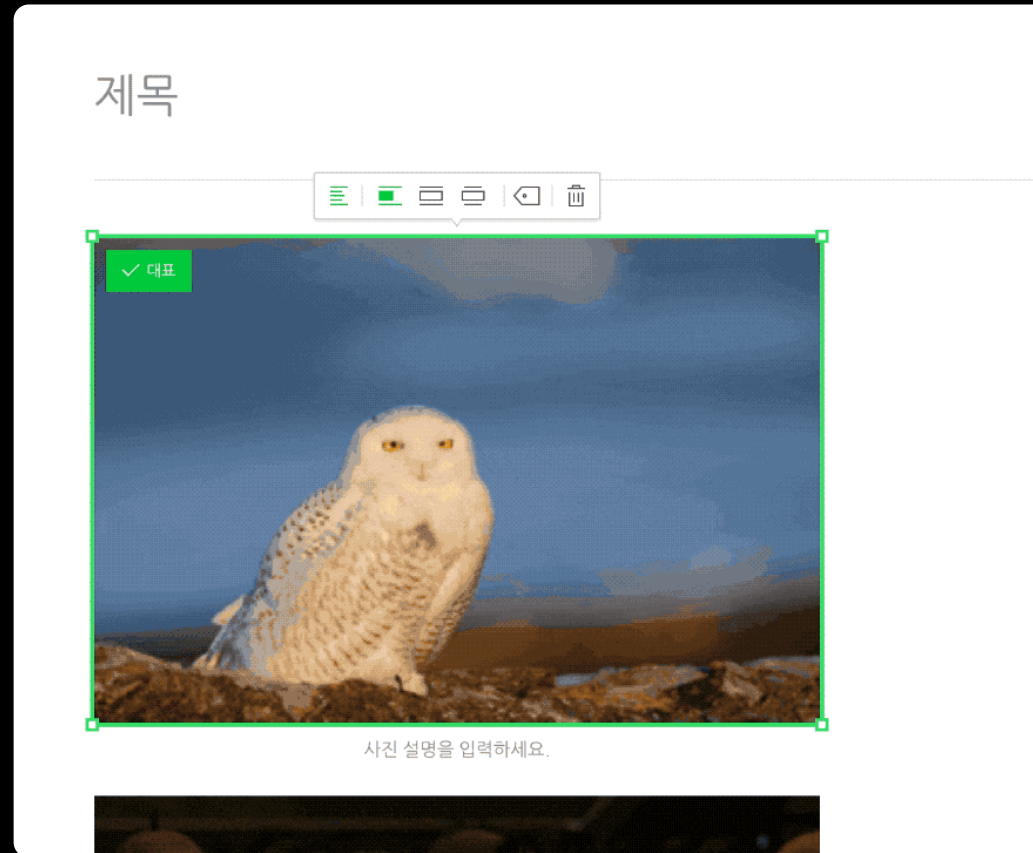


동적 Chunk

글 입력 기존 2000ms > 개선 600ms



컴포넌트 추가 기존 3000s > 개선 1500s



1000 gif Components Rendering
CPU 6x slowdown

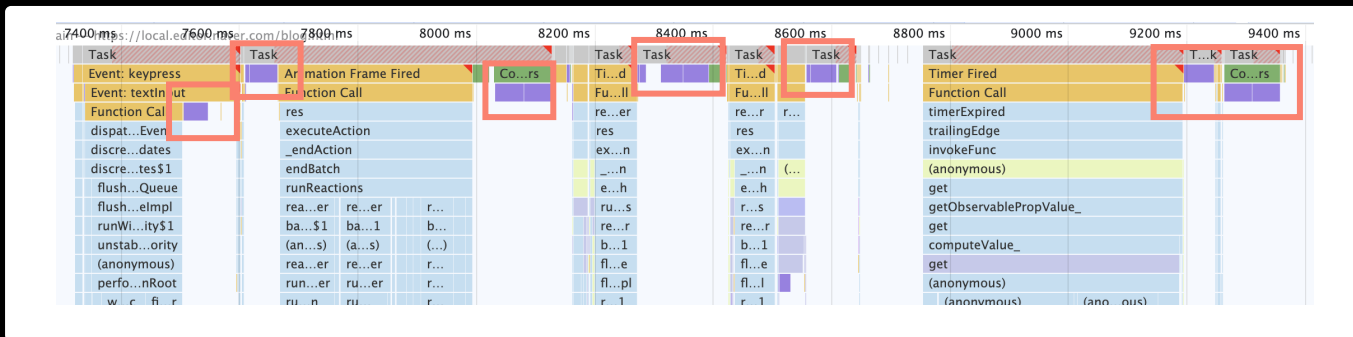
많은 컴포넌트 성능 하락 이슈 해결하기 2

동적 Chunk

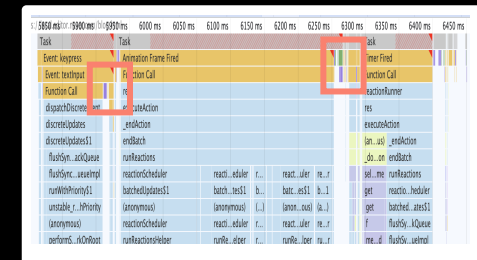
NAVER
DEVVIEW
2023

글 입력

2000ms



600ms



컴포넌트 추가

3000ms

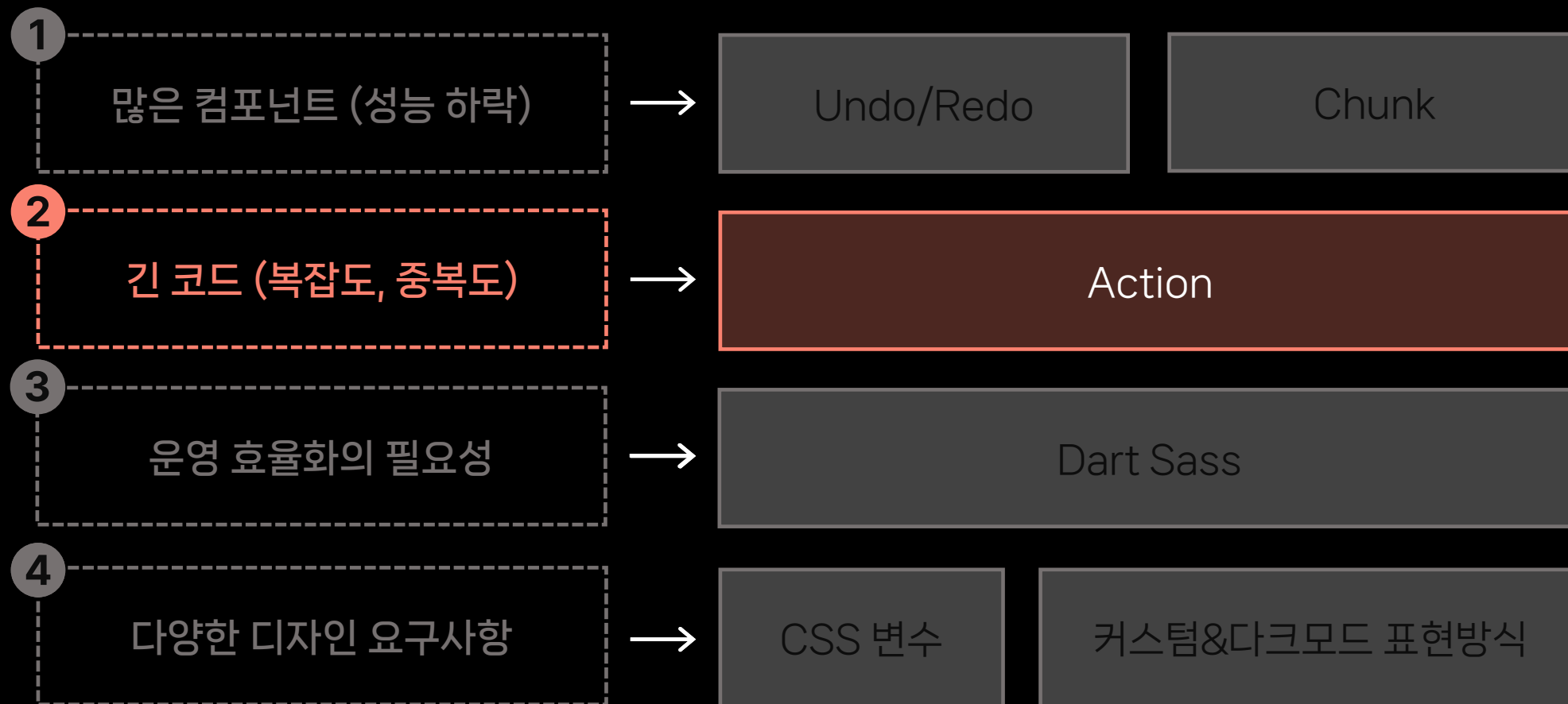


1500ms



1000 gif components Rendering
CPU 6x slowdown

다른 앱에도 적용 가능한 기술



스마트에디터 코드량

전체	코드만	함수	클래스	파일 수
18만줄	14만줄	10,000개	1,200개	1,300개

스마트에디터 코드량

전체	코드만	함수	클래스	파일 수
18만줄	14만줄	10,000개	1,200개	1,300개

복잡도, 중복도 증가

중복도가 높아지는 이유

- **에디터 타입 별로 동작이 일부 다름**

임베드형, 풀페이지형 등 에디터 타입에 따라 일부 동작이 달라지는 문제

- **다양한 횡단 관심 존재**

다양한 횡단 관심(history, selection, log 처리 등)이 존재하는 문제

복잡도가 높아지는 이유 (도메인 로직 제외)

- 동기, 비동기 코드의 혼재

사용자 입력, 네트워크에 따라 동기/비동기가 혼재되는 경우

- 중단 처리

비동기 처리 도중, 사용자가 중단하는 경우

- 실패 처리

특정 동작에 대한 예외처리

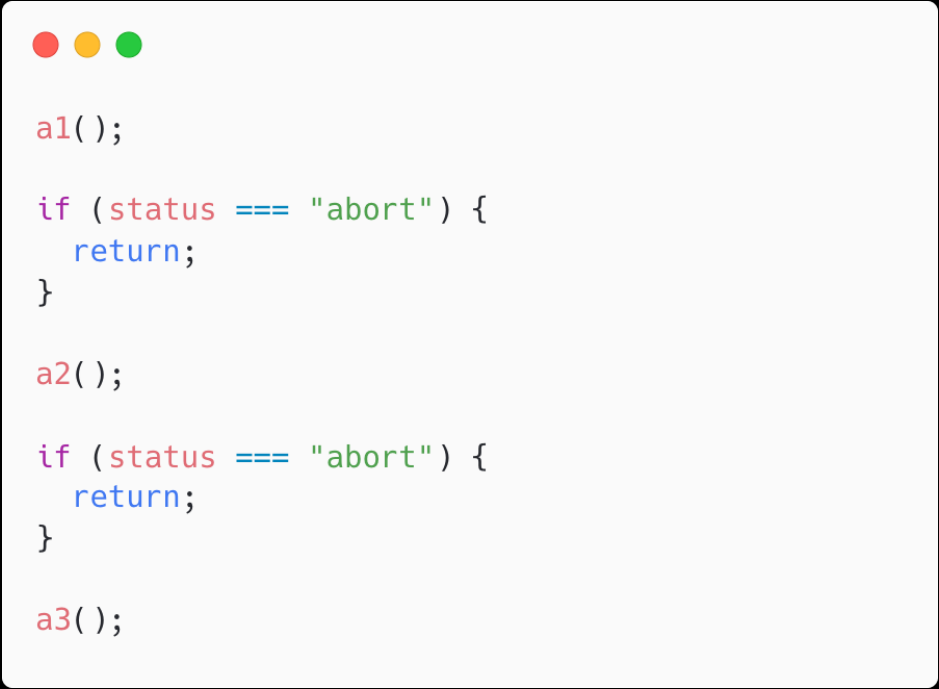
01.

