React에서 Canvas로 인터렉티브한 배경 만들기

YOURSSU 박현우

새로운 상상을 하고, 상상을 현실로 만드는 개발자

박현우

Park Hyunwoo

2004 . 07 . 08

hyuns@hyuns.dev

숭실대학교 글로벌미디어학부 23학번

Work

2024.01 ~ 현재

UniBook 유니북

CoFounder

2022.01 ~현재

Opize 오피즈

Founder

Activities / Project

2023.03 ~ 현재

YOURSSU WebFE 멤버

2024.01 ~ 현재

숭실대학교 총학생회 디지털혁신국

2024

BlueSSU Maintainer

2023

Moyeo 모여! 팀 멤버

2023

싱가포르 글로벌 창업 인재 육성 프로그램 참여

2023

2023 SBLD 컨퍼런스 Maintainer

Awards

2023

숭실대학교 IT 프로젝트 우수상

2023

2023 전국 대학생 SW 창업 아이디어톤 대상

2022

2022 공공데이터 활용 서비스 개발 공모전 최우수상

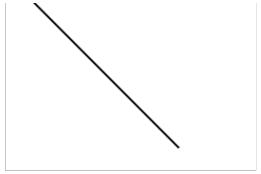
현우공간

https://hyuns.space

목차

Table of Contents

HTML에서 대각선 ／을 표시하려면?



```
<style>
  .diagonal {
    width: 200px;
    height: 1px;
    border-bottom: solid 1px #000000;
    transform: rotate(45deg);
  }
</style>

<div class="diagonal"></div>
```

이렇게 구현할 수는 있습니다.
(레이아웃은 다른 이야기지만...)

그렇다면 더욱 복잡한 그래픽을 표시해야 한다면?

그래픽을 표현할 수 있는 방법이 여러 있지만

- <img> 태그 사용
- 와 css 이용
- <svg> 태그 사용
- Flash 이용(이었던 것)
- ...

큰 한계가 있습니다

애니메이션을 표현하기 어려움

복잡한 그래픽 표시시 성능이 저하된다

인터렉션 구현이 어렵다

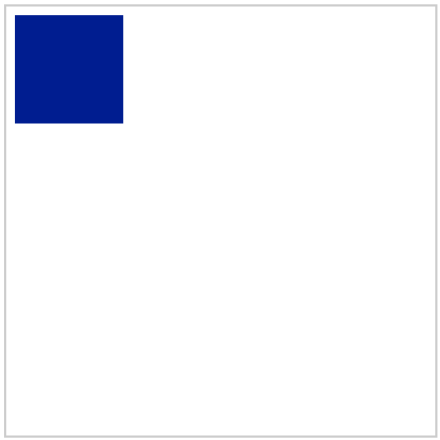
...

Canvas

<Canvas>

Canvas API 혹은 WebGL API를 활용해서
HTML에 그래픽과 애니메이션을 그릴 수 있는 엘리먼트

Canvas API 사용 방법

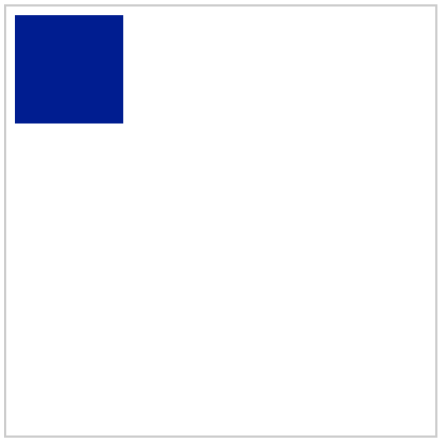


```html
<canvas id="canvas" width="400" height="400"></canvas>
<script>
  const canvas = document.getElementById("canvas");
  const ctx = canvas.getContext("2d");
  ctx.fillStyle = "#001C8A";
  ctx.fillRect(10, 10, 100, 100);
</script>
```

HTML에 Canvas 태그 추가

(width와 height 속성은 필수)

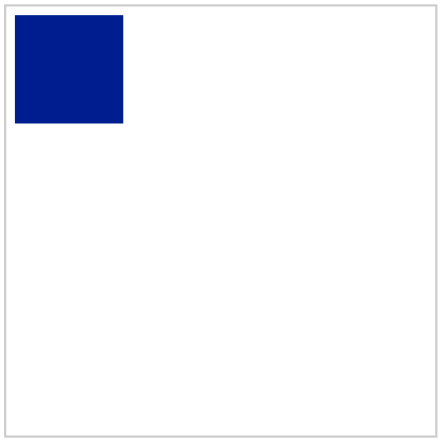
Canvas API 사용 방법



```
<canvas id="canvas" width="400" height="400"></canvas>
<script>
  const canvas = document.getElementById("canvas");
  const ctx = canvas.getContext("2d");
  ctx.fillStyle = "#001C8A";
  ctx.fillRect(10, 10, 100, 100);
</script>
```

canvas 엘리먼트에서 getContext()를 호출하여
canvas Context 획득

Canvas API 사용 방법



```
<canvas id="canvas" width="400" height="400"></canvas>
<script>
  const canvas = document.getElementById("canvas");
  const ctx = canvas.getContext("2d");
  ctx.fillStyle = "#001C8A";
  ctx.fillRect(10, 10, 100, 100);
</script>
```

fillStyle 속성과 fillRect() 메소드로 캔버스에 도형 그리기

Canvas API 사용 방법

```
<canvas id="canvas" width="400" height="400"></canvas>
<script>
  const canvas = document.getElementById("canvas");
  const ctx = canvas.getContext("2d");
  ctx.fillStyle = "#001C8A";
  ctx.fillRect(10, 10, 100, 100);
</script>
```

