**HTML5 Canvas 기반의 라이브러리를 활용한**

**오픈소스 이미지 에디터 개발**

**Opensource Image Editor**

2017100845정세호

2017104038한봉훈

**요약**

HTML5 Canvas API를 이용하면 웹 브라우저에 다양한 컴퓨터 그래픽을 구현할 수 있다. 하지만 Canvas API 만으로는 객체 단위 상호작용 처리가 어렵고, 애니메이션이나 기본 구성에 있어 코드 복잡성 이 증가한다. 따라서 대부분의 경우에는 Canvas API를 직접 활용하기보다는 Canvas API 기반의 라이브 러리를 활용하는 방법을 선택한다. 본 논문에서는 이미지 편집에 초점을 맞추어, Canvas API 기반의 라이 브러리를 활용하여 오픈소스 이미지 에디터 라이브러리를 구현한다. 기존 이미지 에디터에서 제공하는 편 집 기능들을 구현 및 보완하고 추가적인 기능 또한 구현한다.

**1. 서론**

**1.1 연구배경**

Canvas API는 웹페이지에 그래픽을 표현하기 위해 HTML5 이후 등장한 API다. Canvas API는 JavaScript와 HTML <canvas> element를 통해 그래픽을 그리기 위한 수단을 제공하며, 애니메이션, 게임 그래픽, 데이터 시각화, 사진 조작 및 실시간 비디오 처리를 가능하게 한다. 비슷한 기능을 수행하던 소프트웨어 플랫폼으로 Adobe Flash가 있었으나 폐쇄적인 정책과 보안 이슈로 2021년 이후 지원이 종료되었다. 이후 Flash 기반으로 만들어진 콘텐츠를 대체하기 위한 방법의 하나로 HTML5 Canvas가 이용되고 있으며 그 이유는 다음과 같다.

Canvas는 HTML에 직접 포함되고 Canvas에 그리는 스크립트는 HTML 내부 혹은 연결된 외부 파일에 있다. 즉 Canvas는 DOM의 일부이기 때문에, 페이지의 일부와 상호작용하면서 애니메이션이 변경되도록 하는 등의 방식으로 DOM과의 직접적인 상호 작용이 가능하다. 또한 Canvas 요소는 웹 브라우저에서 기본적으로 지원되기 때문에 별도의 플러그인을 설치할 필요가 없다.

하지만 Canvas API는 객체 수준의 상호 작용 기능이 내장되어 있지 않으며, 개발자에게 렌더링에 대한 세밀한 제어를 제공하는 만큼 애니메이션이나 기본 구성 등에 대한 코드의 복잡성이 증가하는 단점이 있다. 따라서 Canvas API를 직접 사용하기 보다는 이러한 문제를 보완하는 Canvas 관련 라이브러리를 설치하여 사용하는 것이 일반적이다.

**1.2 연구목표**

Canvas API를 활용하여 제공하는 서비스 중 하나로 사용자가 웹상에서 이미지를 불러와서 편집할 수 있게 도 와주는 이미지 에디터 라이브러리가 있다. 이미 오픈소스 라이브러리로 제공되는 이미지 에디터들이 있으나, 제공되는 기능이 한정되고 해당 에디터만의 특별한 기능 을 제공하는 경우는 드물다. 따라서 이번 연구에서는 Canvas API를 바탕으로 이미지 에디터 라이브러리를 제작하되, 기존 이미지 에디터 라이브러리의 한정된 기능을 보완하고 추가적인 기능을 구현하고자 한다.

**2.관련연구**

**2.1 관련 오픈소스**

**2.1.1 Fabric.js (**[**http://fabricjs.com/**](http://fabricjs.com/)**)**

Fabric.js는 HTML5 canvas용 오픈소스 Canvas 라이브러리다. Fabric.js의 가장 강력한 기능은, 네이티브 Canvas API 호출을 추상화하고 대부분의 문제를 해결할 수 있는 객체 지향 모델을 제공한다는 점이다. Circle(원형), Triangle(삼각형), ellipses(타원형) 등의 Fabric.js가 기본적으로 제공하는 객체들은 fabric이라는 네임스페이스 하에서 각각의 명칭으로 식별되고 사용되며, event inspector를 통해 이들을 개별적으로 혹은 한 번에 다룰 수 있다. 또한 작업한 캔버스를 JSON 또는 SVG로 직렬화 할 수 있으며 이를 언제든 복원할 수 있는 기능을 제공한다.