동기, 비동기 코드의 혼재

```
// callback으로 구현시  
a1(() => a2(() => a3));
```

a1이 동기라면 callback을 바로 호출,
a1이 비동기라면 then절 이후 호출

02. 중단 처리



```
a1();  
  
if (status === "abort") {  
  return;  
}  
  
a2();  
  
if (status === "abort") {  
  return;  
}  
  
a3();
```

전역 변수나 return 값에 중단값이 있어야 하며, if문으로 처리 필요

03. 실패 처리

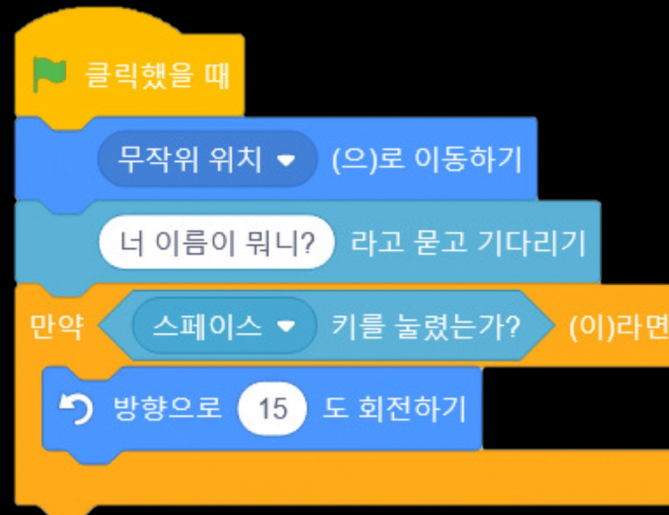
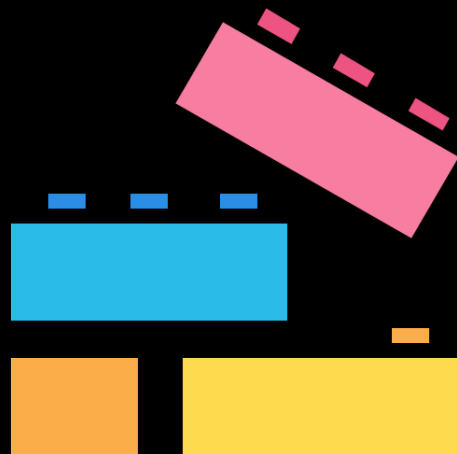
```
try {  
  a1();  
  try {  
    a2();  
  } catch (e) {  
    failA2();  
  }  
} catch (e) {  
  failA1();  
}
```

이전 상태의 성공/실패 여부에 따라 분기하려면
try-catch 문을 중첩 사용해야 함

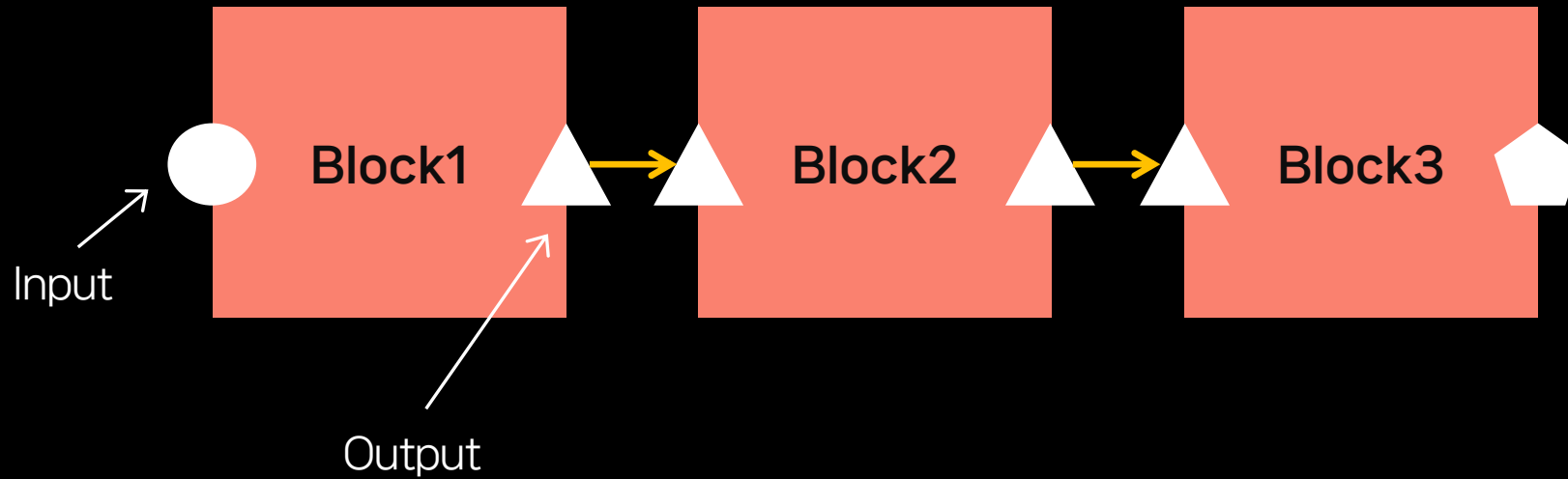
수정 명령을 하나의 블록으로 만들자!

(특허번호 10-2022-0172921)

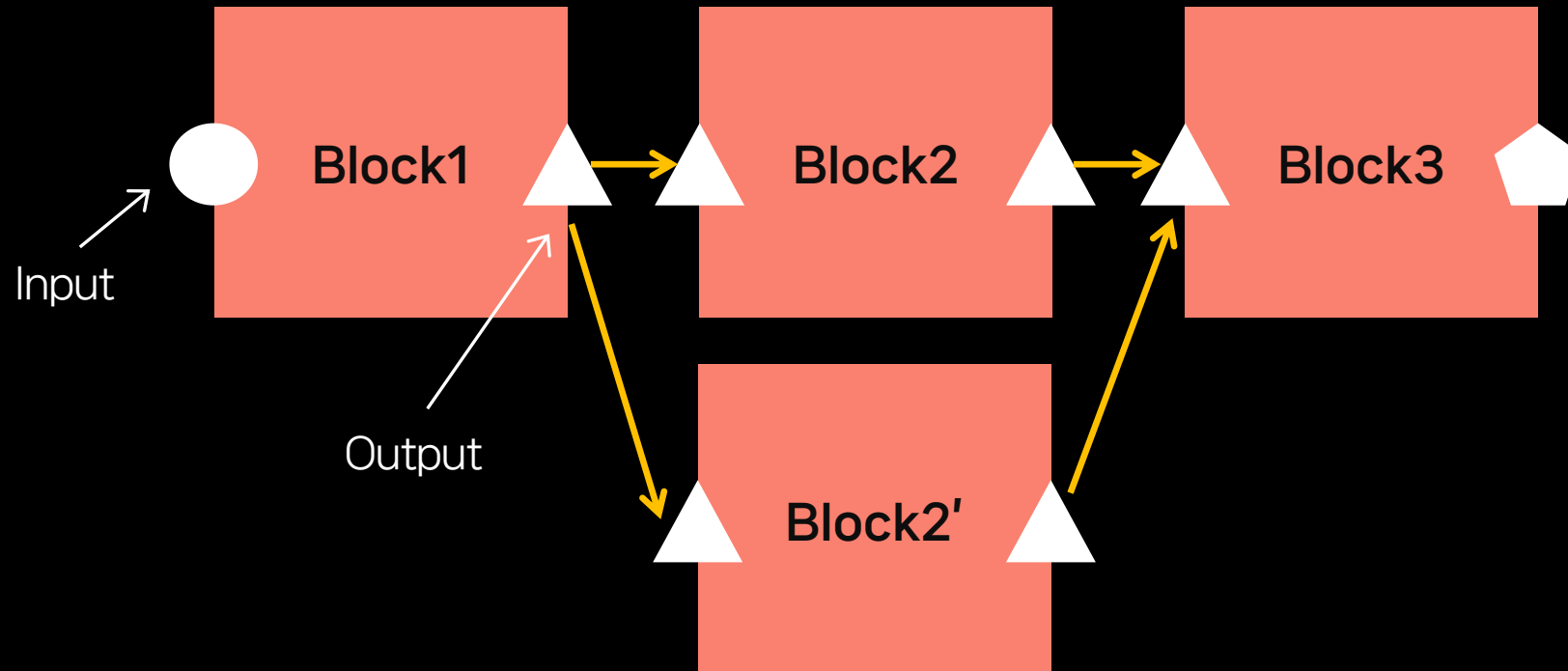
작은 코드블록을 만들고, 이를 조립해 큰 코드블록을 만드는 방법



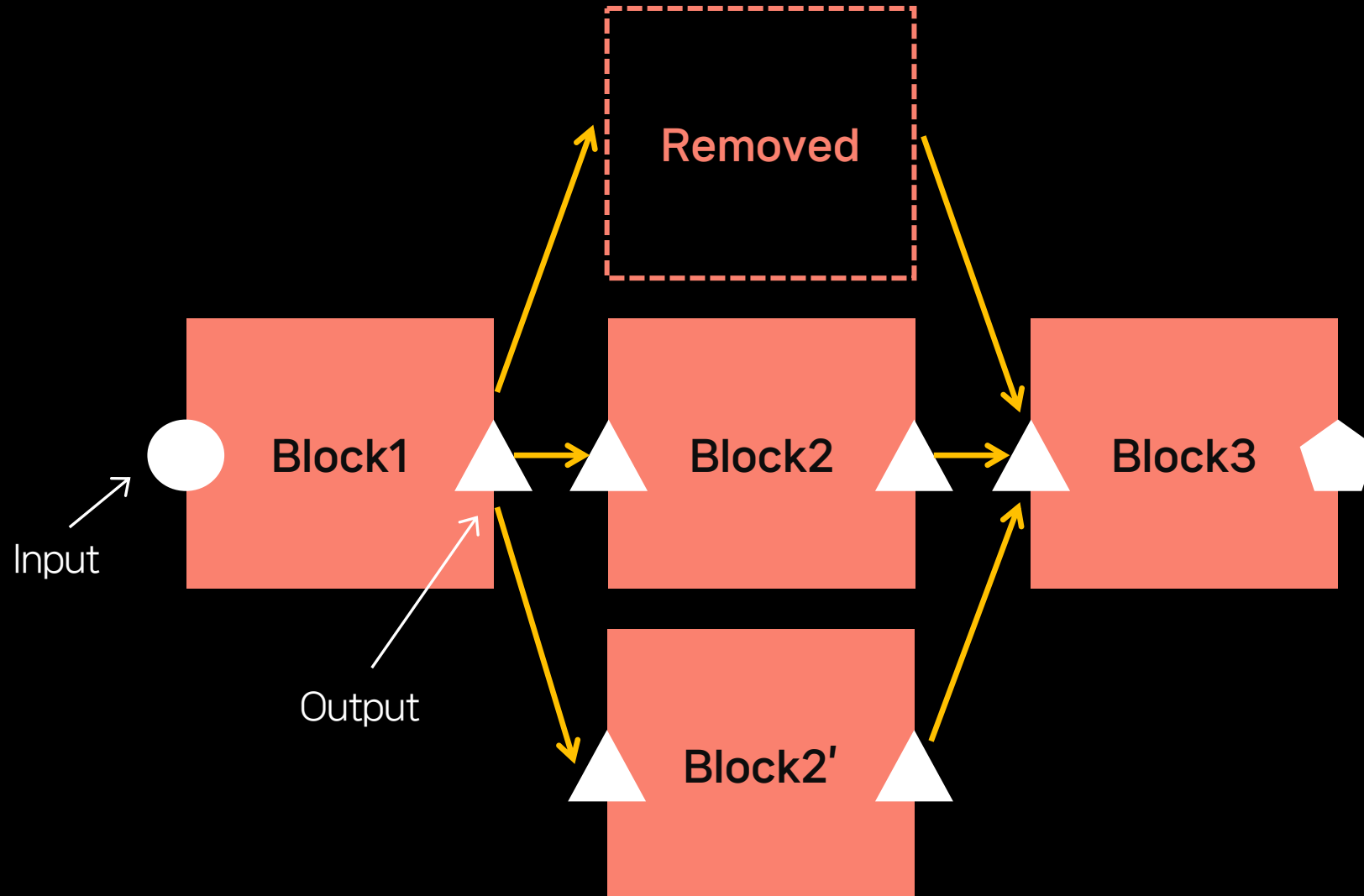
블록코딩



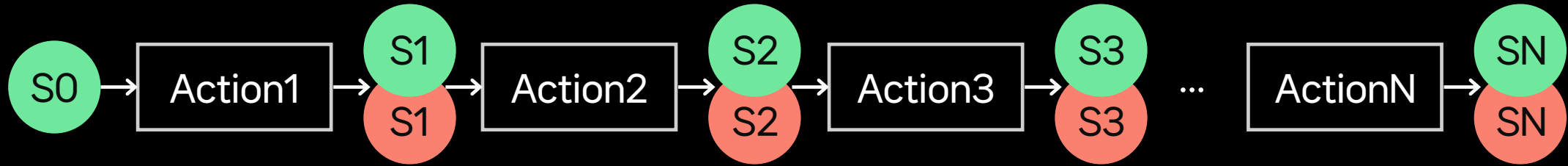
블록코딩



블록코딩



Action



1. 모든 Action은 하나는 선으로 연결되어 있으며, 순차적으로 실행된다.
2. Action 실행 결과는 두 가지 기본 상태(성공, 실패)와 두 가지 특수 상태(Abort, Next)로 존재한다.
3. 이전 상태가 실패라면, 이후 Action은 실행하지 않는다. (Either Action 제외)

Action

```
const abort = Symbol("abort");
const next = Symbol("next");
const fail = Symbol("fail");
const success = Symbol("success");

// Either: [success, value] | [abort | next | fail | success, reason]
class Either extends Array {
  constructor(state, result) {
    super(2);
    this[0] = state;
    this[1] = result;
  }
}

// Result: Either | Promise<Either>
// 동기 | 비동기
// Fn: any => Promise<Either> | Promise<others> | Either | others
// Action: any => Result
// toAction: (fn: Fn) => Action
const toAction = fn => {
  return action = value => {
    return toResult(fn(value));
  };
};
```

특수상태(abort, next)
기본상태(success, fail)

일반함수를 action으로 변경

```
// toResult: (res: any, isFail: boolean) => Result
const toResult = (res, isFail = false) => {
  if (res instanceof Either) {
    return res;
  }
  if (res instanceof Promise) {
    return res.then(
      v => toResult(v),
      err => toResult(err, true),
    );
  }
  return !isFail ? new Either(success, res) : new Either(fail, res);
};
```