Artwork

fff. - form follows function

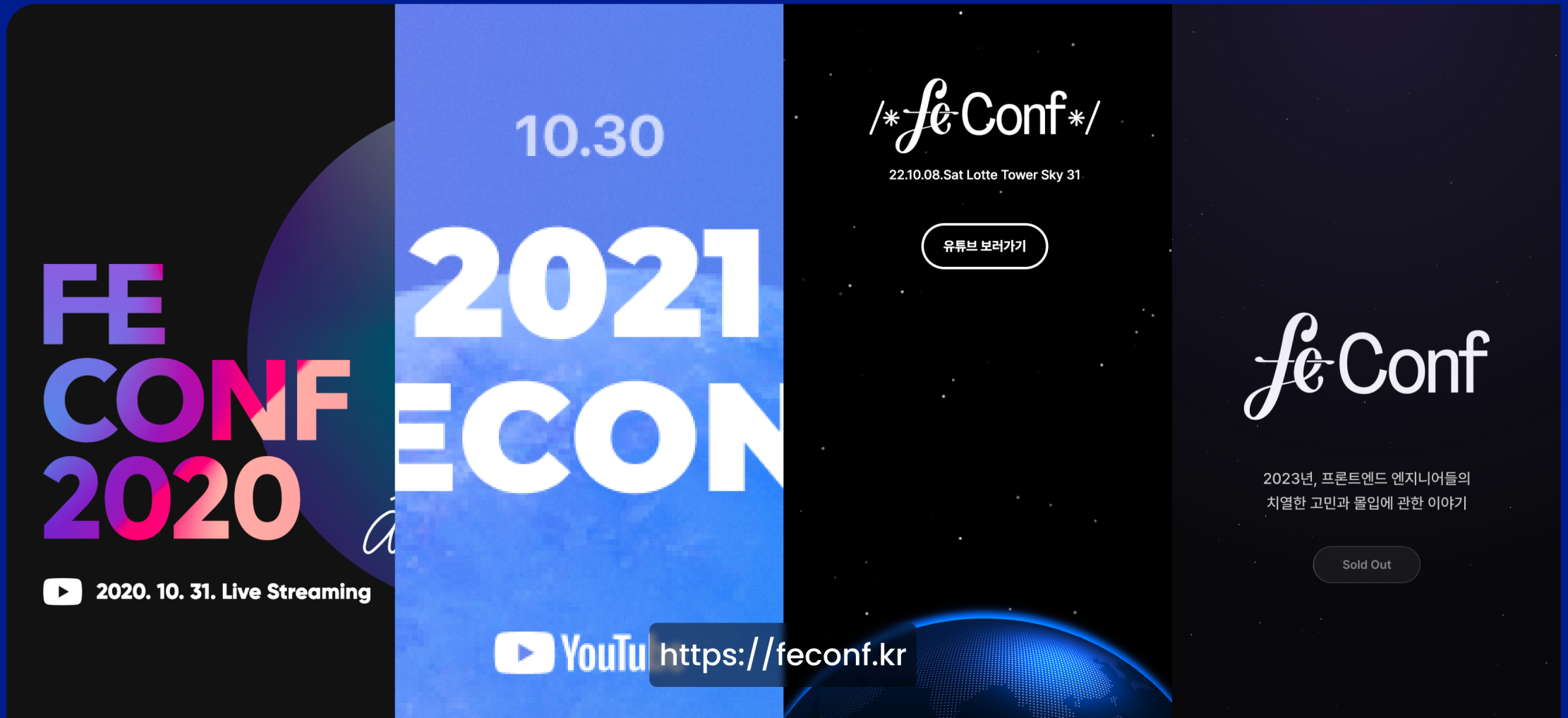
by 김종민(Interactive Developer)