**2.1.2 Konva.js (**[**https://konvajs.org/**](https://konvajs.org/)**)**

Konva.js는 고성능 애니메이션, 이행, 노드 네스트, 계층화, 필터링, 캐싱, 데스크톱 및 모바일 애플리케이션용 이벤트 처리 등을 지원하는 라이브러리다. 라이브러리 자체에서 노드 간 레이어 관리를 알아서 해 주기 때문에 성능 상 이점이 있으나, 객체 자체를 변형하는데 있어 Fabric.js 와 비교했을 때 수동으로 구현해야 하는 단점이 있다.

**2.1.3 Toast UI image Editor (**[**https://github.com/nhn/tui.image-editor**](https://github.com/nhn/tui.image-editor)**)**

Fabric.js 기반 오픈소스 이미지 에디터로 이미지 자르기, 회전, 그리기, 도형, 필터 등의 기능을 제공한다. 텍스트박스 삽입 시 일부 글자만 수정하는 것이 불가능하거나, 이미지를 불러왔을 때 캔버스 자체가 이미지가 되어, 이미지 위에 추가로 이미지를 불러오는 것은 불가능하다는 단점이 있다.

**2.1.4 Filterbot Image Editor (https://github.com/scaleflex/filerobot-image-editor)**

바닐라 js로 만들어졌다. 이미지 크기 조정, 회전, 밝기, 대비, 채도 및 노출 등을 쉽게 조정할 수 있으며, 사용자가 지정한 워터마크를 이미지에 삽입할 수 있다.

**2.1.5 React (https://github.com/scaleflex/filerobot-image-editor)**

UI를 컴포넌트 단위로 관리하여 동적인 웹개발을 하는데 쓰인다. NPM Trend에 따르면 근 5년간 React의 사용수가 다른 경쟁 라이브러리보다 압도적이다. 유지보수가 쉽고 라이브러리의 확장이 용이하다.

**2.2 온라인 이미지 편집 웹사이트**

**2.1.4 Pixlr (https://pixlr.com/kr/e/)**

라이브러리가 아닌 온라인으로 이미지를 편집하게 해주는 웹사이트이다. 해당 사이트에서 제공하는 기능들을 참고하여 기존 이미지 에디터 라이브러리를 개선하고자 한다. 기본적인 사진 편집 기능은 물론, 이미지 검색이나 작업 저장 및 불러오기 등의 기능을 제공한다.

**2.3 Canvas API 기반 라이브러리 선정**

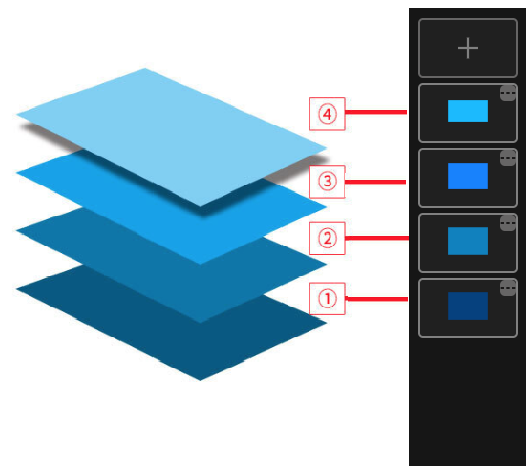
Canvas API는 그래픽 작업에 있어 강력한 기능을 제공하지만 추상화 정도가 낮다. 또한 객체 단위로 이벤트를 처리해야 하는 이미지 에디터의 특성을 고려하였을 때, 높은 추상화 모델과 객체지향 모델을 제공하는 라이브러리를 활용하는 것이 높은 생산성을 기대할 수 있다. 높은 추상화 정도와 객체 모델을 제공하는 Canvas 기반의 라이브러리로 Fabric.js, Konva.js 를 활용할 수 있음을 확인하였다. 이번 프로젝트에서는 Fabric.js 를 사용하기로 결정하였으며 그 근거는 다음과 같다.