동기, 비동기 처리

Action =

Any

→

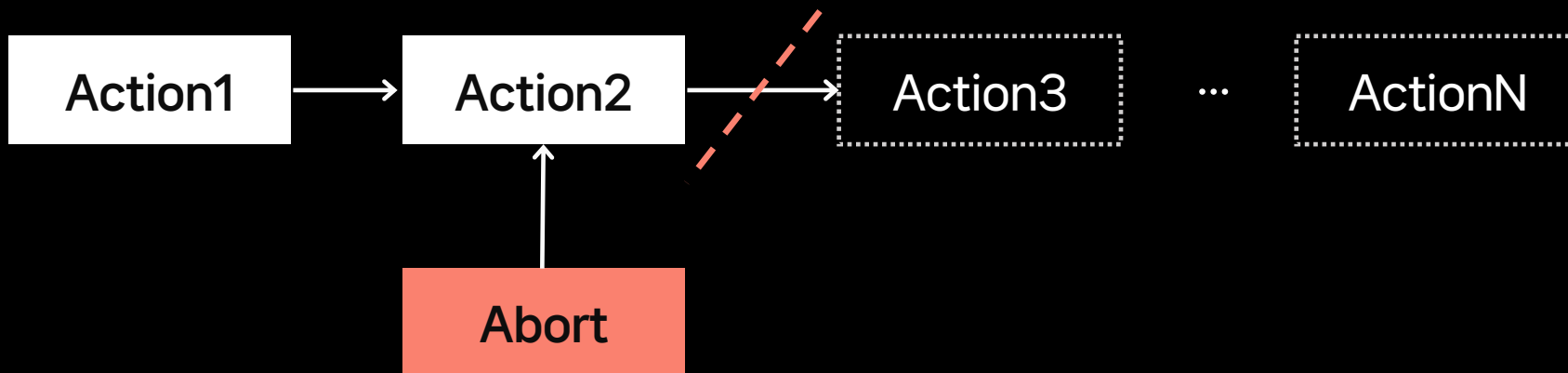
Either | Promise Either

Action

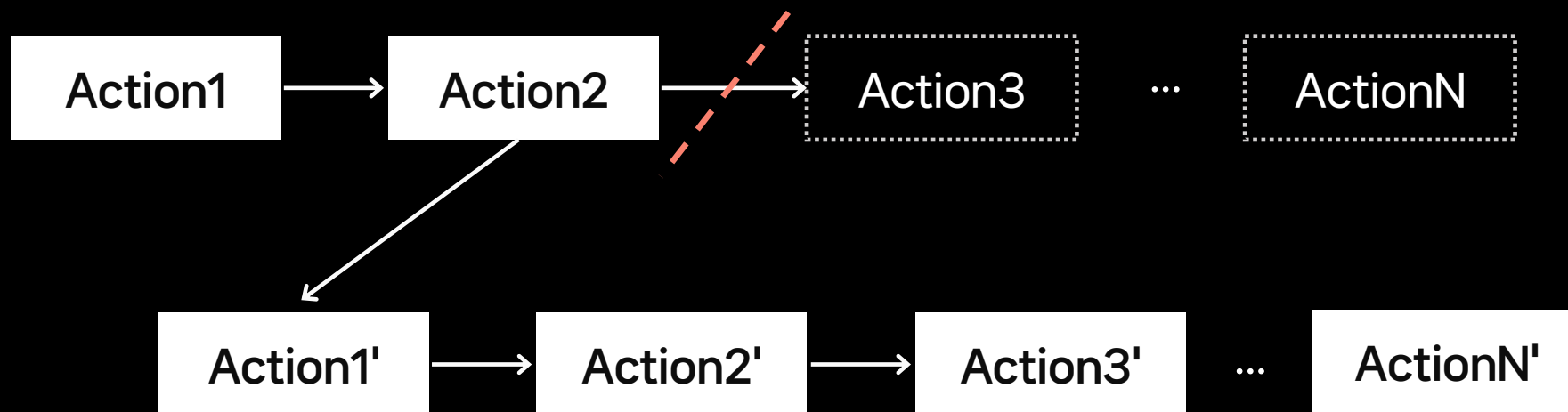


```
const a1 = toAction(() => {  
  return Promise.resolve(Promise.resolve(1));  
});  
  
const a2 = toAction(() => {  
  return Promise.reject("fail");  
});  
  
console.log(a1()); // Promise<Either<success, 1>>  
console.log(a2()); // Promise<Either<fail, "fail">>
```

4. Abort 상태 : 모든 Action은 중단이 가능하며, 중단 이후 Action은 실행하지 않는다.



5. Next 상태 : 모든 Action은 실행 도중 다른 Action을 실행할 수 있다.



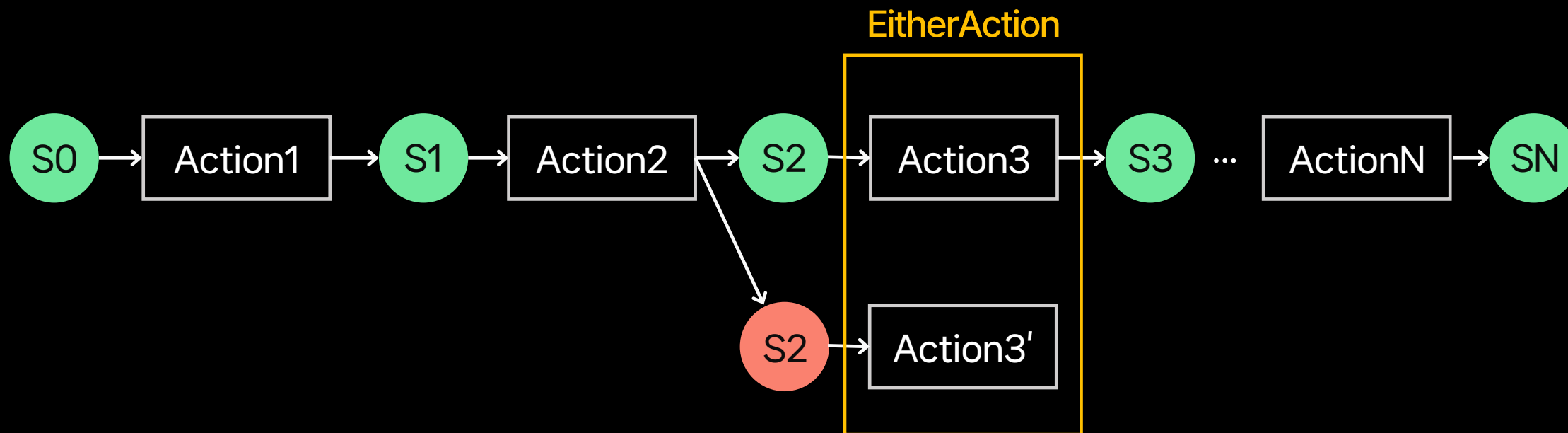
Action helpers

- Either Action : 분기 블록
- Macro Action : 합성 블록
- Decorator : 장식 블록

Action helpers

Either Action

이전 상태에 따라 다른 Action을 실행하는 Action
(성공 시 첫 번째 Action, 실패 시 두 번째 Action을 실행)



Action helpers



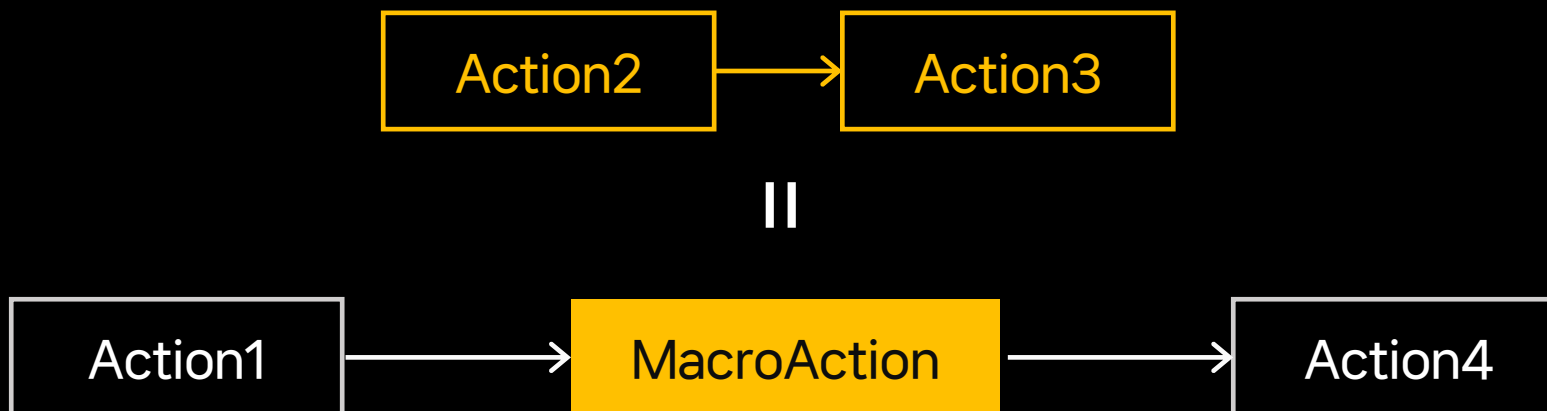
```
const eitherAction = (successAction = identityAction, failAction = identityRejectAction) => {  
  return [successAction, failAction];  
};
```

```
const identity = _ => _;  
const identityAction = toAction(identity);  
const identityRejectAction = toAction((value) => new Either(fail, value));
```

Action helpers

Macro Action

n개 이상의 단일 Action, Either Action, Macro Action



Action helpers

Combine

선행 Action과 후행 Action 합치기

```
// combine: (prevAction: Action, nextActions: [Action, Action]) => Action
const combine = (prevAction, nextActions) => {
  return value => {
    const prevResult = prevAction(value);

    if (prevResult instanceof Promise) {
      return prevResult.then(result => run(result, nextActions));
    }

    return run(prevResult, nextActions);
  };
};
```



Run

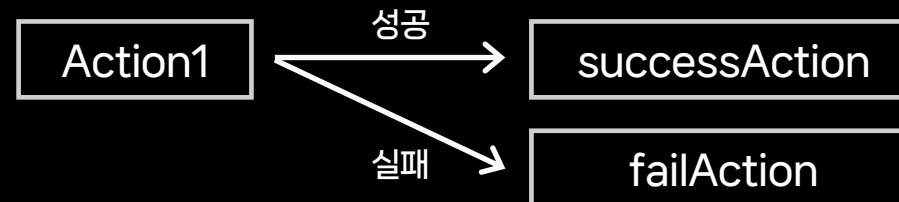
선행 Action 성공 시 success, 실패 시 fail Action 실행

```
// run: (ei: Either, [successAction: Action, failAction: Action]) => Either
const run = (ei, [successAction, failAction]) => {
  const [state, value] = ei;

  if (state === success) {
    return successAction(value);
  }

  if (state === fail) {
    return failAction(value);
  }

  return ei;
};
```



Action helpers

macroAction

combine 함수를 순차 적용하여 하나의 action으로 만드는 함수

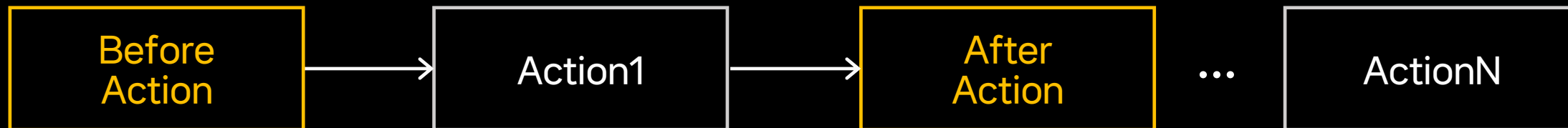


```
const macroAction = (...actions) => {  
  let combinedAction = identityAction;  
  for (const action of actions) {  
    combinedAction = combine(  
      combinedAction,  
      Array.isArray(action) ? action : eitherAction(action)  
    );  
  }  
  
  return combinedAction;  
};
```

Action helpers

Decorator(AOP)

관심사 분리 가능
(종단 Action, 횡단 Action으로 분리해 조합하여 사용 가능)



Action helpers

decorator

기본 액션 앞 뒤에 before, after 액션을 추가하여 반환하는 함수

```
const decorator = (action, beforeAction, afterAction) => {  
  return combine(beforeAction, combine(action, afterAction));  
};
```



Action 적용 전후

```
class Test {  
  @d1  
  async method() {  
    try {  
      let result = await a1();  
  
      if (result === "abort") return;  
  
      result = a2();  
  
      if (result === "abort") return;  
  
      result = a3();  
  
      if (result === "abort") return;  
  
      result = await a4();  
  
      if (result === "abort") return;  
    } catch(e) {  
      if (e === "next") {  
        doNext();  
      }  
    }  
  }  
}
```

Action 적용

```
d1(macroAction(a1, macroAction(a2, a3), a4));
```

- a1, a4: 비동기
- a2, a3: 동기
- d1: decorator

Action 적용 사례

toggleTextStyle

텍스트에 bold를 적용/제거하는 로직 (토글)



```
function toggleTextStyle({styleName, componentListStore, selectionInfoStore, decorator}) {  
  const commonDecorator = decorator.common;  
  
  return commonDecorator(  
    macroAction(  
      1 createAction(() => {  
        const startPoint = selectionInfoStore.start;  
        const endPoint = selectionInfoStore.end;  
        const styles = componentListStore.queryStyle(startPoint, endPoint);  
  
        return {[styleName]: !styles[styleName]};  
      }),  
      2 updateStyleActionFactory(componentListStore, selectionInfoStore),  
    ),  
    ["clearBuffer", "closeUpstream", "history", "selection"],  
    decorator,  
  );  
}
```