matter-js

자바스크립트 물리엔진 라이브러리

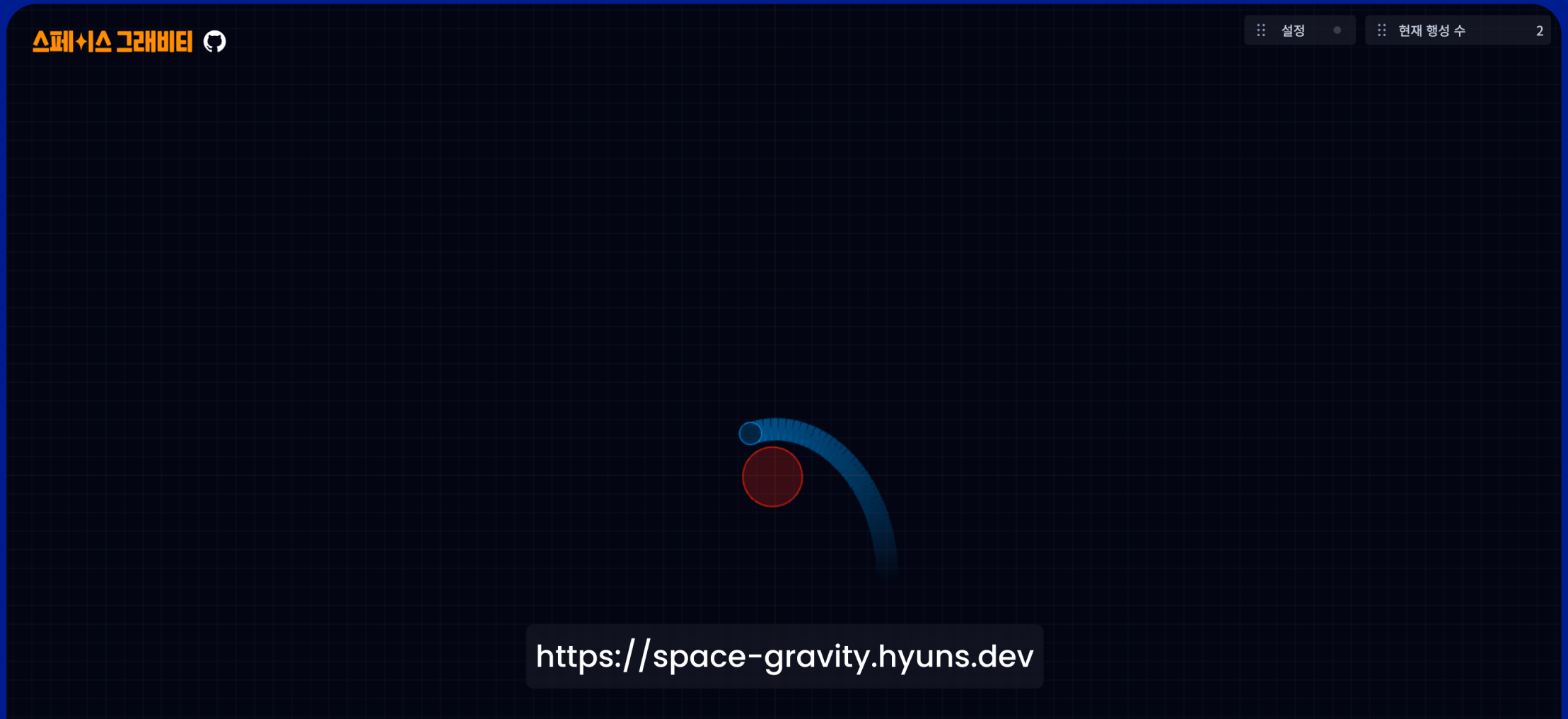
FEConf

대한민국 프론트엔드 개발자 컨퍼런스 웹사이트



스페이스 그래비티

직접 만든 웹 우주 중력 시뮬레이터



체험해보기

체험해보기

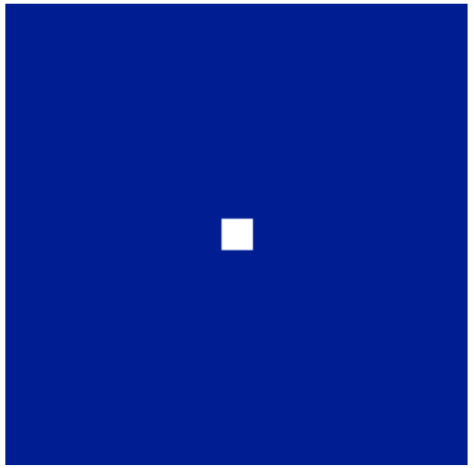
https://canvas-in-react.hyuns.dev

Canvas In React

Canvas In React

가상 DOM을 사용하는 React의 특성상
Canvas를 사용하려면 추가적인 코드가 필요하다

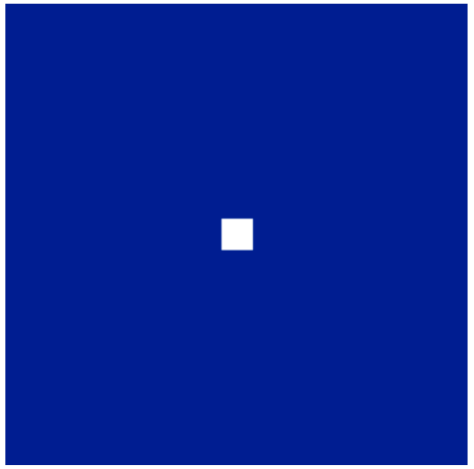
Canvas In React



```tsx
function Canvas() {
  const ref = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = ref.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    ctx.fillStyle = "#001C8A";
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(canvas.width / 2 - 10, canvas.height / 2 - 1, 20, 20);
  }, [ref]);

  return <canvas ref={ref} width={300} height={300} />;
}
```

Canvas DOM에 접근하기 위해 useRef( )와 엘리먼트 작성

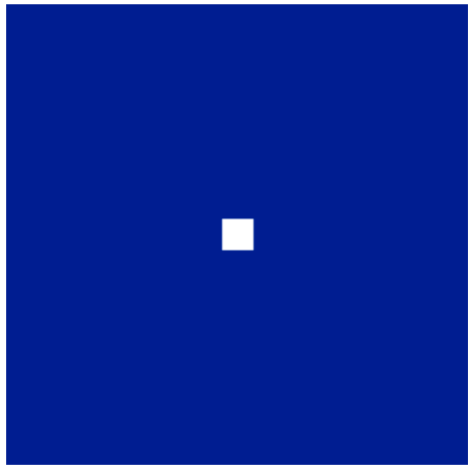
Canvas In React



```tsx
function Canvas() {
  const ref = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = ref.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    ctx.fillStyle = "#001C8A";
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(canvas.width / 2 - 10, canvas.height / 2 - 1, 20, 20);
  }, [ref]);

  return <canvas ref={ref} width={300} height={300} />;
}
```

useEffect() 내부에서 dom와 context 초기화

Canvas In React



```tsx
function Canvas() {
  const ref = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = ref.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    ctx.fillStyle = "#001C8A";
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(canvas.width / 2 - 10, canvas.height / 2 - 1, 20, 20);
  }, [ref]);

  return <canvas ref={ref} width={300} height={300} />;
}
```

canvas Context를 이용하여 Canvas에 그래픽 표시

Animation

Animation

프레임에 맞춰 화면을 새로 그려
애니메이션을 구현할 수 있다

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

개발 편의를 위한 코드 분리

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

Animation을 위한 필수 변수

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

draw()

