2일 (월)

1 .Codecademy - Building Interactive Websites - DOM Events with JavaScript

Lesson 1. DOM Events with JavaScript

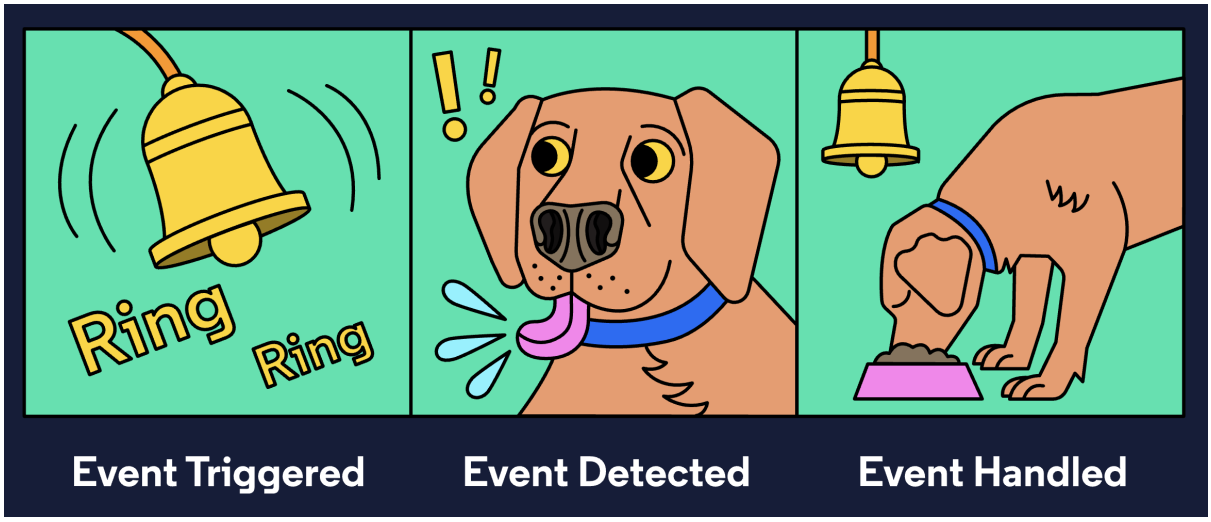
- **What is an Event ?**

마우스의 버튼 클릭, 웹페이지 파일 로딩, 유저의 이미지 옮기기 등 과 같은 것을 이벤트라고 한다.

이벤트는 웹페이지를 동적으로, 유저와 상호적이게 만들어준다.

- **"Firing" Events**

이벤트에 대한 반응은 이벤트 핸들러 함수가 DOM 요소를 수정하거나 업데이트하여 일어나게 된다. 이번 레슨에서는 이벤트 핸들러에 대해 배울 것이다.



이벤트 핸들러는 위 그림과 비유가 가능하다.