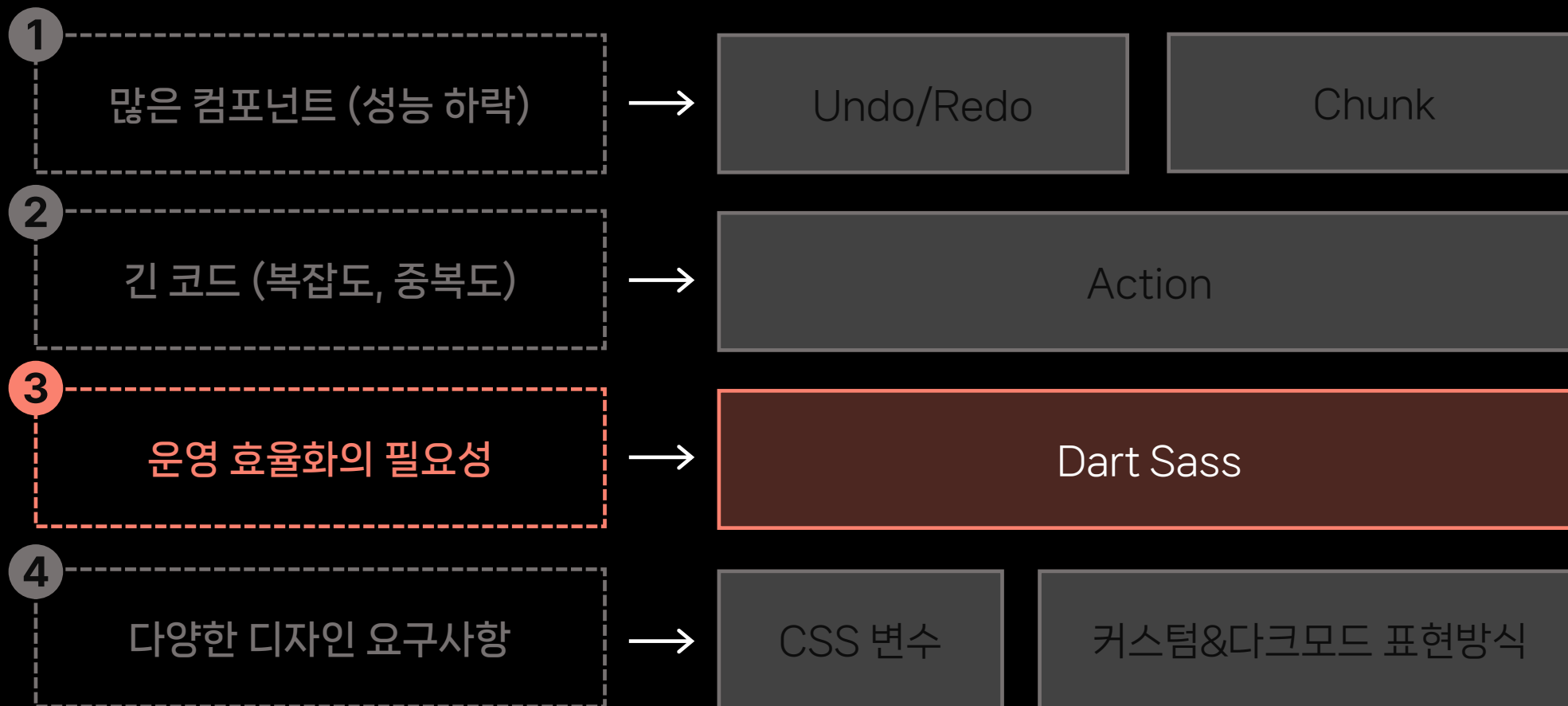
macroAction

decorator

Action을 통한 문제해결

- 중복도 문제
 - 타입 별로 다른 동작> Action 추가, 변경
 - 다양한 횡단 관심> Decorator
- 복잡도 문제
 - 동기, 비동기 처리> Action 규칙1
 - 중단, 실패 처리> 중단 상태, Next 상태

3세대 에디터가 만난 이슈



에디터 스타일 변화

1세대

단일 디자인
(커스텀 X)

2세대

정적인
스타일 커스텀

3세대

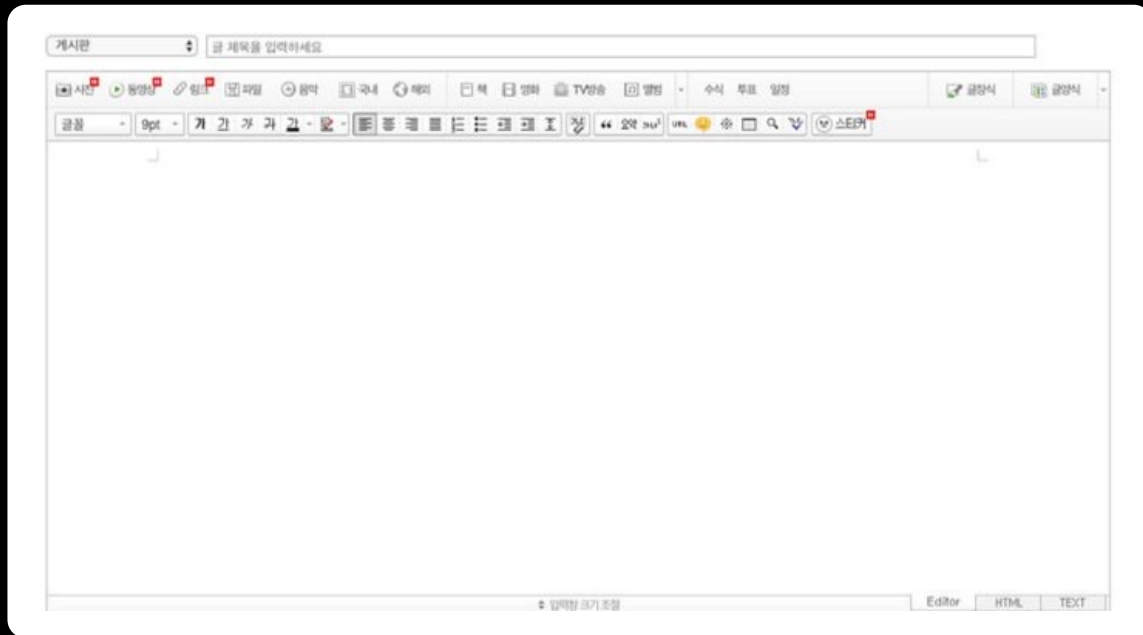
동적인
스타일 커스텀

스마트에디터 스타일 변화

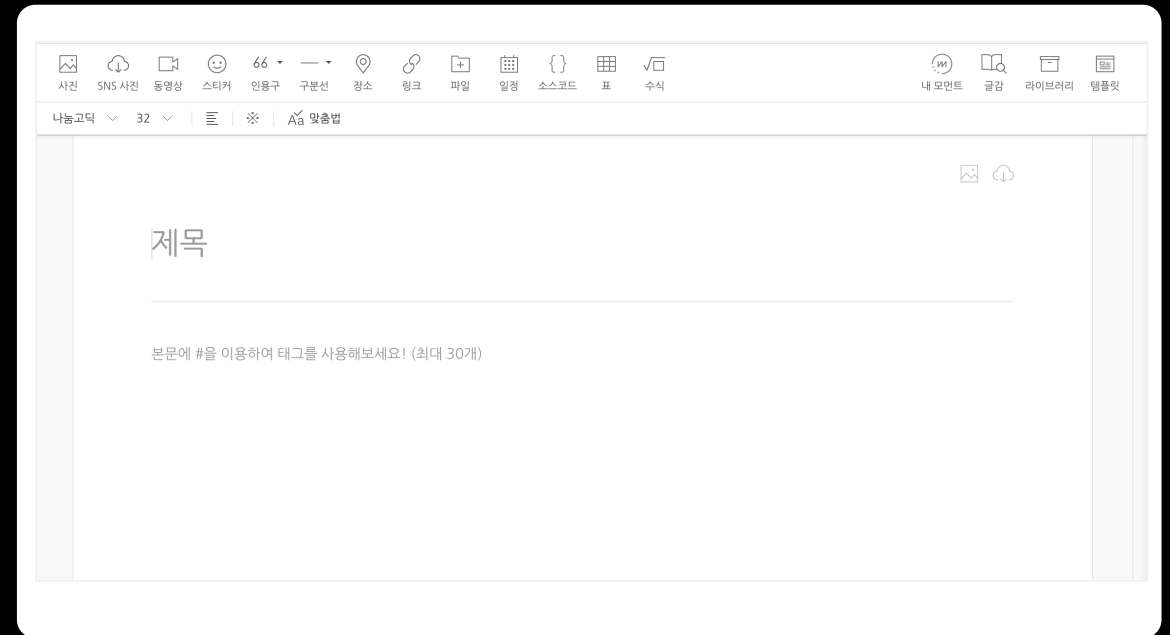
2세대: 스타일 변화 및 설계 변경

NAVER
DEVVIEW
2023

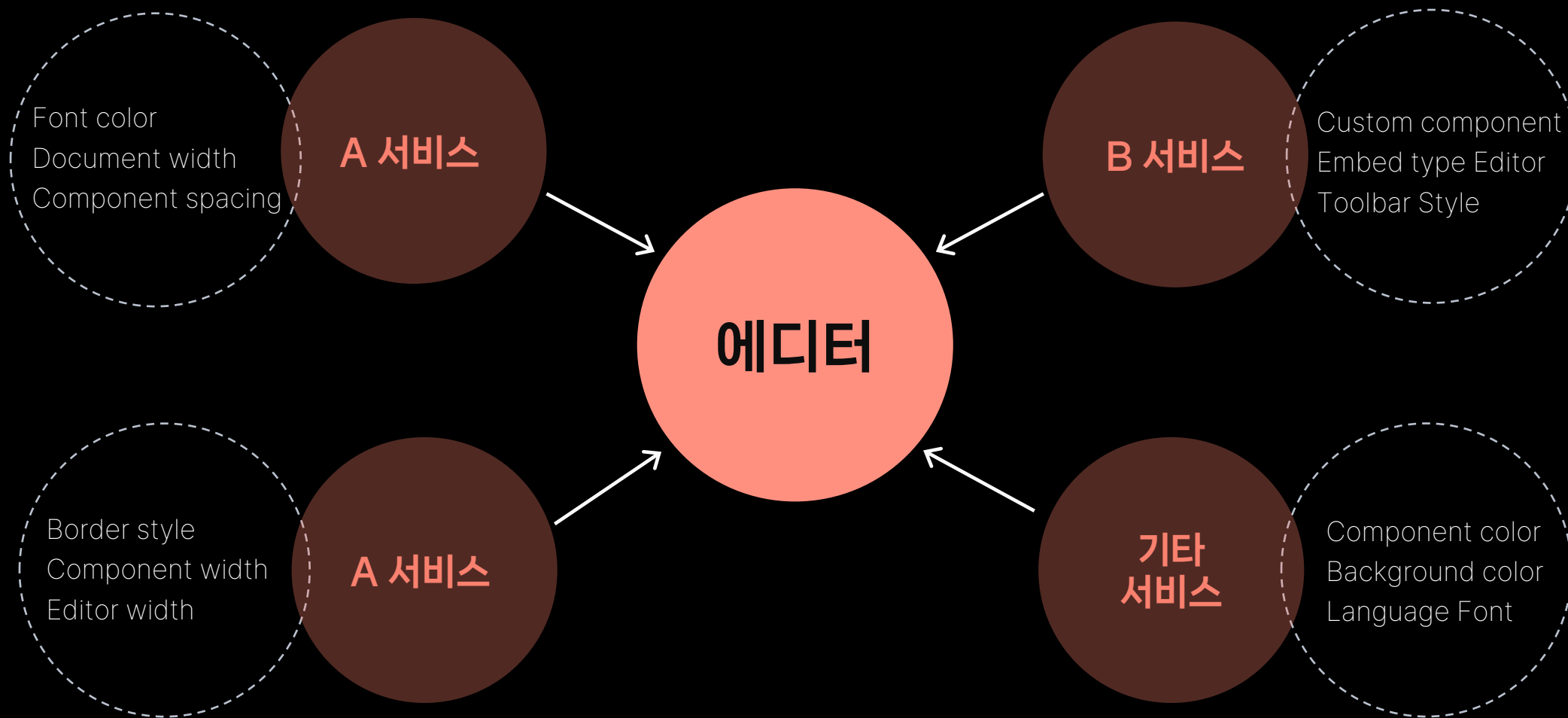
1세대



2세대



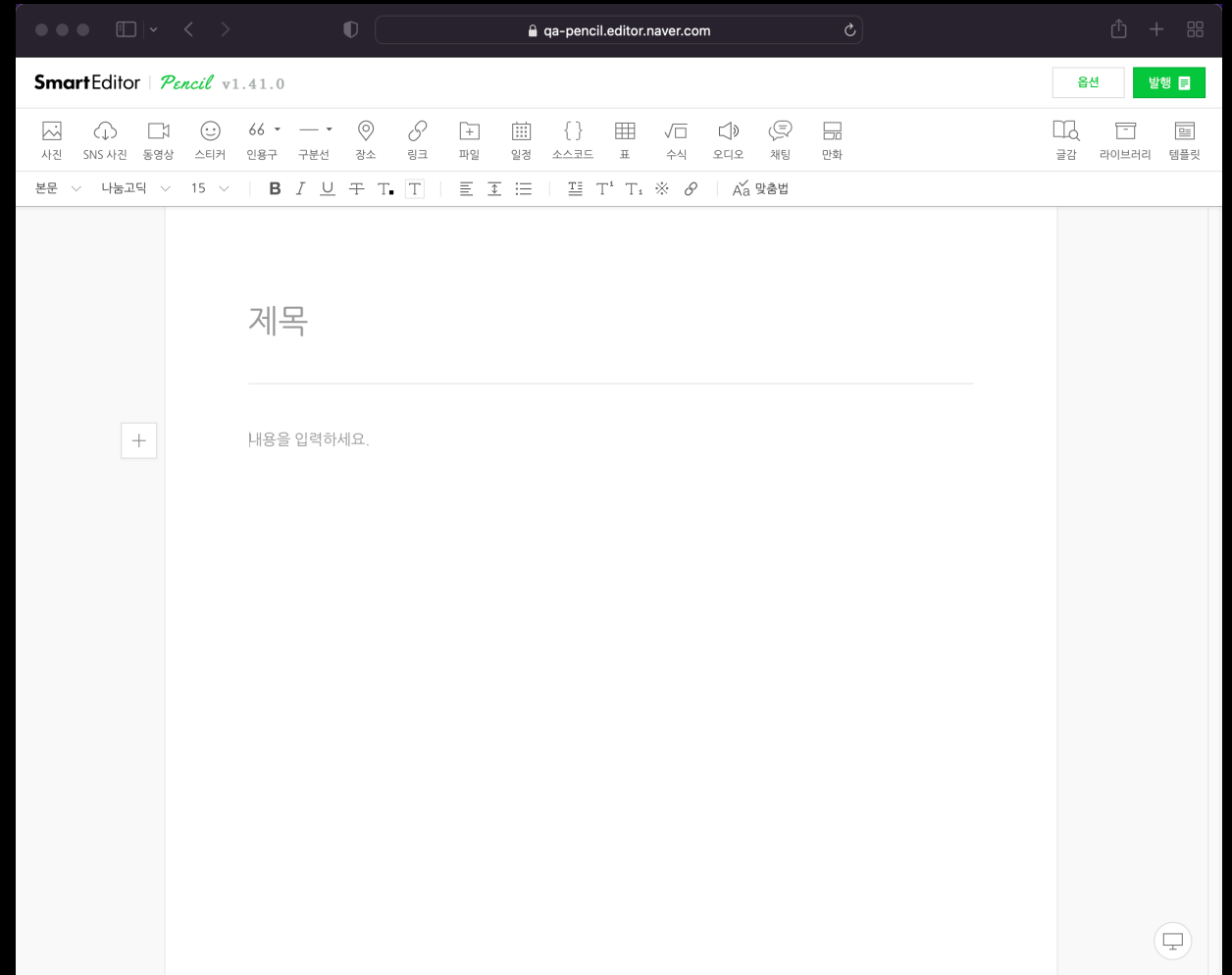
2세대: 다양한 요구사항



스마트에디터 스타일 변화

2세대: 커스텀 방식

간단한 CSS 오버라이딩 방식 가이드의 어려움



2세대: SASS 템플릿 플러그인

문서너비 변경을 위해 필요한 스타일

```
.se-canvas-bottom-button {  
  width: 700px;  
}  
.se-module-formula .se-formula-viewer-wrap,  
.se-component-edge-button,  
.se-drop-indicator-bottom:after,  
.se-component-content > .se-drop-indicator:after,  
.se-component-content,  
.se-component-content-fit,  
.se-cover-button-wrap,  
.se-video-info-wrap,  
.se-caption,  
.se-module-code .se-code-source-editor {  
  max-width: 700px;  
}
```