Fabric.js는 Vue나 React 등 다른 UI 라이브러리와 호환해서 사용할 수 있으며, 필요에 따라 UI 라이브러리를 구현함으로써 독자적인 기능 향상을 기대할 수 있다. Konva.js의 경우 Fabric보다 높은 추상화 수준을 보이고 React-konva라는 라이브러리가 이미 존재한다. 이용자 수 자체는 Konva.js 가 fabric.js보다 높지만, 필요한 기능을 구현하는데 최소한의 추상화 정도를 제공하는 라이브러리를 활용하여 프로젝트를 진행하는 것이 연구 목적에 부합하다고 판단하여 Fabric.js를 프로젝트 진행에 사용할 라이브러리로 선정하였다.

**3. 프로젝트 내용**

**3.1 시나리오**

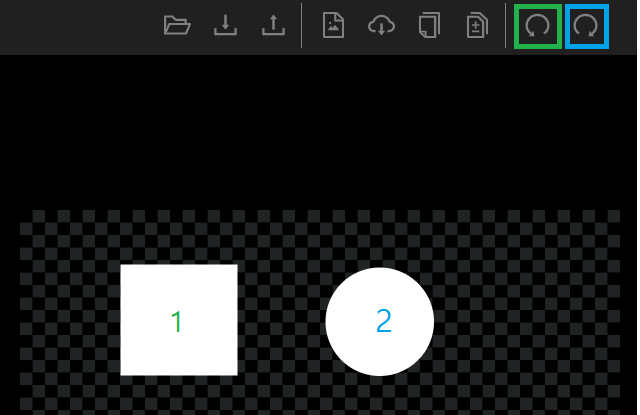
3.1.1 객체의 layer



[그림1] 객체의 layer

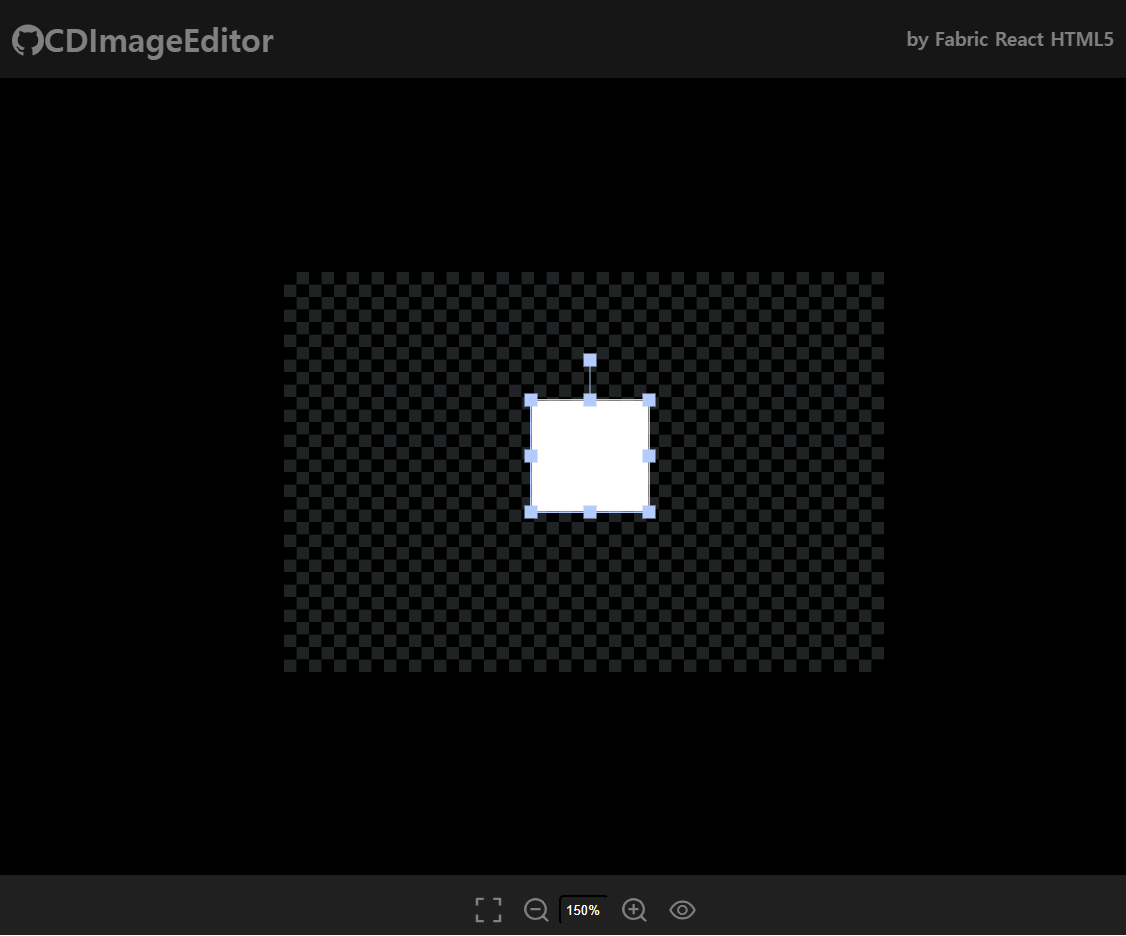
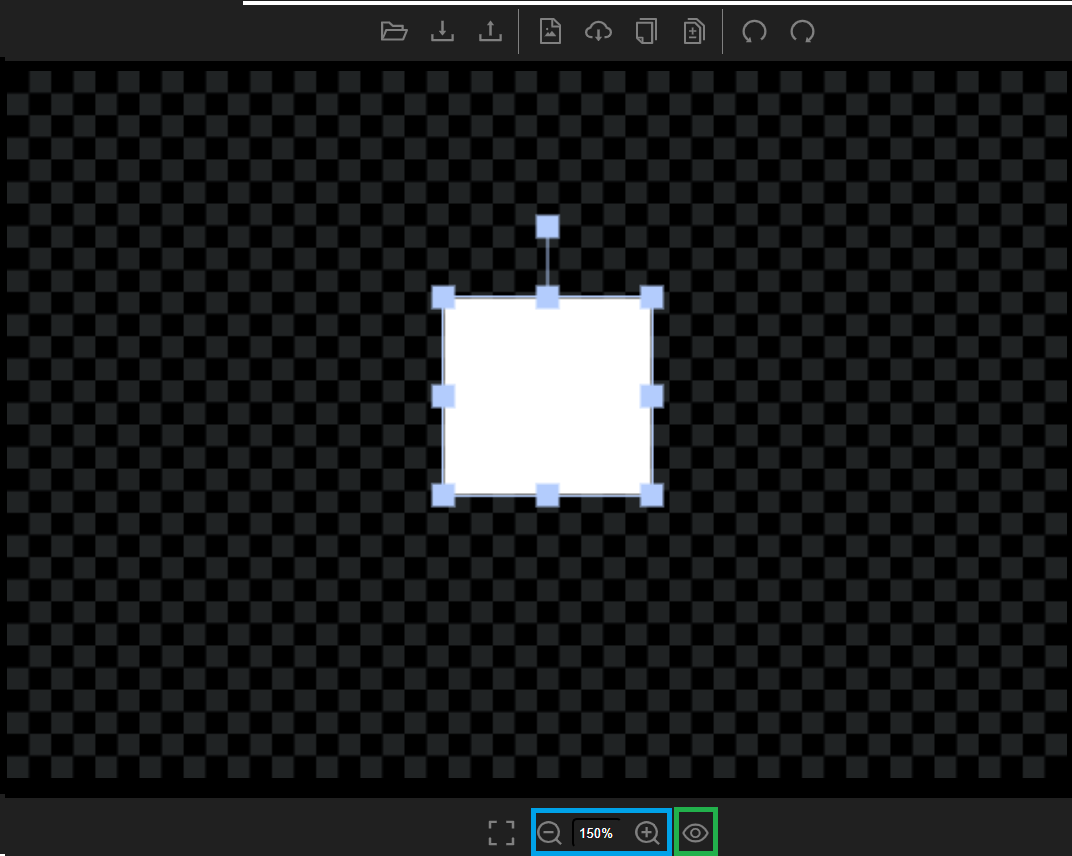
사용자는 객체를 생성하고 상호작용한다. 새 객체를 만들면 그 객체 고유의 layer와 ID가 생성되고 React의 state를 사용해 전역적으로 객체ID를 관리한다. 이 때 사용자의 객체에 대한 상호작용은 레이어에 즉각적으로 반영이 된다. 또한 레이어의 순서를 뒤바꿀시 그에 따라 각 객체의 z-index는 재정렬된다.