- **Event Handler Registration (이벤트 핸들러 등록)**

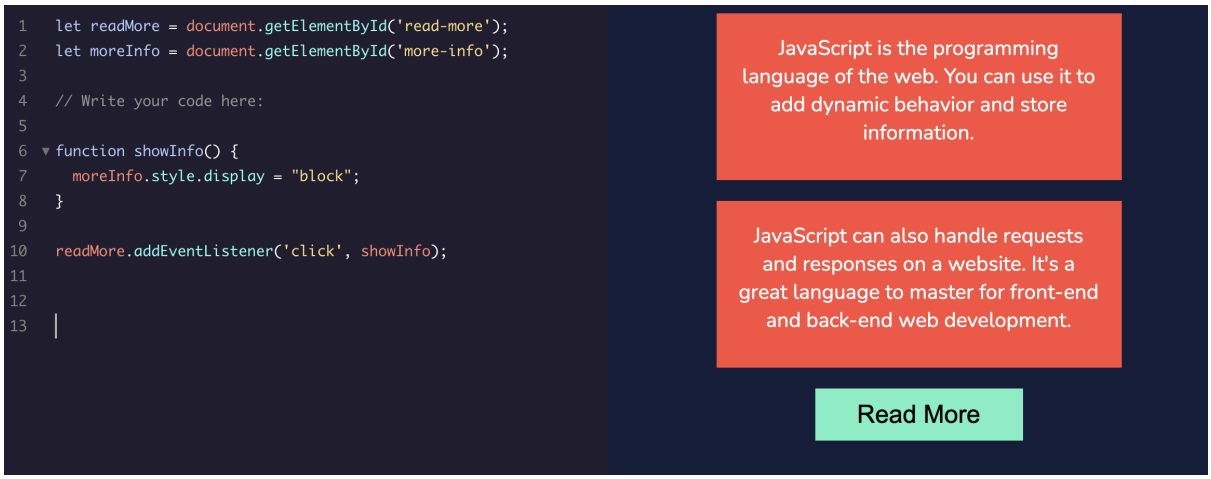
.addEventListener( ) 메소드를 이용하여 특정 DOM 요소에서 특정한 이벤트를 실행시킬 수 있다. 이벤트를 받는 DOM 요소를 event target 이라고 부른다.

```
let eventTarget = document.getElementById('targetElement');

eventTarget.addEventListener('click', function() {
  // this block of code will run when click event happens on
eventTarget element
});
```

addEventListener 메소드는 2개의 인자를 가지며, 첫번 째인자는 문자열 형식의 이벤트 이름이 들어가고, 두번째 인자는 이벤트 핸들러인 함수가 들어간다.

```
function eventHandlerFunction() {
  // this block of code will run when click event happens
}

eventTarget.addEventListener('click', eventHandlerFunction);
```

위와 같이 이벤트 핸들러 함수로써 익명함수로 입력이 가능하지만 재사용성과 가독성을 위해서 함수를 정의하고 사용하는 것이 더 좋은 습관이다.