자동화



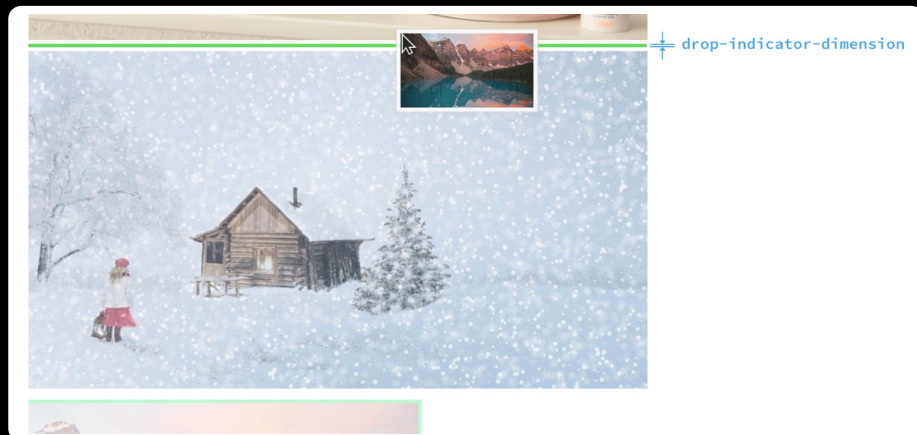
설정 기반 Sass 템플릿

```
$config: (  
  wrapper-class: ".desktop-editor",  
  document-width-config: 700px  
);
```

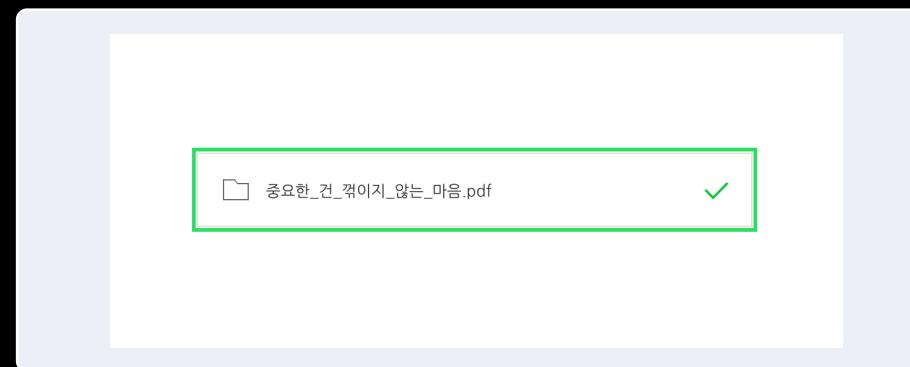
스마트에디터 스타일 변화

2세대: SASS 템플릿 플러그인

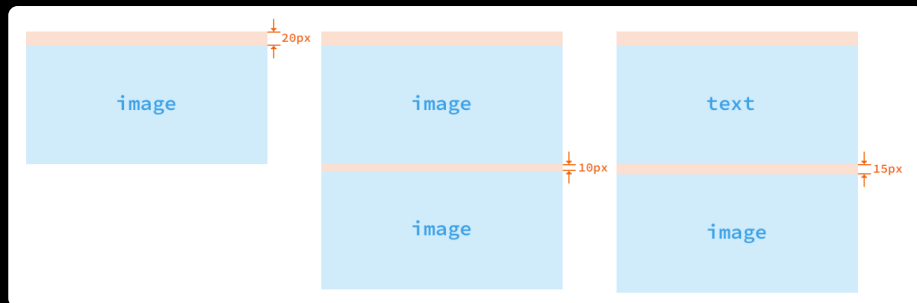
컴포넌트 이동 시 노출되는 가이드라인 높이



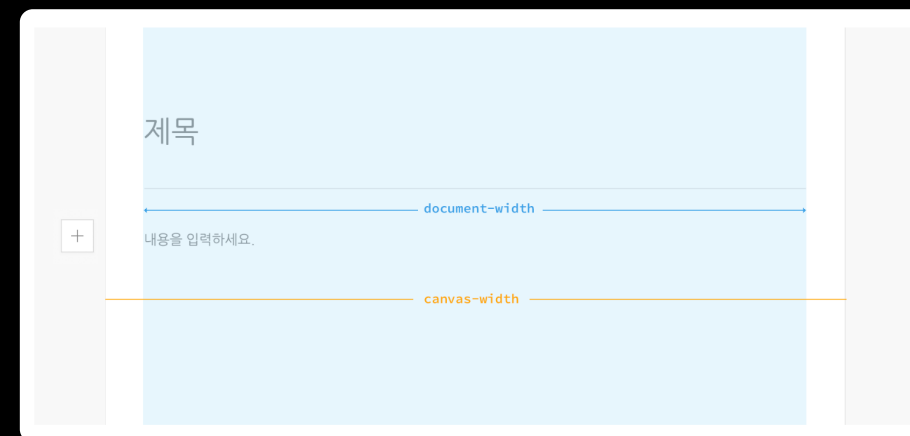
컴포넌트 활성화 스타일



컴포넌트 사이 간격



문서 너비



스마트에디터 스타일 변화

2세대: SASS 템플릿 플러그인의 한계

NAVER
DEVVIEW
2023

많은 서비스

기민한 대응

많은 요구사항

신규 기능 추가

커뮤니케이션
비용 증가

신규 기능 추가에 따른
플러그인 수정 작업

리소스
과부하

3세대: 변화에 대응 하는 에디터

IE 스펙 제외

CSS 변수

&

Node SASS

Dart SASS

스마트에디터 스타일 변화

이전 설계 구조

NAVER
DEVIEW
2023

```
// variables.scss
$primary-color: #00c73c;
$text-placeholder-color: #999;
$selection-background: rgba(110, 192, 255, .4);

$primary-font-family: se-nanumgothic, nanumgothic, sans-serif;
$secondary-font-family: se-nanumsquare, nanumsquare, sans-serif;

$header-document-toolbar-button-height: 54px;
$header-document-toolbar-button-width: 40px;

$sidebar-panel-header-height: 60px;
$sidebar-panel-search-tab-height: 37px;
$sidebar-panel-search-sorting-height: 41px;

$canvas-width-mobile: 400px;
$canvas-width-tablet: 768px;
$canvas-width-desktop: 860px;

$easeInOutCubic: cubic-bezier(0.645, 0.045, 0.355, 1);
$easeOutExpo: cubic-bezier(.19, 1, .22, 1);
$easeOutCubic: cubic-bezier(0.215, 0.610, 0.355, 1.000);
$springCubic: cubic-bezier(.38,1.71,.31,1);
$easeOutBack: cubic-bezier(0.175, 0.885, 0.32, 1.275);

...
```

```
// mixin.scss
@mixin blind() { ... }

@mixin clearfix() { ... }

@mixin placeholder() { ... }

@mixin placeholder-caret-style () { ... }

@mixin virtual-placeholder-style() { ... }

@mixin dragging-ghost-revision() { ... }

@mixin dragging-canvas-revision() { ... }

@mixin sidebar-ghost() { ... }

@mixin ui-interaction() { ... }

...
```

```
// function.scss
@function get-map-value() { ... }

@function str-to-list() { ... }

@function selectors-generator() { ... }

@function get-component-value() { ... }

@function get-component-base-spacing() { ... }

@function get-component-adjacent-spacing() { ... }

@function get-font-size() { ... }

...
```

스마트에디터 스타일 변화

이전 설계 구조

NAVER
DEVVIEW
2023

```
// variables.scss
$primary-color: #00c73c;
$text-placeholder-color: #999;
$selection-background-color: #f0f0f0;

$primary-font-family: se-nanumsquare, nanumsquare, sans-serif;
$secondary-font-family: se-nanumsquare, nanumsquare, sans-serif;

$header-document-toolbar-button-height: 54px;
$header-document-toolbar-button-width: 40px;

$sidebar-panel-header-height: 60px;
$sidebar-panel-search-tab-height: 37px;
$sidebar-panel-search-sorting-height: 41px;

$canvas-width-mobile: 400px;
$canvas-width-tablet: 768px;
$canvas-width-desktop: 860px;

$easeInOutCubic: cubic-bezier(0.645, 0.045, 0.355, 1);
$easeOutExpo: cubic-bezier(.19, 1, .22, 1);
$easeOutCubic: cubic-bezier(0.215, 0.610, 0.355, 1.000);
$springCubic: cubic-bezier(.38,1.71,.31,1);
$easeOutBack: cubic-bezier(0.175, 0.885, 0.32, 1.275);

...
```

변수 약 140개

```
// mixin.scss
@mixin blind() { ... }

@mixin clearfix() { ... }

@mixin placeholder() { ... }

@mixin placeholder-caret-style () { ... }

@mixin virtual-placeholder-style() { ... }

@mixin dragging-ghost-revision() { ... }

@mixin dragging-canvas-revision() { ... }

@mixin sidebar-ghost() { ... }

@mixin ui-interaction() { ... }

...
```

mixin 약 50개

```
// function.scss
@function get-map-value() { ... }

@function string-replace() { ... }

@function string-replace-all() { ... }

@function get-component-value() { ... }

@function get-component-base-spacing() { ... }

@function get-component-adjacent-spacing() { ... }

@function get-font-size() { ... }

...
```

function 약 20개

이전 설계 구조의 개선 방향

문제점

중복 선언의 위험성

길어지는 네이밍

관리 효율성 부족



개선 방향

네이밍 간소화

명확한 참조관계 표현

높은 재사용성

관리 효율성 증가

»» 모듈화

스마트에디터 스타일 변화

모듈화 방식 적용

NAVER
DEVIEW
2023

```
// variables.scss
$primary-color: #00c73c;
$text-placeholder-color: #999;
$selection-background: rgba(110, 192, 255, .4);

$primary-font-family: se-nanumgothic, nanumgothic, sans-serif;
$secondary-font-family: se-nanumsquare, nanumsquare, sans-serif;

$header-document-toolbar-button-height: 54px;
$header-document-toolbar-button-width: 40px;

$sidebar-panel-header-height: 60px;
$sidebar-panel-search-tab-height: 37px;
$sidebar-panel-search-sorting-height: 41px;

$canvas-width-mobile: 400px;
$canvas-width-tablet: 768px;
$canvas-width-desktop: 860px;

$easeInOutCubic: cubic-bezier(0.645, 0.045, 0.355, 1);
$easeOutExpo: cubic-bezier(.19, 1, .22, 1);
$easeOutCubic: cubic-bezier(0.215, 0.610, 0.355, 1.000);
$springCubic: cubic-bezier(.38,1.71,.31,1);
$easeOutBack: cubic-bezier(0.175, 0.885, 0.32, 1.275);

...
```

```
// mixin.scss
@mixin blind() { ... }

@mixin clearfix() { ... }

@mixin placeholder() { ... }

@mixin placeholder-caret-style() { ... }
@mixin variable-placeholder-style() { ... }
@mixin dragging-ghost-revision() { ... }

@mixin dragging-canvas-revision() { ... }

@mixin sidebar-ghost() { ... }

@mixin ui-interaction() { ... }

...
```

관심사 분리

```
// function.scss
@function get-map-value() { ... }

@function str-to-list() { ... }

@function selectors-generator() { ... }

@function get-component-value() { ... }

@function get-component-base-spacing() { ... }

@function get-component-adjacent-spacing() { ... }

@function get-font-size() { ... }

...
```

관심사 분리 작업

variables.scss

color,
font,
sidebar

...