실제로 화면을 그리는 메소드

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

animate()

재귀적으로 this.draw()를 호출하여 프레임을 그리는 메소드

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

run()

초기화, 실행 메소드

Animation

```
class Animation {
  time: number = 0;
  frameId: number = 0;
  constructor(private ctx: CanvasRenderingContext2D) {}

  draw() { ... }

  animate() {
    this.draw();
    this.frameId = requestAnimationFrame(this.animate.bind(this));
  }

  run() {
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
  }
}
```

stop()

애니메이션 종료 메소드

requestAnimationFrame()

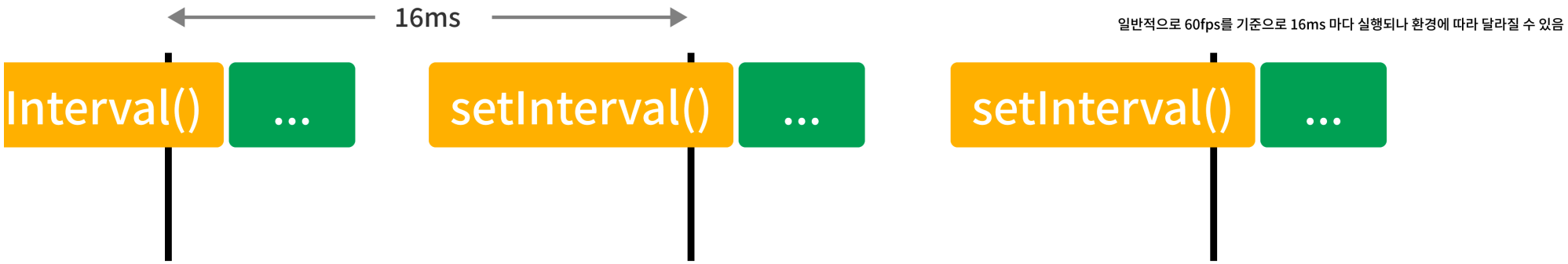
브라우저에게 “다음 프레임"에 콜백 함수를 실행하도록 요청하는 함수

```
animate() {
  this.draw();
  this.frameId = requestAnimationFrame(this.animate.bind(this));
}
```

requestAnimationFrame()

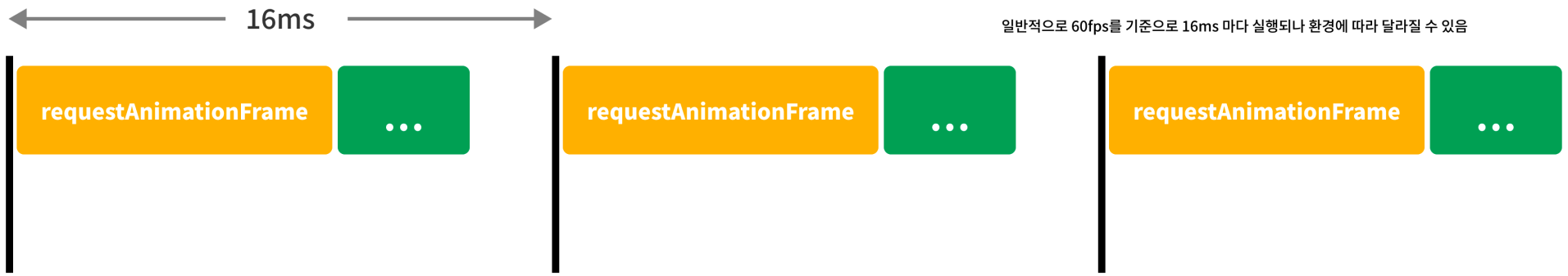
왜 `setInterval()`이나 `setTimeout()`을 사용하지 않나요?

requestAnimationFrame()



`setInterval()`이나 `setTimeout()`의 경우
자바스크립트가 화면을 그리는 주기와 맞지 않아
화면이 끊기거나 불필요한 프레임이 생길 수 있다

requestAnimationFrame()

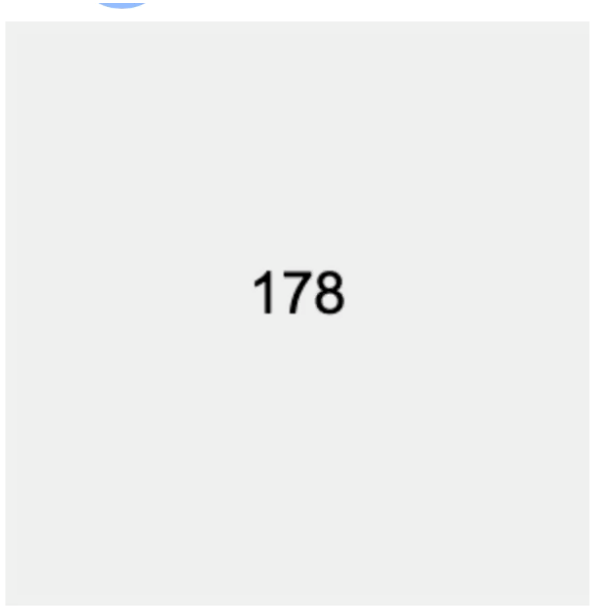


`requestAnimationFrame()`은 화면이 갱신되는 주기에 맞춰 실행되므로, 끊임없이 재생하고, 불필요한 실행을 막을 수 있다.