예시

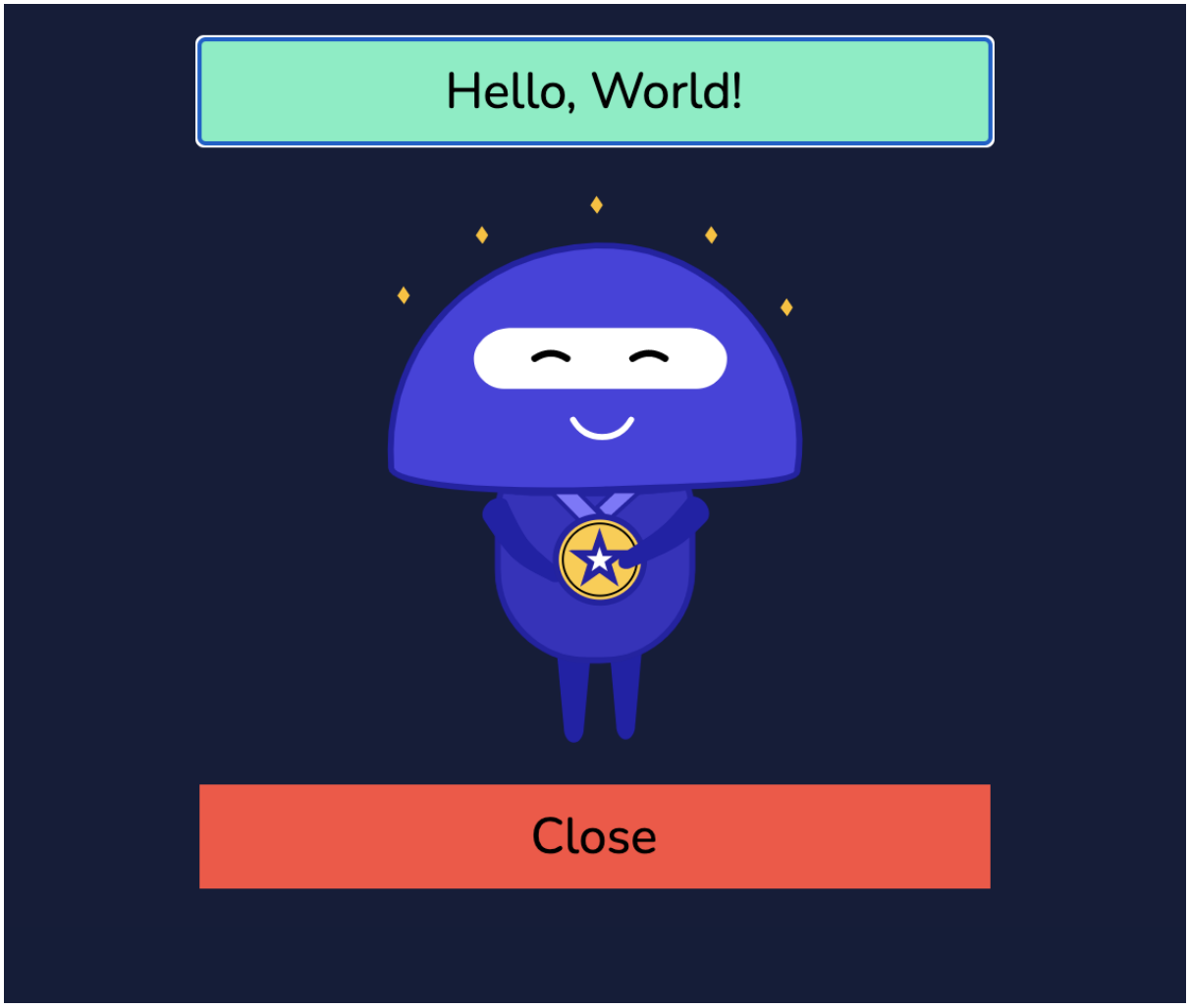
- **Adding Event handlers**

위에서 배운 것처럼 이벤트를 등록할 수 있지만 또 다른 방법이 존재한다. 바로 .onevent 프로퍼티를 이용하는 것이다.

```
eventTarget.onclick = eventHandlerFunction;
```

.onevent 는 오직 하나의 이벤트핸들러 함수만 등록이 가능하지만, addEventListner( ) 메소드는 여러개의 이벤트 핸들러 함수를 가질 수 있는 점에서 두 핸들러의 차이점을 볼 수 있다. 하지만 두개 다 널리 쓰이고 있기에 알아 두어야 한다.

```
let view = document.getElementById('view-button');
let close = document.getElementById('close-button');
let codey = document.getElementById('codey');

let open = function() {
  codey.style.display = 'block';
  close.style.display = 'block';
};

let hide = function() {
  codey.style.display = 'none';
  close.style.display = 'none';
};

view.addEventListener('click', open);
close.addEventListener('click', hide);


// Write your code here

let textChange = function() {
  view.innerHTML = "Hello, World!";
};

let textReturn = function() {
  view.innerHTML = "View";
}

view.addEventListener('click', textChange);
close.onclick = textReturn;
```



- **Removing Event Handlers**

.removeEventListener( ) 메소드는 이벤트 타겟이 더이상 이벤트를 리스닝하지 않도록 하는 메소드이다. 즉, addEventListener( ) 과 반대의 개념이다.

```
eventTarget.removeEventListener('click',
eventHandlerFunction);
```