mixin.scss

color,
font,
sidebar

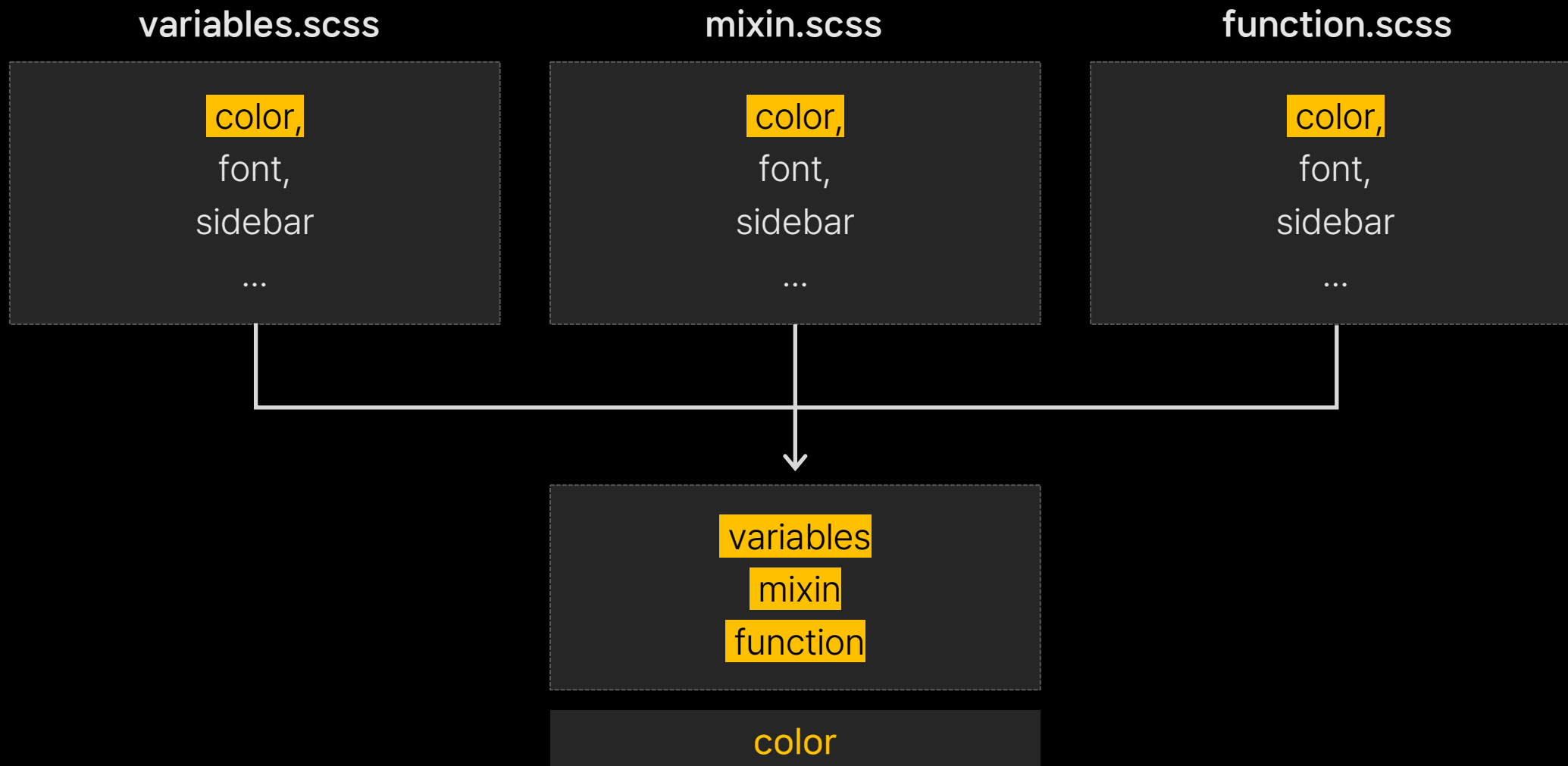
...

function.scss

color,
font,
sidebar

...

관심사 분리 작업



설계 구조 개선1. 관심사 분리

```
// color
$primary-color: #00c73c;
$text-placeholder-color: #999;
$selection-background: rgba(110, 192, 255, .4);

// font
$primary-font-family: se-nanumgothic, nanumgothic, sans-serif;
$secondary-font-family: se-nanumsquare, nanumsquare, sans-serif;

// header
$header-document-toolbar-button-height: 54px;
$header-document-toolbar-button-width: 40px;

// sidebar
$sidebar-panel-header-height: 60px;
$sidebar-panel-search-tab-height: 37px;
$sidebar-panel-search-sorting-height: 41px;

// canvas
$canvas-width-mobile: 400px;
$canvas-width-tablet: 768px;
```



color 모듈

```
// color
$primary: #00c73c;
$text-placeholder: #999;
$selection-background: rgba(110, 192, 255, .4);
```

설계 구조 개선1. 관심사 분리



```
// color
$primary-color: #00c73c;
$text-placeholder-color: #999;
$selection-background: rgba(110, 192, 255, .4);
```

```
// font
$primary-font-family: se-nanumgothic, nanumgothic, sans-serif;
$secondary-font-family: se-nanumsquare, nanumsquare, sans-serif;
```

```
// header
$header-document-toolbar-button-height: 54px;
$header-document-toolbar-button-width: 40px;
```

```
// sidebar
$sidebar-panel-header-height: 60px;
$sidebar-panel-search-tab-height: 37px;
$sidebar-panel-search-sorting-height: 41px;
```

```
// canvas
$canvas-width-mobile: 400px;
$canvas-width-tablet: 768px;
```



color 모듈

```
// color
$primary: #00c73c;
$text-placeholder: #999;
$selection-background: rgba(110, 192, 255, .4);
```



font 모듈

```
// font
$primary: se-nanumgothic, nanumgothic, sans-serif;
$secondary: se-nanumsquare, nanumsquare, sans-serif;
```

스마트에디터 스타일 변화

설계 구조 개선2. 모듈화

header 모듈

```
// header/index.scss
```

```
@use "font";  
@use "color";
```

```
.header {  
  font-family: font.$primary;  
  color: color.$primary;  
  
  button {  
    font-family: font.$secondary;  
  }  
}
```

color 모듈

```
// color
```

```
$primary: #00c73c;  
$text-placeholder: #999;  
$selection-background: rgba(110, 192, 255, .4);
```

font 모듈

```
// font
```

```
$primary: se-nanumgothic, nanumgothic, sans-serif;  
$secondary: se-nanumsquare, nanumsquare, sans-serif;
```

스마트에디터 스타일 변화

설계 구조 개선3. Private Members

header 모듈

```
// header/index.scss
@use "font";
@use "color";

$-toolbar-width: 40px;
$-toolbar-height: 54px;

.header {
  font-family: font.$primary;
  color: color.$primary;

  button {
    width: $-toolbar-width;
    height: $-toolbar-height;
    font-family: font.$secondary;
  }
}
```

스마트에디터 스타일 변화

설계 구조 개선

● ● ●

oglink 모듈

```
// component/oglink/index.scss
@use "component-generator" as generator;
@use "ghost";
@use "utils";
@use "selection";
@use "sprite";

@include generator.component(oglink);
@include ghost.component(default, oglink, null, 450, 126);

@mixin -oglink-init() { ... }

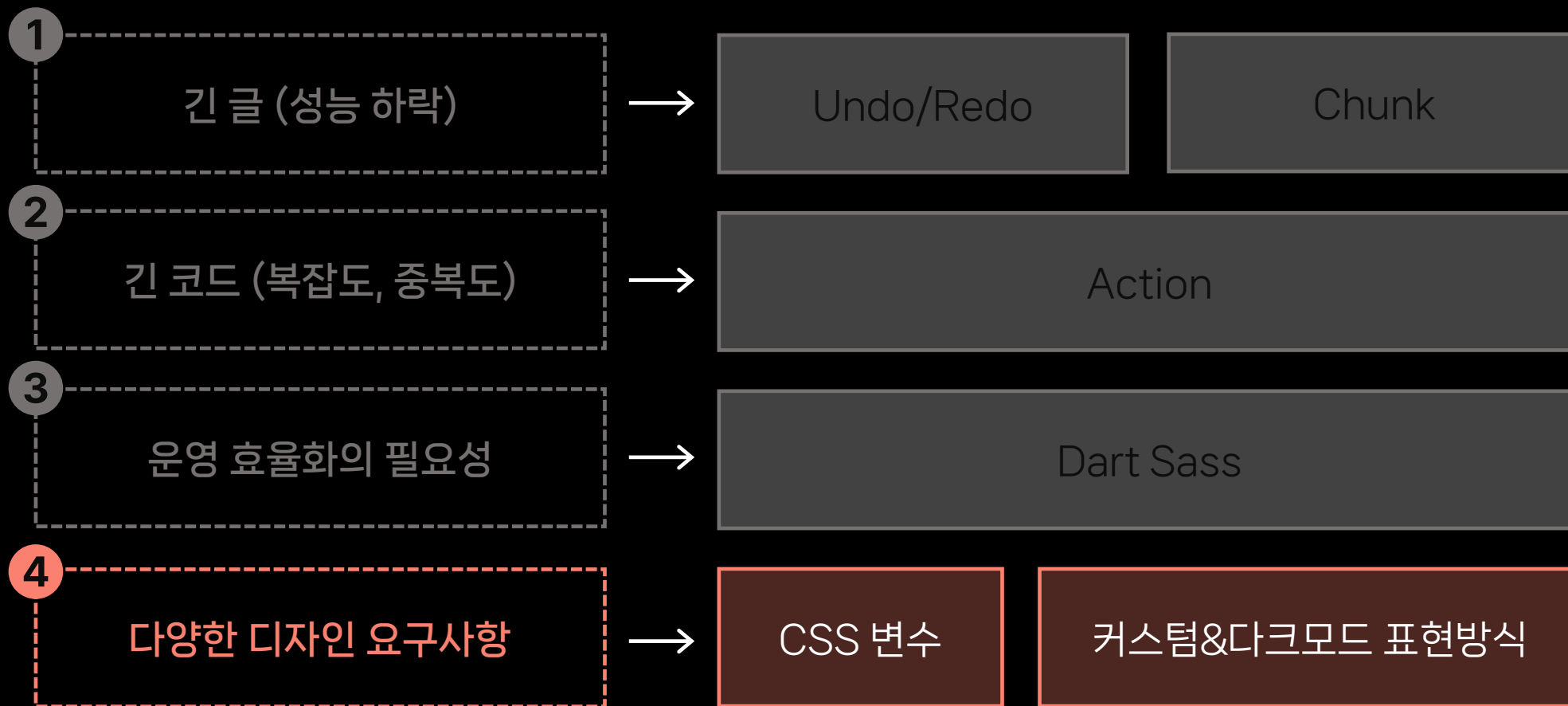
.se-oglink {
  @include -oglink-init();
}

.se-section-oglink {
  width: 100%;
  max-width: 450px;

  @include selection.selection();
}
```

- ✓ 네이밍 간소화
- ✓ 명확한 참조관계 표현
- ✓ 높은 재사용성
- ✓ 관리 효율성 증가

3세대 에디터가 만난 이슈



커스텀 방식 개선

문제점

코드 이원화

리소스 과부하

관리 효율성 부족



개선 방향

코드 단일화

쉬운 적용 방법

관리 효율성 증가

»»» CSS 변수

CSS 변수를 이용한 커스텀 방법

CSS 변수 사용법

```
:root {  
  --se-default-component-text-color: rgba(#000, 0.9);  
  --se-default-component-background-color: #fff;  
}  
  
.component {  
  color: var(--se-component-text-color, var(--se-default-component-text-color));  
  background: var(--se-component-background-color, var(--se-default-component-background-color));  
}
```


CSS 변수를 이용한 커스텀 방법

스마트에디터에서 사용 시

```
:root {  
  --se-default-component-text-color: rgba(#000, 0.9);  
  --se-default-component-background-color: #fff;  
}  
  
.component {  
  color: var(--se-component-text-color, var(--se-default-component-text-color));  
  background: var(--se-component-background-color, var(--se-default-component-background-color));  
}
```



서비스에서 Override 사용 시

```
// service.css  
:root {  
  --se-component-text-color: #000;  
  --se-component-background-color: #fff;  
}
```



CSS 변수를 이용한 커스텀 방법

스마트에디터에서 사용 시

```

:root {
  --se-component-text-color: rgba(#000, 0.9);
  --se-component-background-color: #fff;
}

.component {
  color: var(--se-component-text-color);
  background: var(--se-component-background-color);
}
```

서비스에서 Override 사용 시

```

// service.css
:root {
  --se-component-text-color: #000;
  --se-component-background-color: #fff;
}
```

CSS 변수를 이용한 커스텀 방법

스마트에디터에서 사용 시

```

:root {
  --se-component-text-color: rgba(#000, 0.9);
  --se-component-background-color: #fff;
}

.component {
  color: var(--se-component-text-color);
  background: var(--se-component-background-color);
}
```

```

// color/index.scss
$v-white: #fff;

$v-gray-900: rgba(#000, 0.9);
$v-gray-800: rgba(#000, 0.8);
$v-gray-700: rgba(#000, 0.7);

...
```

color 모듈

CSS 변수를 이용한 커스텀 방법

스마트에디터에서 사용 시

```

@use "color";

:root {
  --se-component-text-color: #{color.$v-gray-900};
  --se-component-background-color: #{color.$v-white};

.component {
  color: var(--se-component-text-color);
  background: var(--se-component-background-color);
}

```

```

// color/index.scss
$v-white: #fff;

$v-gray-900: rgba(#000, 0.9);
$v-gray-800: rgba(#000, 0.8);
$v-gray-700: rgba(#000, 0.7);

...

