



ELUC크립트는 DOM 계층을 파니아하는기에 용이하出

```
function handleDrag(eDown: Event) {
  const targetEl = eDown.currentTarget;
  targetEl.classList.add('dragging');
  const dragStart = [eDown.clientX, eDown.clientY];
  const handleUp = (eUp: Event) => {
    targetEl.classList.remove('dragging');
    targetEl.removeEventListener('mouseup', handleUp);
    const dragEnd = [eUp.clientX, eUp.clientY];
    console.log('dx, dy = ', [0, 1].map(i => dragEnd[i] - dragStart[i]));
  }
  targetEl.addEventListener('mouseup', handleUp);
}
const div = document.getElementById('surface');
div.addEventListener('mousedown', handleDrag);
```



자바스크립트에서는 문제가 없는 코드같아보이지만, 타입스크립트에서는 많은 오류를 표시하며요

```
const handleUp = (eUp: Event) => {
    targetEl.classList.remove('dragging');
                     개체가 'null'인 것 같습니다.
            ~~~~~~ 'EventTarget' 형식에 'classList' 속성이 없습니다.
    targetEl.removeEventListener('mouseup', handleUp);
// ~~~~~~ 개체가 'null'인 것 같습니다.
    const dragEnd = [
       eUp.clientX, eUp.clientY];
        // ~~~~~
                               'Event' 형식에 'clientX' 속성이 없습니다.
           ~~~~~~ 'Event' 형식에 'clientY' 속성이 없습니다.
    console.log('dx, dy = ', [0, 1].map(i => dragEnd[i] - dragStart[i]));
  targetEl.addEventListener('mouseup', handleUp);
// ~~~~~~ 개체가 'null'인 것 같습니다.
   const div = document.getElementById('surface');
   div.addEventListener('mousedown', handleDrag);
// ~~~ 개체가 'null'인 것 같습니다.
```



에게로 보는 DOM / TS I. EventTarget

and <i>yet</i> it moves

const p = document.getElementsByTagName('p')[0];
p instanceof HTMLParagraphElement
// 참(true)

P 엘리먼트는 HTMLParagraphElement 타입이라는걸 알 수 있어요

HTMLParagraphElement는 HTMLElement의 서브타입,
HTMLElement는 Element의 서브타입,
Element는 Node의 서브타입,
node는 EventTarget의 서브타입...

	타입	예시
	EventTarget	window, XMLHttpRequest
	Node	document, Text, Comment
	Element	HTMLElement, SVGElement 포함
	HTMLElement	<i>, </i>
	HTMLButtonElement	<button></button>



이제 기 보는 DOM / TS

I. EventTarget

타입	예시
EventTarget	window, XMLHttpRequest
Node	document, Text, Comment
Element	HTMLElement, SVGElement 포함
HTMLElement	<i>, </i>
HTMLButtonElement	<button></button>

EventTarget은 DOM 타입중가장 추상화된 타입이네요 이벤트 리스너를 추가하거나, 이벤트를 보내는것만 가능하요

Event의 current Target은 Event Target | Nullo 1710 HINUIL 7 H 등성이 오류로 표시되었어요, 또 Event Target 에는 class List 속성이 없기때문에 또한 오류가 되었어요

eDown.currentTarget은 실제로 HTMLElementEF입이지만, EF입 관점에서는 window나 XMLHTTPRequest7F 될수도있어요



에게로 보는 DOM / TS 2. NODE

타입	예시
EventTarget	window, XMLHttpRequest
Node	document, Text, Comment
Element	HTMLElement, SVGElement 포함
HTMLElement	<i>, </i>
HTMLButtonElement	<button></button>

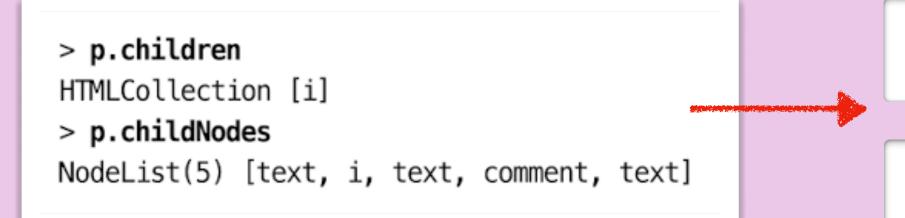
Element가 아닌 Node인 경우를 몇 가지 예로 들면, 텍스트 조각과 주석이 있어요