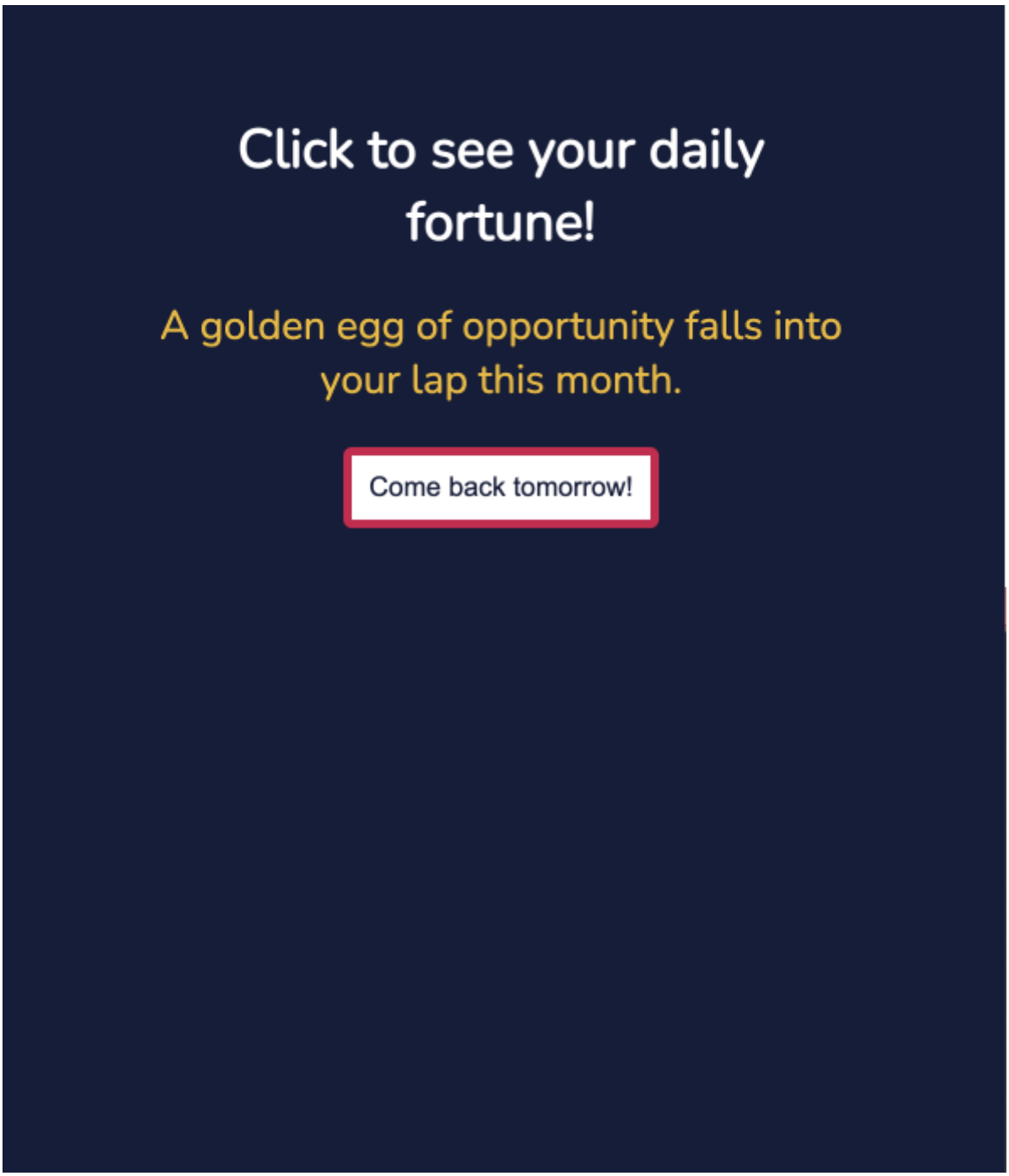
형식은 똑같다.

addEventListener 메소드와 같이 여러개의 이벤트핸들러 함수를 담을 수 있기 때문에, 익명 함수를 쓰게 된다면 실행이 되지 않을 수 있다. 그렇기에 항상 named function 을 쓰는 것이 좋다.

예시

```
let fortunes = ["A beautiful, smart, and loving person will be coming into your life.",
  "A fresh start will put you on your way.",
  "A golden egg of opportunity falls into your lap this month.",
  "A smile is your personal welcome mat.",
  "All your hard work will soon pay off."
]

let button = document.getElementById('fortuneButton');
let fortune = document.getElementById('fortune');

function fortuneSelector(){
  let randomFortune = Math.floor(Math.random() * fortunes.length);
  return fortunes[randomFortune];
}

function showFortune(){
  fortune.innerHTML = fortuneSelector();
  button.innerHTML = "Come back tomorrow!";
  button.style.cursor = "default";

  //add your code here
  button.removeEventListener('click', showFortune);
}

button.addEventListener('click', showFortune);
```

showForturn 함수 안에 removeEventListener 을 추가하였다.



button 을 누르면 계속 나오는 오류를 고친 뒤 이제 결과가 나온 후 버튼을 눌러도 눌리지 않게 되었다.