requestAnimationFrame()

1. `requestAnimationFrame()`은 디스플레이 주사율에 맞춰 실행된다.
2. 현재 창에 표시되지 않거나 브라우저의 정책(주로 절전 관련)에 따라 실행되지 않거나 느리게 실행될 수 있다.
3. 자동으로 반복되는 것이 아닌 재귀적으로 호출해야 한다.
4. `cancelAnimationFrame()`은 요청을 취소한다.

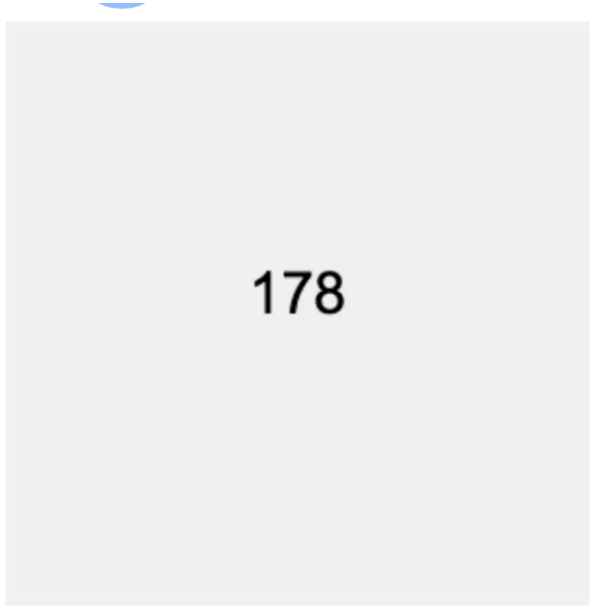
Animation




```tsx
function Canvas() {
  const ref = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = ref.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    const animation = new Animation(ctx);
    animation.run();

    return () => {
      animation.stop();
    };
  }, [ref]);

  return <canvas ref={ref} width={300} height={300} />;
}
```

인스턴스를 생성하여 애니메이션 `run()`

Animation




```tsx
function Canvas() {
  const ref = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = ref.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    const animation = new Animation(ctx);
    animation.run();

    return () => {
      animation.stop();
    };
  }, [ref]);

  return <canvas ref={ref} width={300} height={300} />;
}
```

컴포넌트가 디마운트되더라도 감지할 수 없으므로

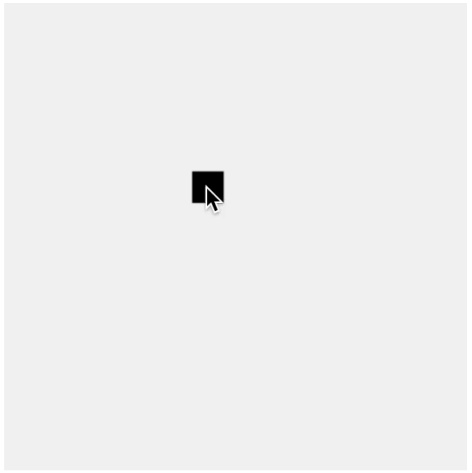
useEffect()의 cleanUp에 애니메이션 stop() 추가

Event Listener

만약 마우스를 따라가는 그래픽을 그리려면?

Event Listener

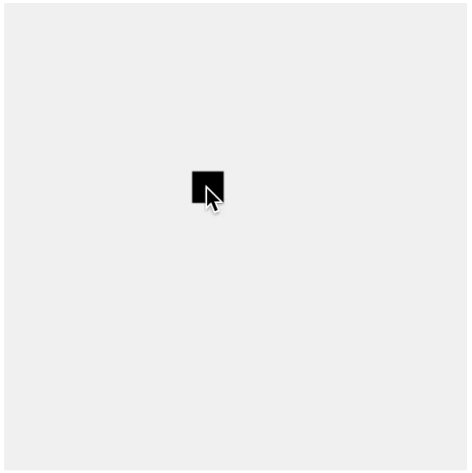
Event Listener



마우스 위치 관련
변수&함수 추가

```
class Animation {
  ...
  mousePos: { x: number; y: number } = { x: 0, y: 0 };

  updateMousePos(e: MouseEvent) {
    this.mousePos = { x: e.offsetX, y: e.offsetY };
  }

  draw() {
    ...
    this.ctx.fillStyle = "#000000";
    this.ctx.fillRect(this.mousePos.x - 10, this.mousePos.y - 10, 20, 20);
  }

  animate() { ... }

  run() {
    this.ctx.canvas.addEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    this.ctx.canvas.removeEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
  }
}
```