 And <i>yet</i> it moves <!-- quote from Galileo -->

가장바깥쪽의 엘리먼트는 HTMLParagraphElement어요 그리고 children고ト childNodes 속성을 가지고있어요

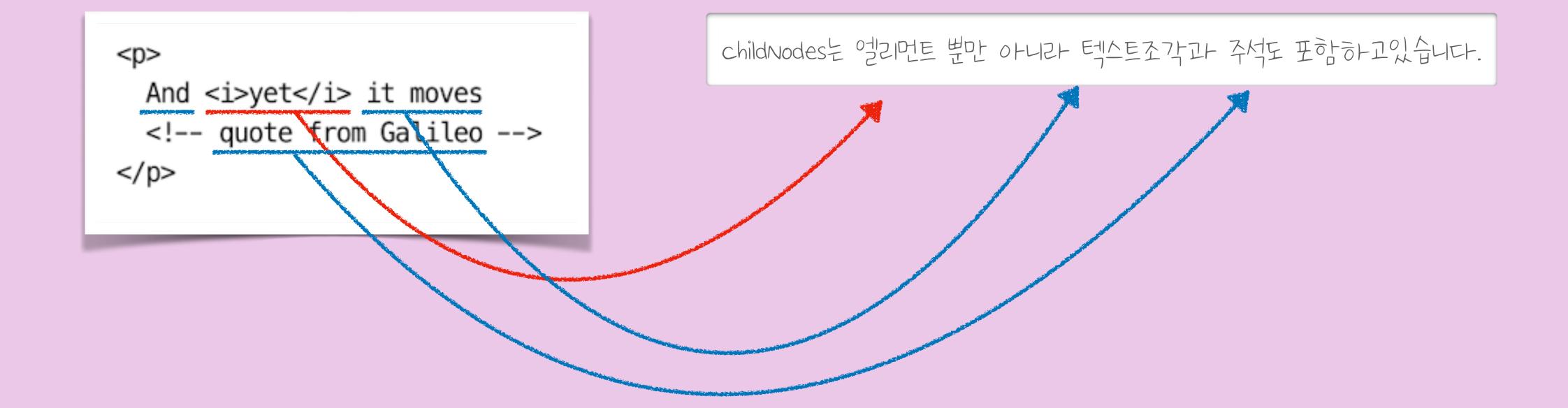
> p.children
HTMLCollection [i]
> p.childNodes
NodeList(5) [text, i, text, comment, text]





children은 자식 엘리먼트를 포함하는 배열과 유사한 구조인 HTML collection이네요.

반면 ChildNodes는 배열고나 유사한Node의 컬렉션인 NodeList에요.





에게로 보는 DOM / TS 3. (HTML)Element

타입	예시
EventTarget	window, XMLHttpRequest
Node	document, Text, Comment
Element	HTMLElement, SVGElement 포함
HTMLElement	<i>, </i>
HTMLButtonElement	<button></button>

SV역 타니의 전체 계층 구조를 포함하다면서 HTML이 아닌 엘리먼트가 존재하는는데, 바로 Element의 또 다른 종류인 SV역 Element 어요

예를 들어, (html)은 HTMLHtmlElementol고 (svg)는 SVGSvgElementoll



이기 제로 보는 DOM / TS

4. HTMLXXXXElement

타입	예시
EventTarget	window, XMLHttpRequest
Node	document, Text, Comment
Element	HTMLElement, SVGElement 포함
HTMLElement	<i>, </i>
HTMLButtonElement	<button></button>

HTMLXXXXElement 형태의 특정 엘리먼트들은 지나신만의 특별한 고유 속성을 가지고있어요

어를 들어, HTMLImageElement의 src, HTMLInputElement의 value 처럼요

이런 속성에 접근하는라면, 타입 정보 역시 실제 엘리먼트 타입이어야 하므로 상당히 구체적으로 타입을 지정하는하요



어제에로 보는 DOM / TS 4. HTMLXXXXElement

보통은 HTML 타고 값에 해당하는 button/같은 기타걸 값을 사용하며 DOM에 대한 정확한 타입 값을 얻을 수 있어요

document.getElementsByTagName('p')[0]; // HTMLParagraphElement
document.createElement('button'); // HTMLButtonElement
document.querySelector('div'); // HTMLDivElement

군데 항상 그런건 아니고, 가끔은 안됨... 특히 document.getElementById에서 그래요

document.getElementById('my-div'); // HTMLElement

document.getElementById('my-div') as HTMLDivElement;

일반적으로 타입 단언문은 지양하바하고 하는데, DOM 관련하버는 타입스크립트보다 우리가 더 정확히 알고있는 경우기이때문에 단언문을 사용하다 좋아요 저 경우에서 StrictNullchecks가 설정되었다면 null인 경우를 체크하버줘야하요

```
function handleDrag(eDown: Event) {

// ...

const dragStart = [

eDown.clientX, eDown.clientY];

// 

'Event'에 'clientX' 속성이 없습니다.

// 

'Event'에 'clientY' 속성이 없습니다.

// ...
}
```

- UIEvent: 모든 종류의 사용자 인터페이스 이벤트
- MouseEvent: 클릭처럼 마우스로부터 발생되는 이벤트
- TouchEvent: 모바일 기기의 터치 이벤트
- WheelEvent: 스크롤 휠을 돌려서 발생되는 이벤트
- KeyboardEvent: 키 누름 이벤트

EventTarget의 계층 뿐만아니라 EventEH입에도 별도의 계층 구조가 있어요 Event는 가장 추상화된 계층이에요

Clientx와 ClientY에서 발생한 오류의 원인은,
HandleDrag 함수의 매가변수는 Event로 선언되었지만
Clientx와 ClientY는 구체적인 MouseEvent여서에요

```
function addDragHandler(el: HTMLElement) {
  el.addEventListener('mousedown', eDown => { --
    const dragStart = [eDown.clientX, eDown.clientY];
    const handleUp = (eUp: MouseEvent) => {
      el.classList.remove('dragging');
      el.removeEventListener('mouseup', handleUp);
      const dragEnd = [eUp.clientX, eUp.clientY];
      console.log('dx, dy = ', [0, 1].map(i => dragEnd[i] - dragStart[i]));
   el.addEventListener('mouseup', handleUp);
 });
const div = document.getElementById('surface');
if (div) {
 addDragHandler(div);
```

그럼 어떻게 고치냐? 'Mousedown' 이벤트 핸들러를 인라인 함수로 만들면 타입스크립트는 더 많은 문맥 정보를 사용하셔 대부분의 오류를 제거함 수 있어요

그 외에도, 매가변수타입을 Even+대신 MouseEven+로 바꾸는 방법도 있어요