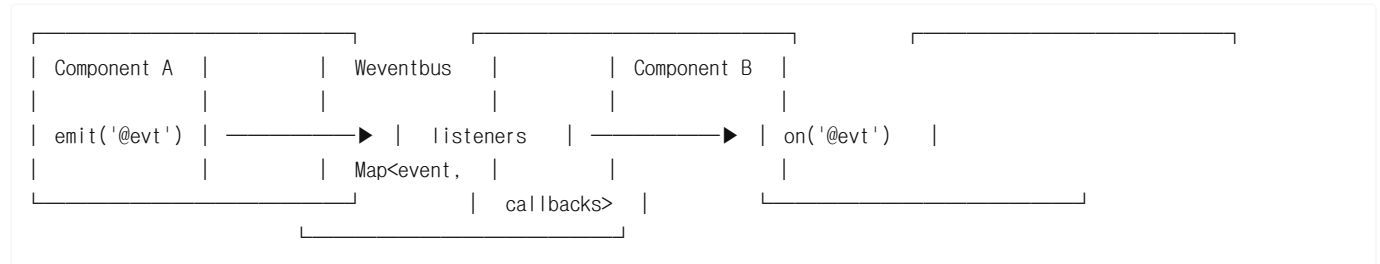


Weventbus API Reference

컴포넌트 간 통신을 위한 이벤트 버스.

개요



API

on(event, callback)

이벤트 리스너 등록.

```
/**
 * @param {string} event - 이벤트 이름
 * @param {Function} callback - 이벤트 핸들러
 */
Weventbus.on('@buttonClicked', (data) => {
  console.log('버튼 클릭:', data);
});
```

동일 이벤트에 여러 핸들러 등록 가능:

```
Weventbus.on('@dataLoaded', handlerA);
Weventbus.on('@dataLoaded', handlerB);
// 둘 다 호출됨
```

off(event, callback)


이벤트 리스너 해제.

```
/**
 * @param {string} event - 이벤트 이름
 * @param {Function} callback - 해제할 핸들러 (참조 동일해야 함)
 */
const handler = (data) => console.log(data);

Weventbus.on('@myEvent', handler);
Weventbus.off('@myEvent', handler); // 정확한 함수 참조 필요
```

주의: 익명 함수는 해제 불가.

```
// ❌ 잘못됨 - 해제 불가
Weventbus.on('@evt', (data) => console.log(data));
Weventbus.off('@evt', (data) => console.log(data)); // 다른 함수 참조
```

```
//  올바른 - 참조 저장
const handler = (data) => console.log(data);
Weventbus.on('@evt', handler);
Weventbus.off('@evt', handler);
```

emit(event, data)

이벤트 발생.

```
/**
 * @param {string} event - 이벤트 이름
 * @param {any} data - 전달할 데이터
 */
Weventbus.emit('@buttonClicked', {
  event: clickEvent,
  targetInstance: this
});
```

once(event, callback)

1회성 이벤트 리스너.

```
/**
 * @param {string} event - 이벤트 이름
 * @param {Function} callback - 핸들러 (1회 실행 후 자동 해제)
 */
Weventbus.once('@initComplete', (data) => {
  console.log('초기화 완료 (1회만 실행)');
});
```

사용 패턴

Wkit과 함께 사용

```
// 초기화 시 - 핸들러 일괄 등록
this.eventBusHandlers = {
  '@buttonClicked': (data) => this.handleButtonClick(data),
  '@formSubmitted': (data) => this.handleFormSubmit(data)
};

Wkit.onEventBusHandlers(this.eventBusHandlers);

// 정리 시 - 핸들러 일괄 해제
Wkit.offEventBusHandlers(this.eventBusHandlers);
```

customEvents와의 관계

Wkit.bindEvents가 자동으로 Weventbus.emit을 호출합니다.

```
// customEvents 정의
this.customEvents = {
```

```
    click: {
      '.my-button': '@buttonClicked' // trigger 이벤트명
    }
  };

  // bindEvents 호출
  Wkit.bindEvents(this, this.customEvents);

  // .my-button 클릭 시 내부적으로:
  // Weventbus.emit('@buttonClicked', { event, targetInstance })

  // 따라서 핸들러에서:
  Weventbus.on('@buttonClicked', ({ event, targetInstance }) => {
    console.log('클릭된 인스턴스:', targetInstance.name);
  });
```

컴포넌트 간 통신

```
// Component A - 이벤트 발생
Wkit.emitEvent('@productSelected', this);
// 또는 직접:
Weventbus.emit('@productSelected', { productId: 123, instance: this });

// Component B - 이벤트 수신
Weventbus.on('@productSelected', ({ productId }) => {
  this.loadProductDetails(productId);
});
```

이벤트 네이밍 컨벤션

접두사	용도	예시
@	사용자 정의 이벤트	@buttonClicked, @dataLoaded
@trigger	customEvents 트리거	@triggerNavLink, @triggerSubmit

내부 구조

```
const listeners = new Map();
// Map<eventName, callback[]>
```

관련 문서

- [WKIT_API.md](#) - onEventBusHandlers, emitEvent
- [DEFAULT_JS_NAMING.md](#) - customEvents 형태