- **Event Object Properties**

자바스크립트는 이벤트들을 event object 로서 저장을 한다. 한 이벤트가 발생했을 때, 이벤트 객체는 이벤트핸들러 함수의 인자로써 전달될 수 있다.

```
function eventHandlerFunction(event){
   console.log(event.timeStamp);
}

eventTarget.addEventListener('click', eventHandlerFunction);
```

이벤트 객체내에 이미 지정된 프로퍼티들이 있다. 이벤트에 대한 정보를 볼 수 있는 프로퍼티들이다.

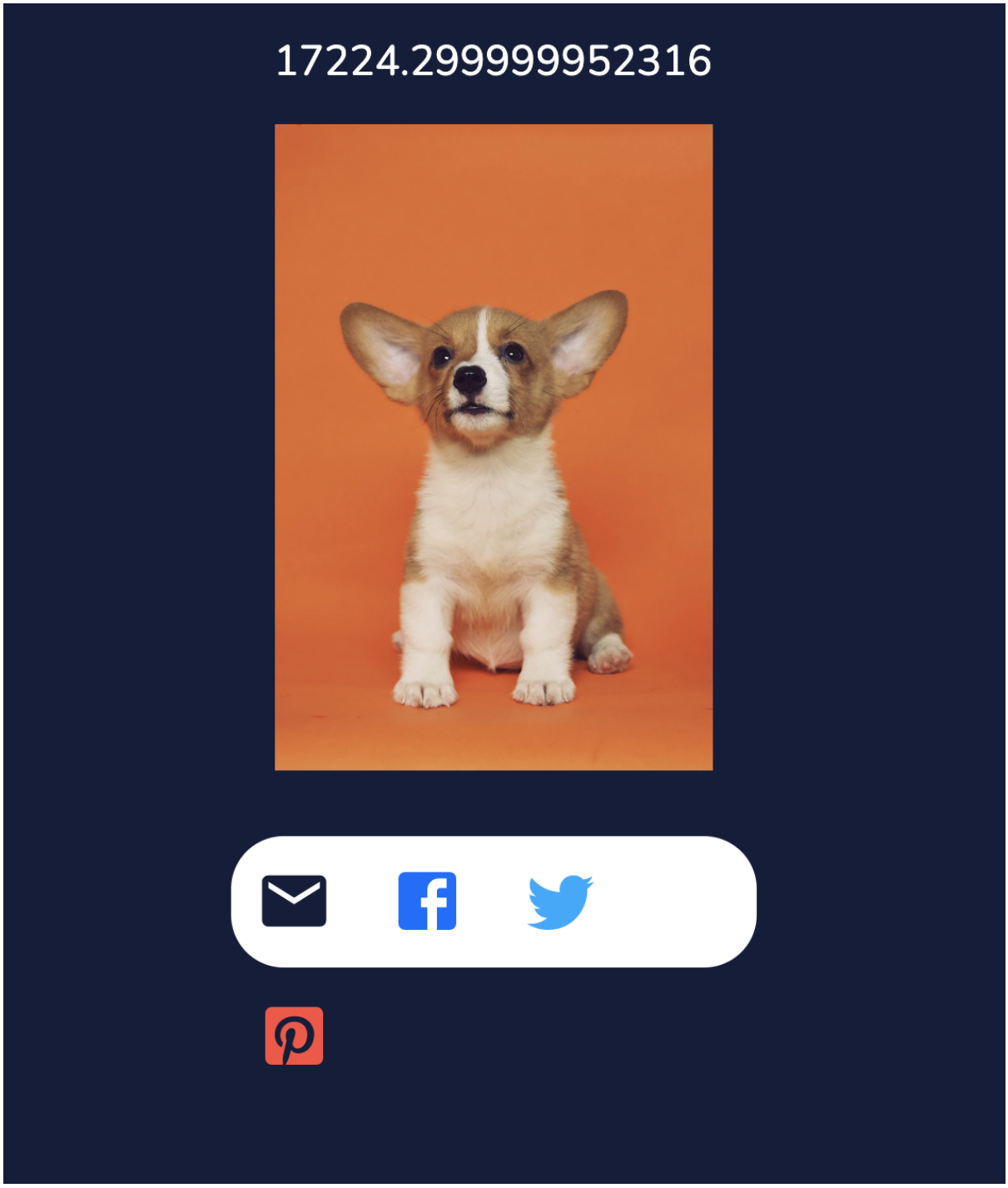
예를 들어, .target 프로퍼티는 이벤트가 기재되어 있는 태그를 참조하며, .type 프로퍼티는 이벤트의 이름에 접근하며, .timeStamp 프로퍼티는 파일이 로드되고 이벤트가 발생할 때까지의 시간을 전달한다.