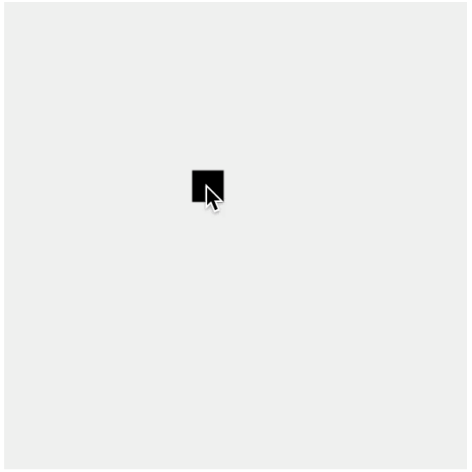
Event Listener



해당 변수를 draw()에서 사용

```
class Animation {
  ...
  mousePos: { x: number; y: number } = { x: 0, y: 0 };

  updateMousePos(e: MouseEvent) {
    this.mousePos = { x: e.offsetX, y: e.offsetY };
  }

  draw() {
    ...
    this.ctx.fillStyle = "#000000";
    this.ctx.fillRect(this.mousePos.x - 10, this.mousePos.y - 10, 20, 20);
  }

  animate() { ... }

  run() {
    this.ctx.canvas.addEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    this.ctx.canvas.removeEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
  }
}
```

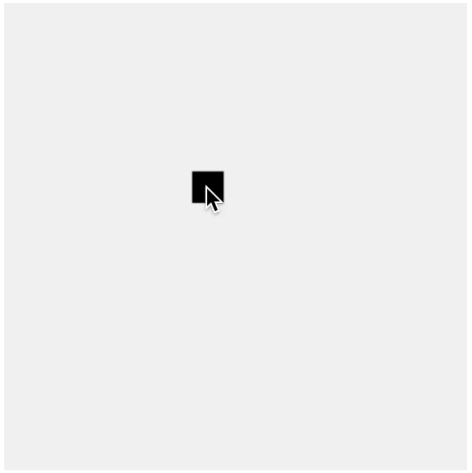
Event Listener



EventListener를 추가해
이벤트 리스닝

```
class Animation {
  ...
  mousePos: { x: number; y: number } = { x: 0, y: 0 };

  updateMousePos(e: MouseEvent) {
    this.mousePos = { x: e.offsetX, y: e.offsetY };
  }

  draw() {
    ...
    this.ctx.fillStyle = "#000000";
    this.ctx.fillRect(this.mousePos.x - 10, this.mousePos.y - 10, 20, 20);
  }

  animate() { ... }

  run() {
    this.ctx.canvas.addEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    this.ctx.canvas.removeEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
  }
}
```

Event Listener



애니메이션 종료시
이벤트 리스너 삭제

```
class Animation {
  ...
  mousePos: { x: number; y: number } = { x: 0, y: 0 };

  updateMousePos(e: MouseEvent) {
    this.mousePos = { x: e.offsetX, y: e.offsetY };
  }

  draw() {
    ...
    this.ctx.fillStyle = "#000000";
    this.ctx.fillRect(this.mousePos.x - 10, this.mousePos.y - 10, 20, 20);
  }

  animate() { ... }

  run() {
    this.ctx.canvas.addEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    this.ctx.canvas.removeEventListener(
      "mousemove",
      this.updateMousePos.bind(this)
    );
  }
}
```

Full Screen

Canvas는 기본적으로 크기를 구체적으로 정해야 한다.
즉, 처음 화면 크기로 맞출 경우 화면 크기가 변경되었을 때 반영되지 않는다.

화면 크기가 변할 때 마다 자바스크립트를 통해
Canvas의 크기를 변경시켜야 한다!