3.1.2 undo/redo

[그림2] undo와 redo

fabric.js 라이브러리가 제공하는 메서드를 사용하면 특정 시점의 canvas 객체를 JSON 형태로 저장하거나 불러오는 것이 가능하다. 하지만 부분적으로 필요한 내용이 아닌 캔버스 상태 전체 내용을 불러오고 저장하므로 필요 이상의 무거운 작업을 하게된다. 이를 개선하기 위해 undoStack, redoStack 배열을 각각 할당하여 이벤트가 발생했을 때 undoStack 배열에 부분적으로 필요한 내용(객체, 필터 내용 등)만을 추가한다. 이 상태에서 이전(undo)을 했을 때는 undoStack에서 가장 최근에 추가된 요소를 redoStack에 추가하고 undoStack에서 가장 나중의 index에 해당하는 캔버스 상태 정보를 캔버스에 반영한다. 되돌리기(redo)를 했을 때는 가장 나중의 index에 해당하는 요소를 undoStack에 추가하고 캔버스에 반영하는 형태로 성능을 개선하였다. [그림2]에서 초록색 undo를 누르면 1만 남고 2는 지워지며 파란색 redo를 누르면 사라졌던 2가 다시 나타난다.

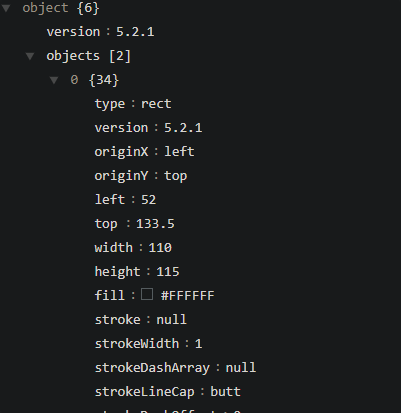
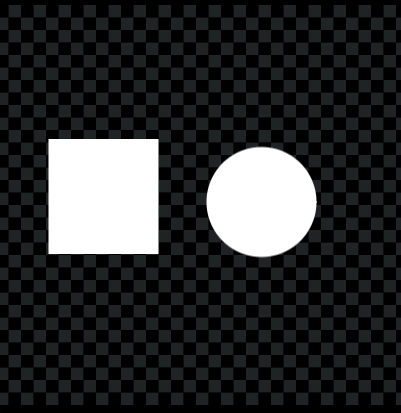
3.1.3 zoom, 미리보기



[그림3] zoom, 미리보기 [그림4] 미리보기 후 전체화면

zoom기능은 사용자의 세세한 작업을 지원함으로써 결과물 완성도에 직결된다. 마우스 휠과 버튼을 통해 canvas의 크기를 유동적으로 조절함 사용자는 직관적으로 zoom기능을 사용할 수 있다. 또한 미리보기 기능을 통해 작업물 외의 다른 UI를 제거하여 사용자의 결과물을 객관적으로 어떻게 저장될 것인지 볼 수 있다. [그림3]에서 파란색으로 표시된 부분은 zoom의 확대도를 표시하고 버튼으로도 확대를 조절할 수 있게 구현하였다. 초록색은 미리보기 버튼으로, 클릭시 하단의 UI만 남기고 나머지 UI는 사라진다.