Share this cute puppy photo with your friends!

Share

```
let social = document.getElementById('social-media');
let share = document.getElementById('share-button');
let text = document.getElementById('text');

// Write your code below
let sharePhoto = function(event) {
  event.target.style.display = 'none';
  text.innerHTML = event.timeStamp;
}


share.addEventListener('click', sharePhoto);


```

share 버튼을 누르면 display 가 none 으로 바뀌고, text 에 이벤트 소요시간이 나오도록 설정했다.



결과물

- **Event Types**

click 이벤트 뿐만 아니라, 많은 DOM 이벤트들이 있다. DOM 안의 대부분의 이벤트들은 그들에게 연결되어 있는 이벤트들이 없기 때문에 어떠한 알림없이 발생한다는 것을 알아두는 것은 중요하다.

또한 몇몇 기재된 이벤트들은 유저의 상호작용에 의존하지 않는다는 것을 알아두는 것 또한 중요하다. 예를 들어, load 이벤트는 웹사이트의 파일들이 완전히 로드되면 이벤트가 자동으로 실행된다.

브라우저의 많은 이벤트들은 유저 없이도 실행된다.

https://developer.mozilla.org/en-US/docs/Web/Events 여기서 많은 이벤트들을 참조할 수 있다.

그렇지만 유저의 상호작용이 필요한 마우스와 키보드 입력에 관한 이벤트들이 많다.

Random Color Generator

Find your new favorite color!

Pick a Color

Mystery Color

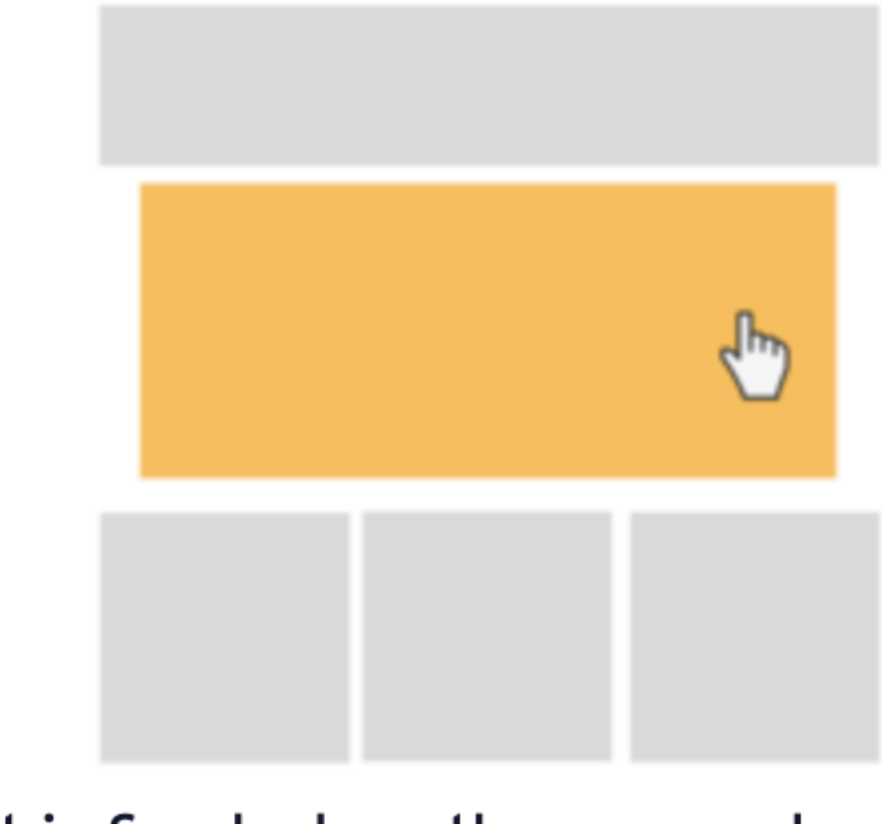
```
1   // This variable stores the "Pick a Color" button
2   let button = document.getElementById('color-button');
3
4   // This variable stores the "Mystery Color" button
5   let mysteryButton = document.getElementById('next-button');
6
7   // This random number function will create color codes for
    the randomColor variable
8   function colorValue() {
9     return Math.floor(Math.random() * 256);
10  }
11
12  function colorChange(event){
13    let randomColor = 'rgb(' + colorValue() +',' + colorValue()
    + ',' + colorValue() +')';
14    event.target.style.backgroundColor = randomColor;
15  }
16
17  button.addEventListener('click', colorChange);
18
19  mysteryButton.onwheel = colorChange;
```