```

color 모듈

다양한 디자인 요구사항 대응

CSS 변수를 이용한 커스텀: 다크모드

NAVER
DEVVIEW
2023

color 모듈

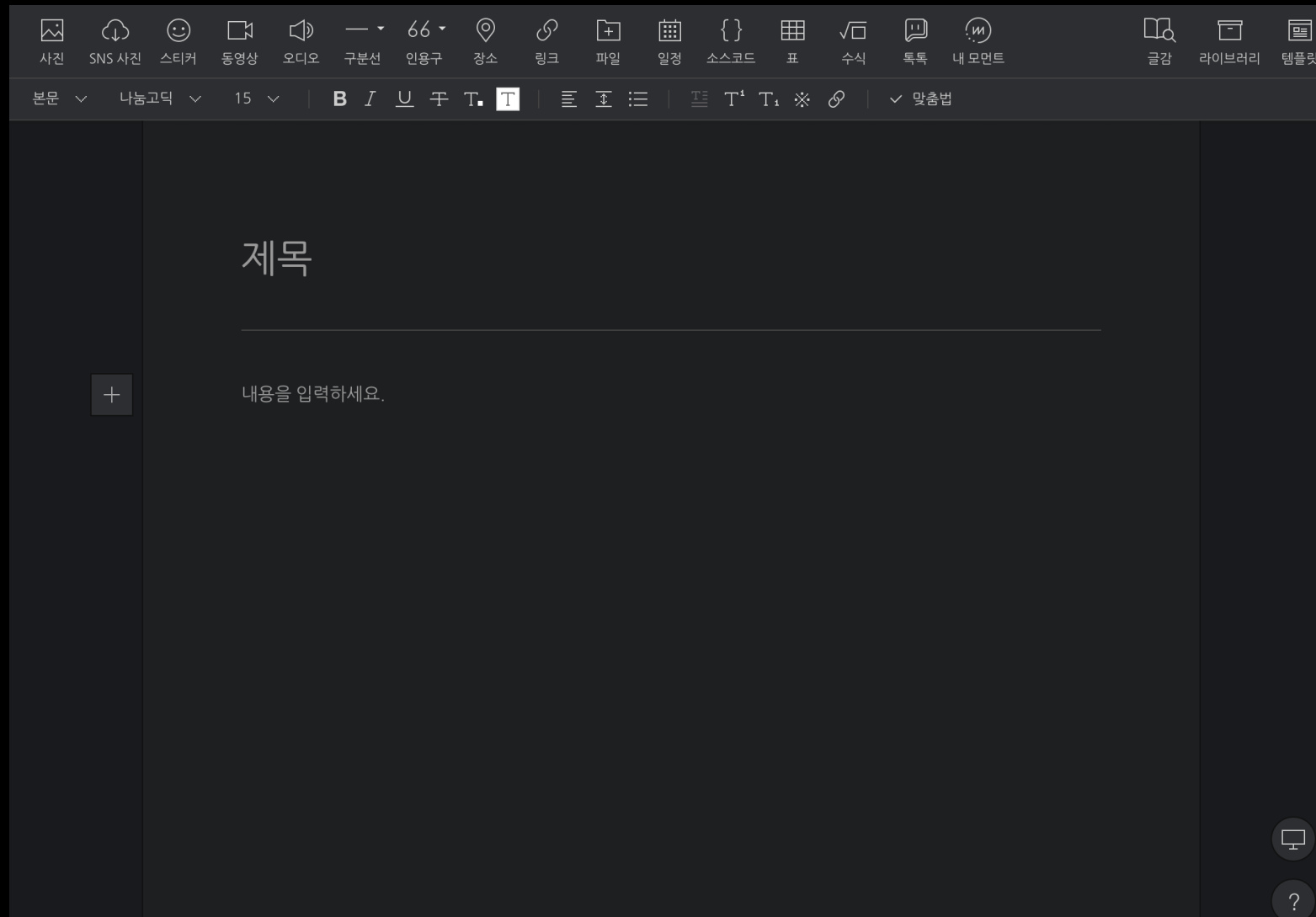
```
// color/index.scss
@use "media";

$v-white: #fff;

$v-gary-900: rgba(#000, 0.9);
$v-gary-800: rgba(#000, 0.8);
$v-gary-700: rgba(#000, 0.7);

@include media.dark-mode() {
  $v-white: #000;

  $v-gary-900: rgba(#fff, 0.9);
  $v-gary-800: rgba(#fff, 0.8);
  $v-gary-700: rgba(#fff, 0.7);
  ...
}
```