Event Listener

canvas의 크기를
window의 크기로 설정
하는 메소드

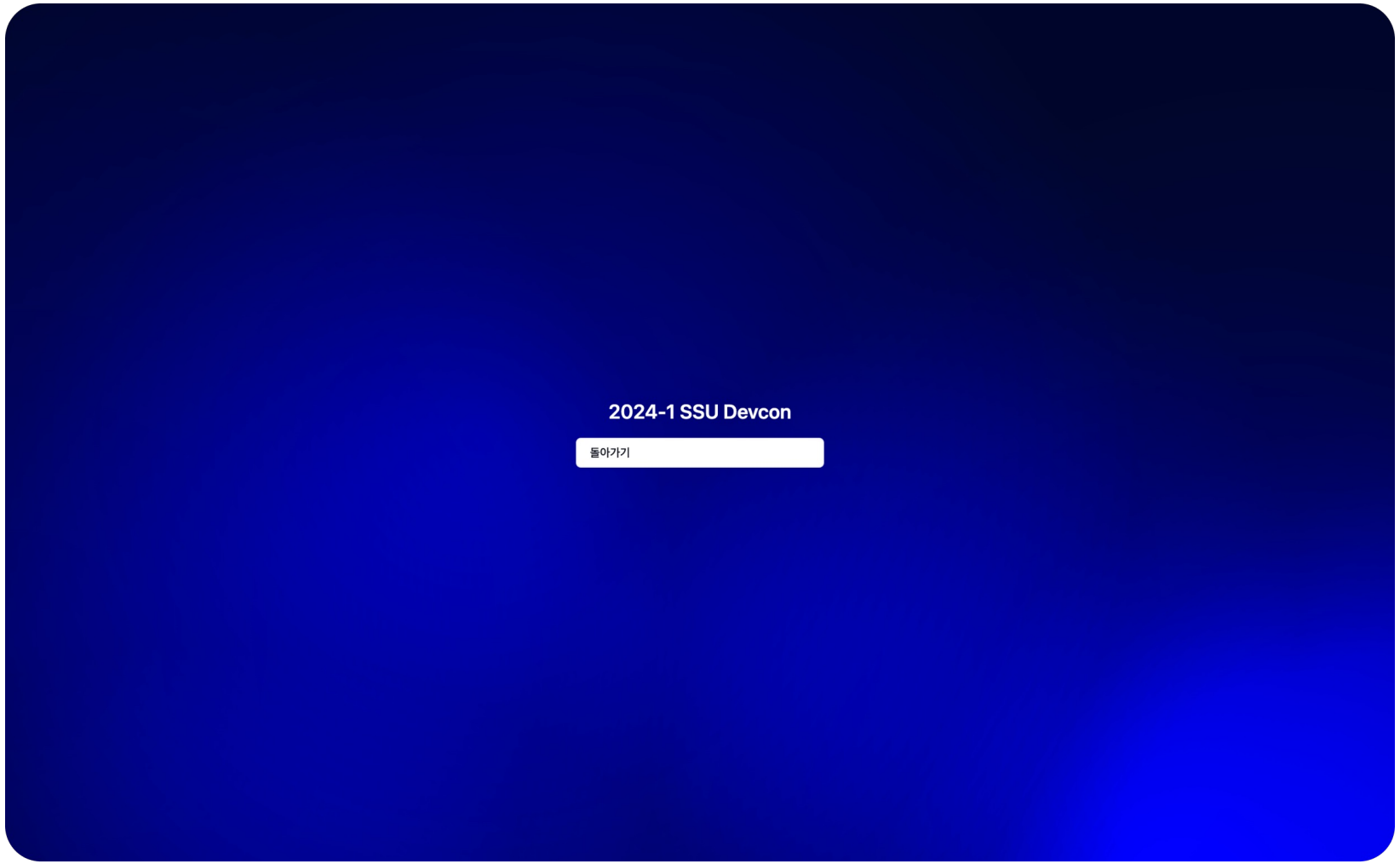
```
class Animation {
  ...
  constructor(private ctx: CanvasRenderingContext2D) {}
  draw() { ... }
  animate() { ... }

  resize() {
    this.ctx.canvas.width = window.innerWidth;
    this.ctx.canvas.height = window.innerHeight;
  }

  run() {
    this.resize();
    window.addEventListener("mousemove", this.updateMousePos.bind(this));
    window.addEventListener("resize", this.resize.bind(this));
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    window.removeEventListener("mousemove", this.updateMousePos);
    window.removeEventListener("resize", this.resize);
  }
}
```

Event Listener

window resize
이벤트 리스너 추가

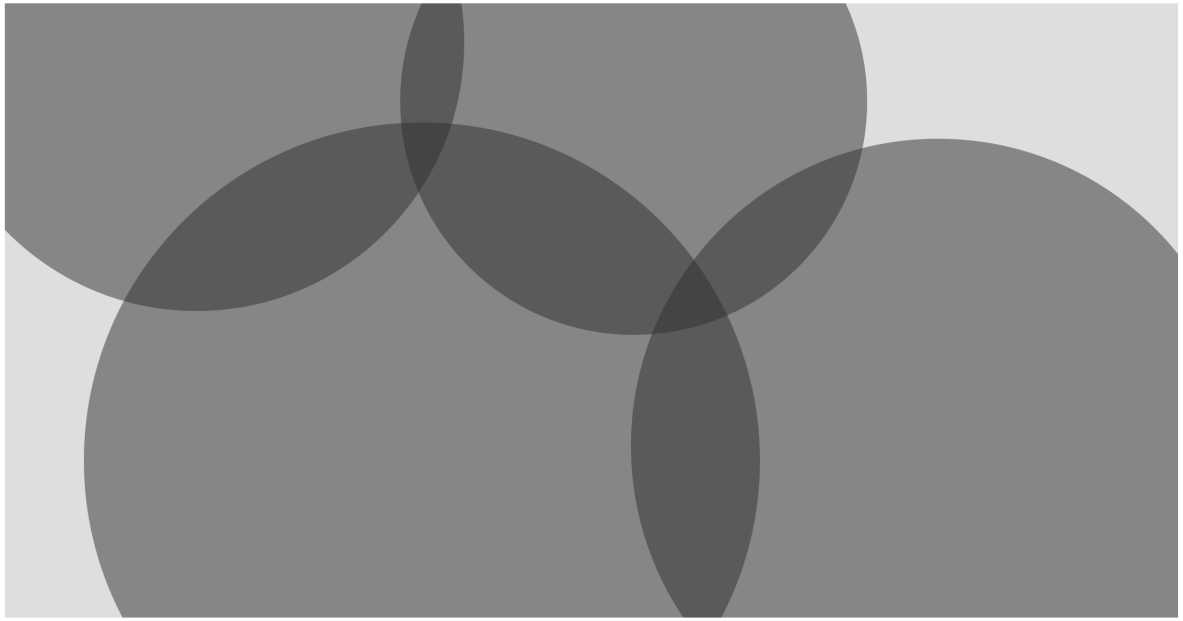
```
class Animation {
  ...
  constructor(private ctx: CanvasRenderingContext2D) {}
  draw() { ... }
  animate() { ... }

  resize() {
    this.ctx.canvas.width = window.innerWidth;
    this.ctx.canvas.height = window.innerHeight;
  }

  run() {
    this.resize();
    window.addEventListener("mousemove", this.updateMousePos.bind(this));
    window.addEventListener("resize", this.resize.bind(this));
    this.animate();
  }

  stop() {
    cancelAnimationFrame(this.frameId);
    window.removeEventListener("mousemove", this.updateMousePos);
    window.removeEventListener("resize", this.resize);
  }
}
```