onwheel 은 마우스 휠을 위 아래로 움직이면 이벤트가 발생하는 프로퍼티이다.

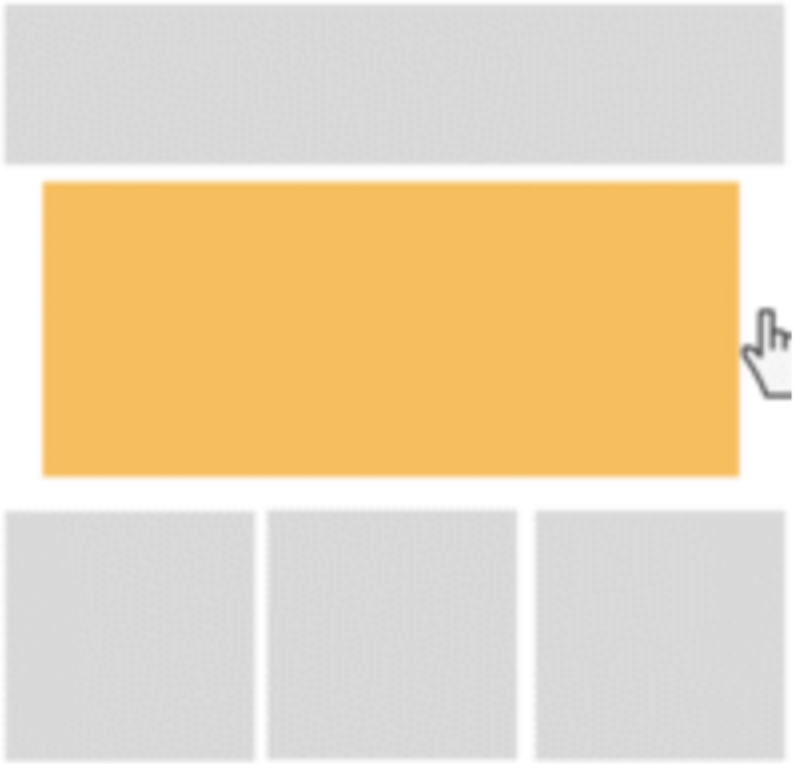
- **Mouse Events**

mousedown 이벤트 (마우스 버튼을 눌렀을 때)

mouseup 이벤트 (마우스 버튼을 눌렀다가 놓을 때)



mouseover 이벤트 (마우스를 컨텐츠에 올려놓았을 때)



mouseout 이벤트 (커서가 컨텐츠를 떠났을 때)

- **Keyboard Events**

The `keydown` event is fired while a user presses a key down.

The `keyup` event is fired while a user releases a key.

The `keypress` event is fired when a user presses a key down and releases it. This is different from using `keydown` and `keyup` events together, because those are two complete events and `keypress` is one complete event.