다양한 디자인 요구사항 대응

CSS 변수 활용 방법

1

폰트 서체 스타일

2

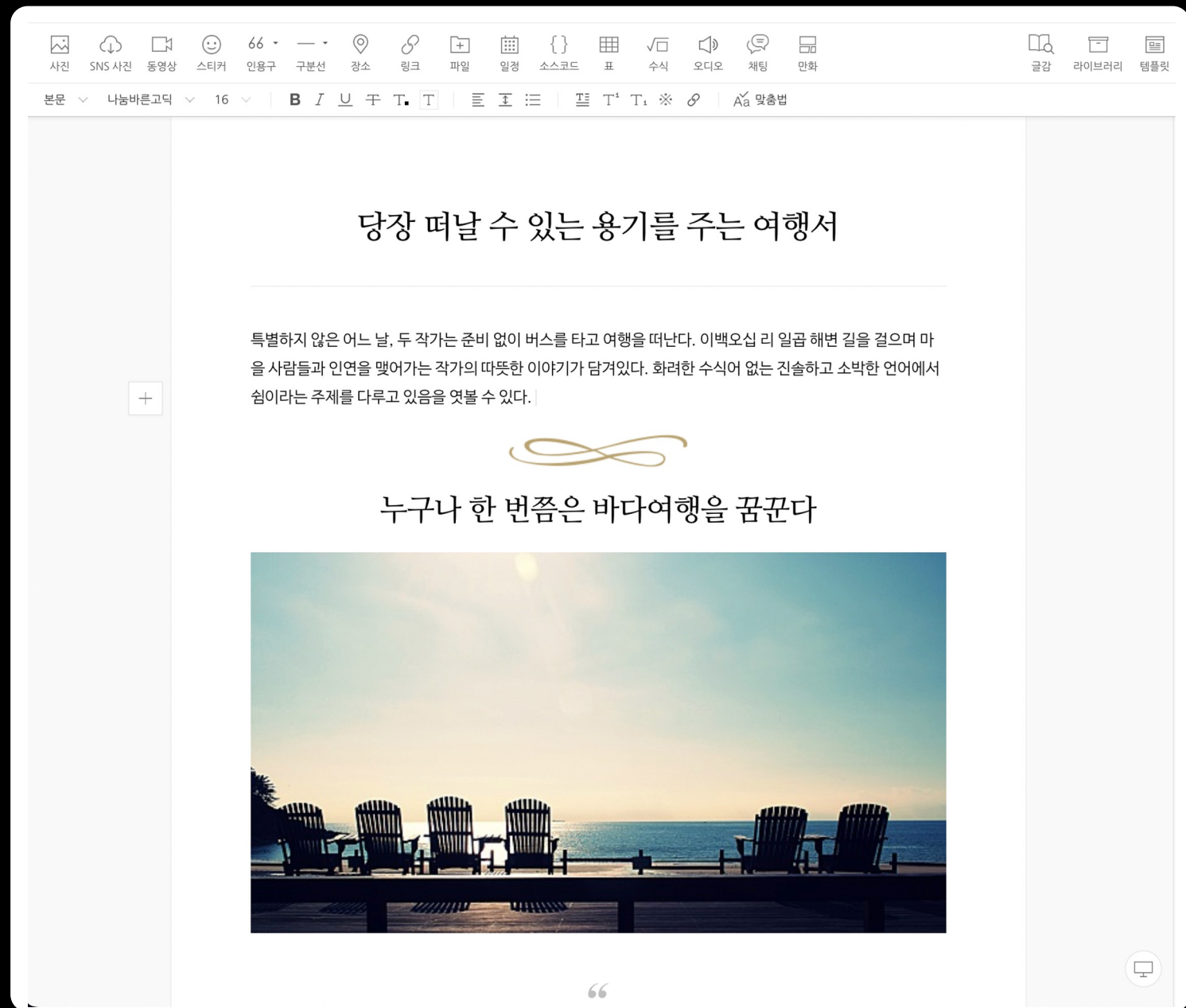
컴포넌트 간격

3

에디터의 문서 너비

4

다양한 테마 디자인



다양한 디자인 요구사항 대응

CSS 변수 활용 방법

1

폰트 서체 스타일

2

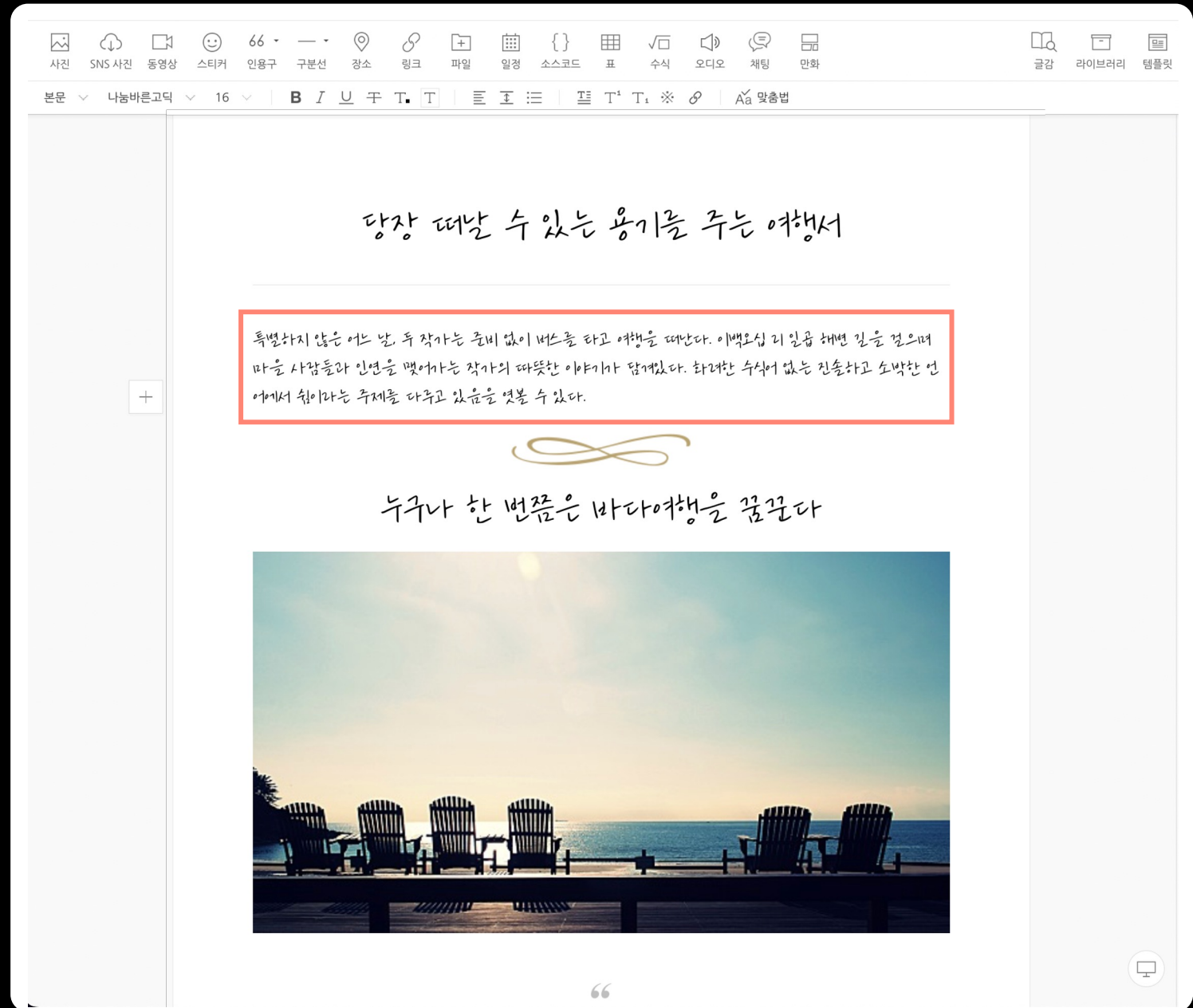
컴포넌트 간격

3

에디터의 문서 너비

4

다양한 테마 디자인



다양한 디자인 요구사항 대응

CSS 변수 활용 방법

1

폰트 서체 스타일

2

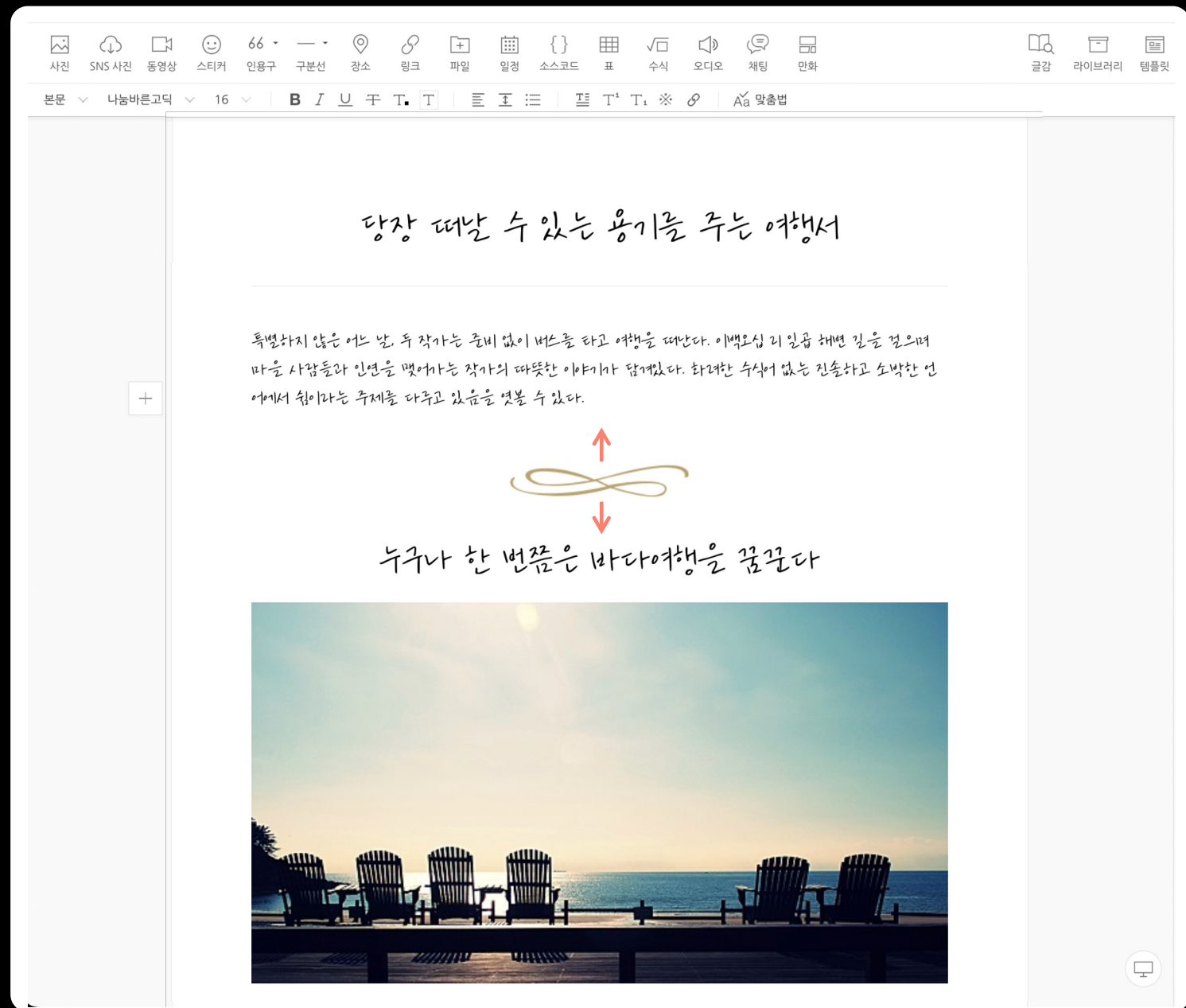
컴포넌트 간격

3

에디터의 문서 너비

4

다양한 테마 디자인



다양한 디자인 요구사항 대응

CSS 변수 활용 방법

1

폰트 서체 스타일

2

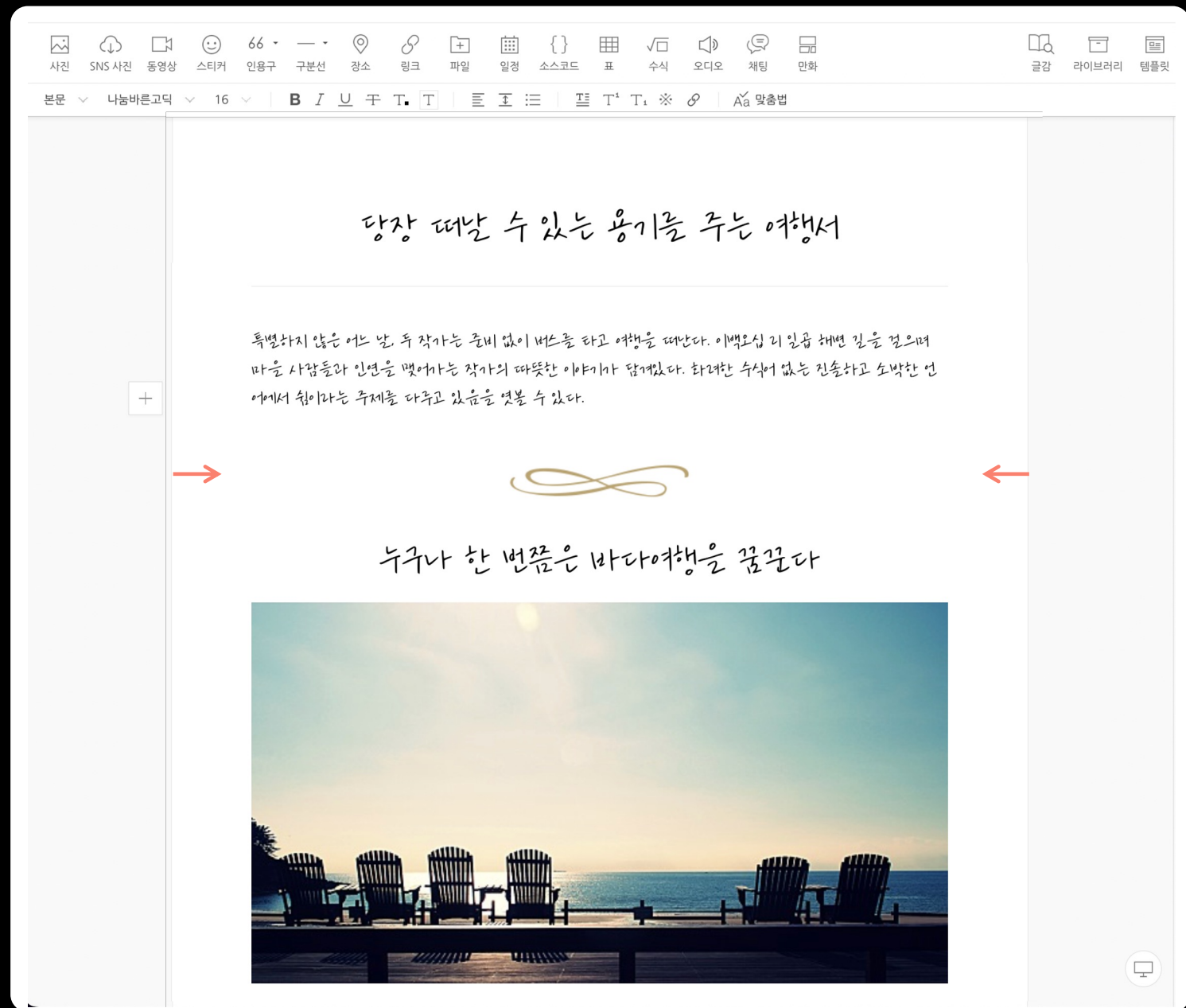
컴포넌트 간격

3

에디터의 문서 너비

4

다양한 테마 디자인



다양한 디자인 요구사항 대응

CSS 변수 활용 방법

1

폰트 서체 스타일

2

컴포넌트 간격

3

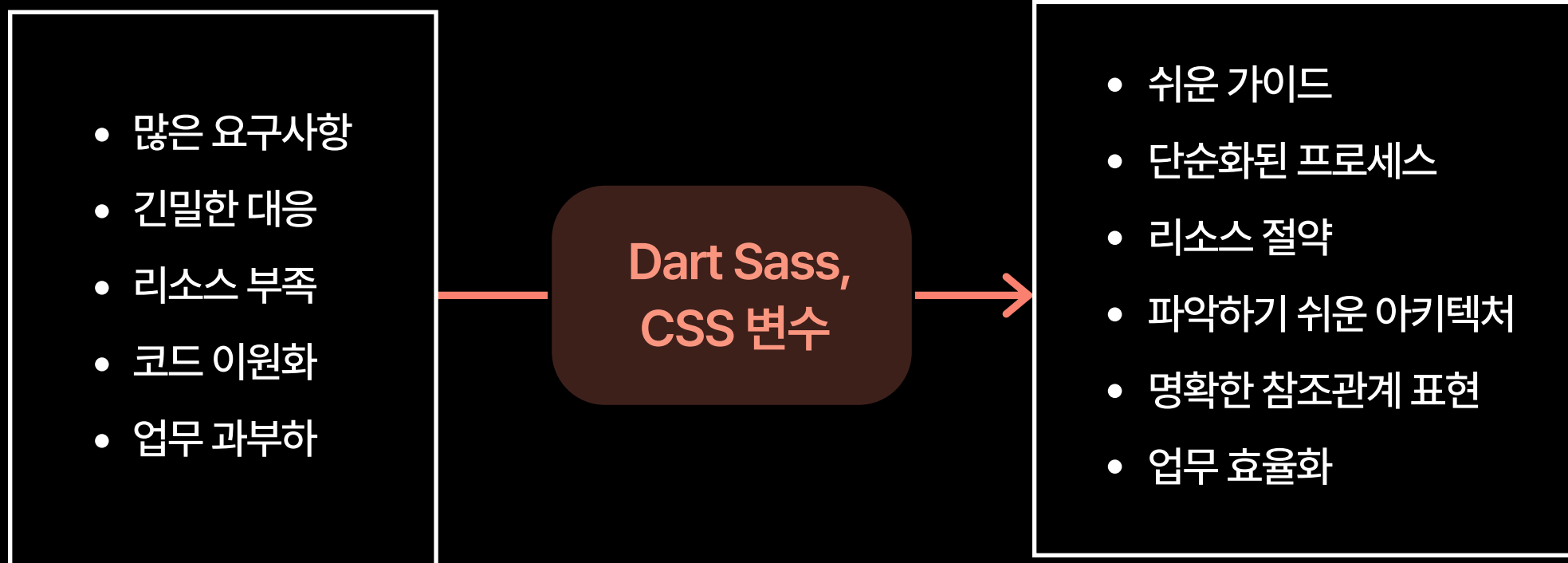
에디터의 문서 너비

4

다양한 테마 디자인



스마트에디터 UI 개발



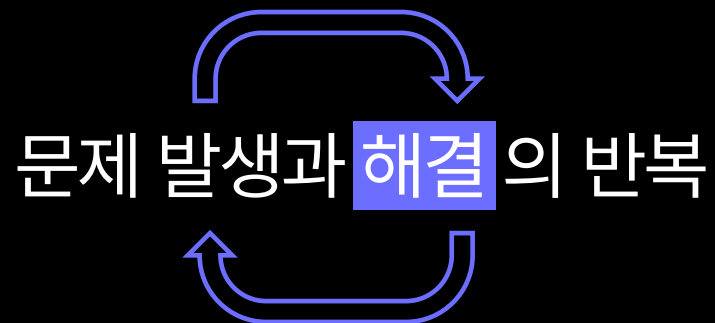
2-3

시도 정리

스마트에디터의 시도들

중요한 건 꺾이지 않는 마음

NAVER
DEVVIEW
2023



SE 2.0

멀티디바이스
대응의 어려움

SE 3.0

문서 모델링,
블록별 편집

어색한
Selection

SE ONE

가상커서,
input buffer,
History, Chunk

브라우저 네이티브
활용의 어려움

SE ONE V2

Upstream & Downstream,
Action

CSS 전처리기
Sass 도입

Sass 템플릿
플러그인 제공

커스텀 스타일
대응의 어려움

CSS 변수 도입
Dart Sass 변경

2-4

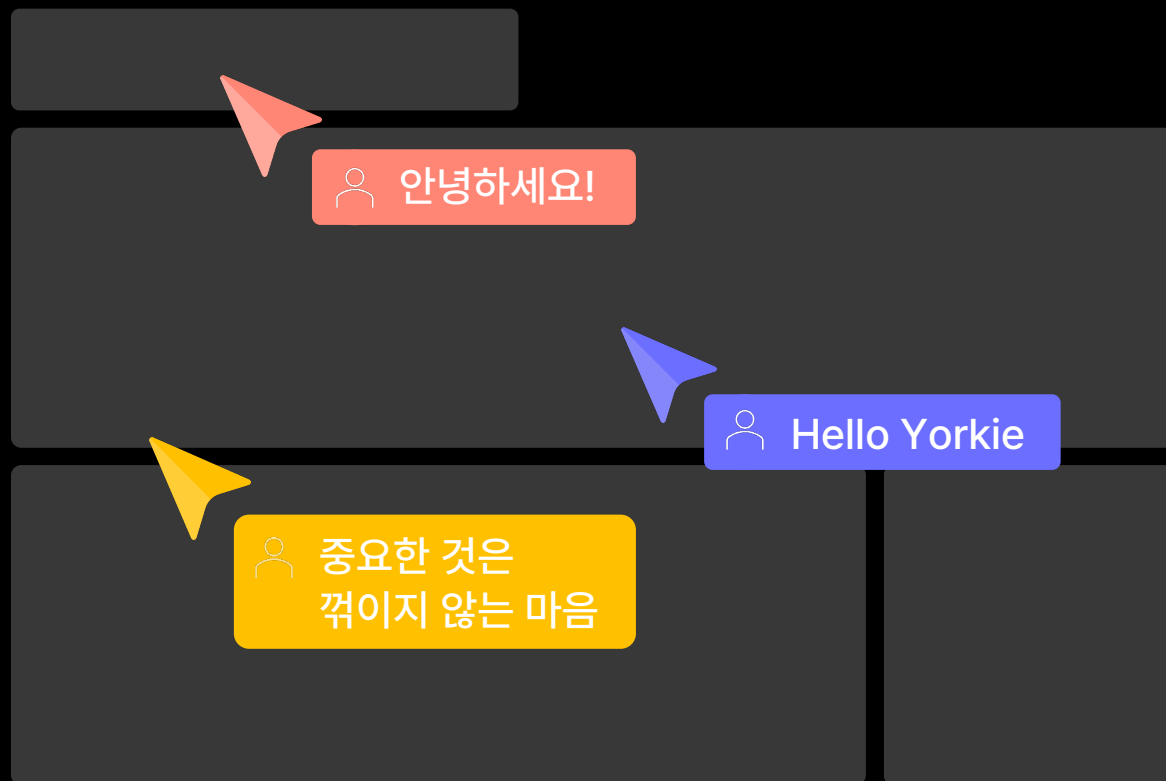
스마트에디터의 미래



동시편집

Yorkie

NAVER
DEVIEW
2023



- **Google**

OT 방식, 100명 동시 편집 가능

- **Yorkie**

CRDT 방식, 300명 이상 동시 편집 가능

<https://yorkie.dev/dashboard/>

<https://github.com/yorkie-team/yorkie>

NAVER

ETECH

에서 함께 성장하실 분을 모십니다.

NAVER
DEVIEW
2023

Channels

- ETECH 직무 소개 : <https://naver-career.gitbook.io/kr/service/etech>
- ETECH 기술 문의 : etech@navercorp.com
- ETECH 채용 문의 : etech-recruit@navercorp.com



ETECH 직무 소개

Q & A

SmartEditor 기술/채용 문의

smarteditor-recruit@navercorp.com

SmartEditor 직무소개

<https://naver-career.gitbook.io/kr/service/etech#smartstudio>

감사합니다.

PPT 제작

SmartEditor: 김주혜 / 노용구