Background


2024-1 SSU Devcon
돌아가기

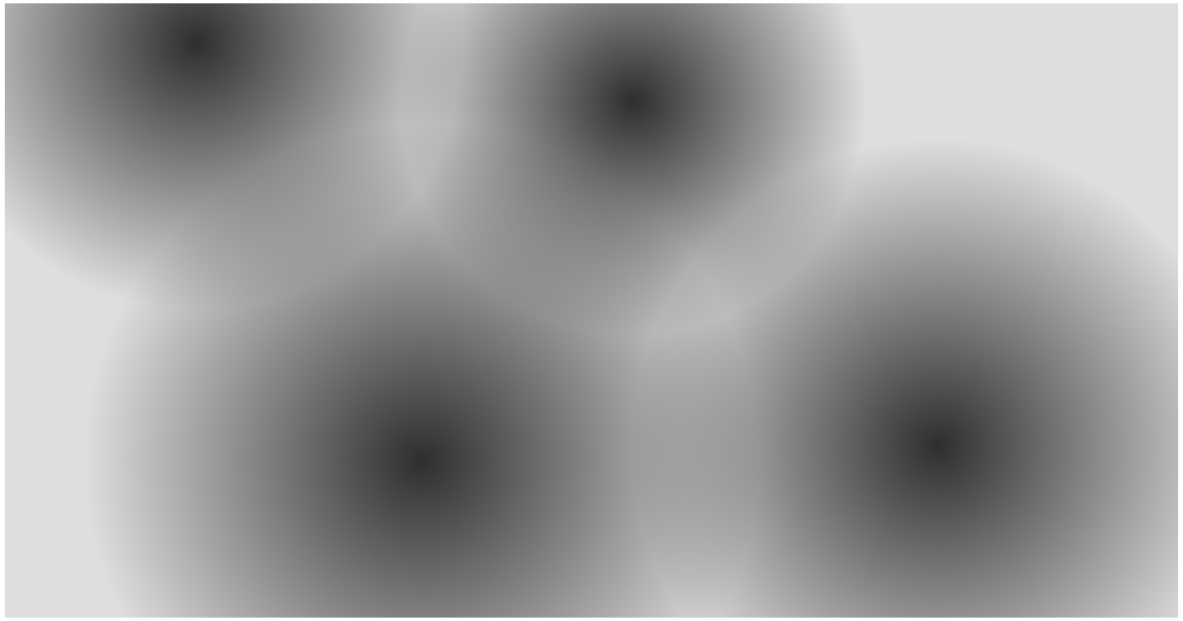
개인 웹사이트에 적용된 모습

구현 방법



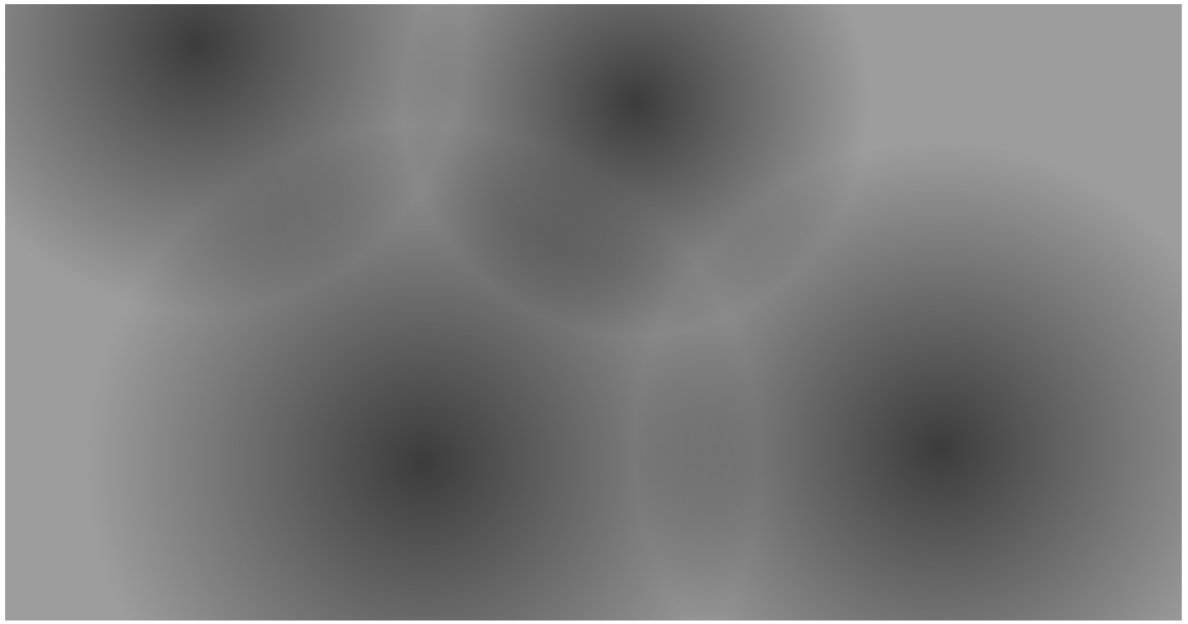
화면 전체를 Canvas로 설정하고, 그 안에 랜덤한 위치에 원을 추가한다.

구현 방법



시작으로부터 멀어질 수록 불투명도가 감소하는 그라데이션을 추가한다

구현 방법



compositeOperation을 “overlay” 로 설명하고
색을 적절히 조절한다

구현 방법

프레임이 진행되는 것에 맞추어 원의 위치를 변화시킨다

감사합니다

YOURSSU 박현우