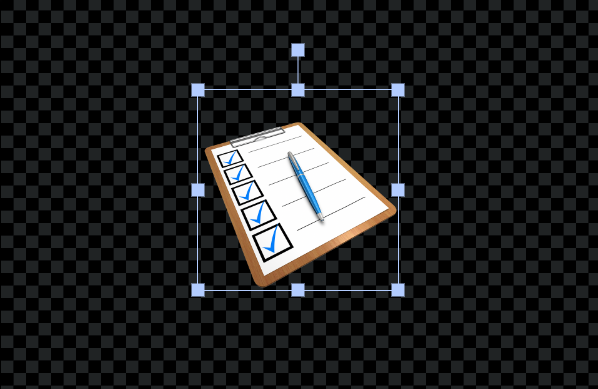
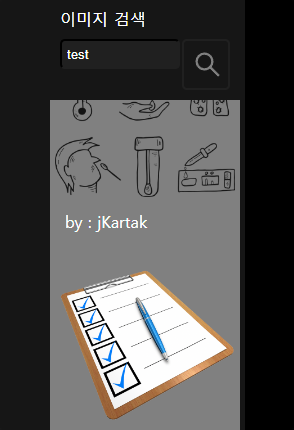
3.1.5 저장 및 복구



[그림5] 캔버스 화면 [그림6] 캔버스 직렬화 후 JSON

fabric js 내 toJSON() 과 loadFromJSON() 메서드를 활용하면 캔버스 객체를 JSON 포맷으로 직렬화하고, 직렬화한 JSON 파일을 다시 역 직렬화하는 것이 가능하다. 캔버스 자체에 정의된 이벤트뿐만 아니라, 각 객체의 위치, 크기, 객체 단위의 이벤트 등의 모든 내용을 그대로 저장하고 불러올 수 있다. 사용자가 캔버스나 객체에 추가로 정의한 attribute 또한 직렬화 및 역 직렬화 과정에서 사라지지 않고 유지되므로, 이전의 캔버스를 불러오고 작업을 이어나갈 때 충돌이 발생하지 않는다. [그림 5]에서 캔버스의 위에 두 객체가 있을 때 이를 직렬화하면 [그림 6]과 같이 캔버스와 관련한 모든 내용이 JSON파일로 저장됨을 확인할 수 있다.

3.1.6 온라인 이미지 요소 추가



[그림7] 온라인 이미지 검색 [그림8] 캔버스 위에 추가된 이미지

온라인상에서 실시간으로 이미지를 불러오는 기능을 구현하였다. Pixabay API를 활용하면 해당 사이트가 제공하는 이미지 파일을 JSON 형태로 받을 수 있다. [그림 7]과 같이 좌측 메뉴바에서 이미지 검색어를 입력했을 때 Pixabay에서 제공하는 이미지 파일 중 배경이 없는 PNG 형태의 파일 목록을 스크롤이 가능한 형태로 좌측 메뉴에 표시하였다. 그리고, 특정 객체를 클릭했을 때 캔 캔버스 위의 요소로 추가해 사용자가 활용할 수 있게 하였다(그림 8).

**4. 결론 및 기대 효과**

본 프로젝트에서는 HTML5의 Canvas API 기반의 라이브러리를 활용하여 이미지 에디터 라이브러리를 제작하였다(https://github.com/gurum505/CD1ImageEditor). 이미지 에디터 관련 라이브러리가 공통으로 제공하는 기능을 구현하였으며, 애플리케이션 레벨에서 적합한 수준의 추상화를 제공하는 라이브러리 및 스택 패턴 등을 활용하여 레이어 단위의 작업 처리와 작업의 연속성을 제공하였다.

**5. 참고문헌**

[1] “사진에디터 Pixlr X- 온라인 무료 사진 에디터.” 사 진 편집기 및 템플릿 디자이너.accessed March 22, 2022, https://pixlr.com/kr/x/#editor.

[2] 정종윤(Jongyoon Jeong),and 박성배(Seong-Bae Park). "HTML5 Canvas 기반 오픈소스 이미지 에디터 라이브 러리 개발." 한국정보과학회 학술발표논문집 2021.12 (2021): 1318-1320.

[3] "LogRocket", 'When to use HTML5's canvas', last modified Arpil 8. 2019, accessed March 30,2022,https://blog.logrocket.com/when-to-use-html 5s-canvas-ce992b100ee8/.

[4] "NAVER D2",'Chrome의 Flash 차단 정책과 Flash에 서 Canvas로 전환 사례', 2016년 10월 20일, 2022년 4월22일접속,https://d2.naver.com/helloworld/1899560.