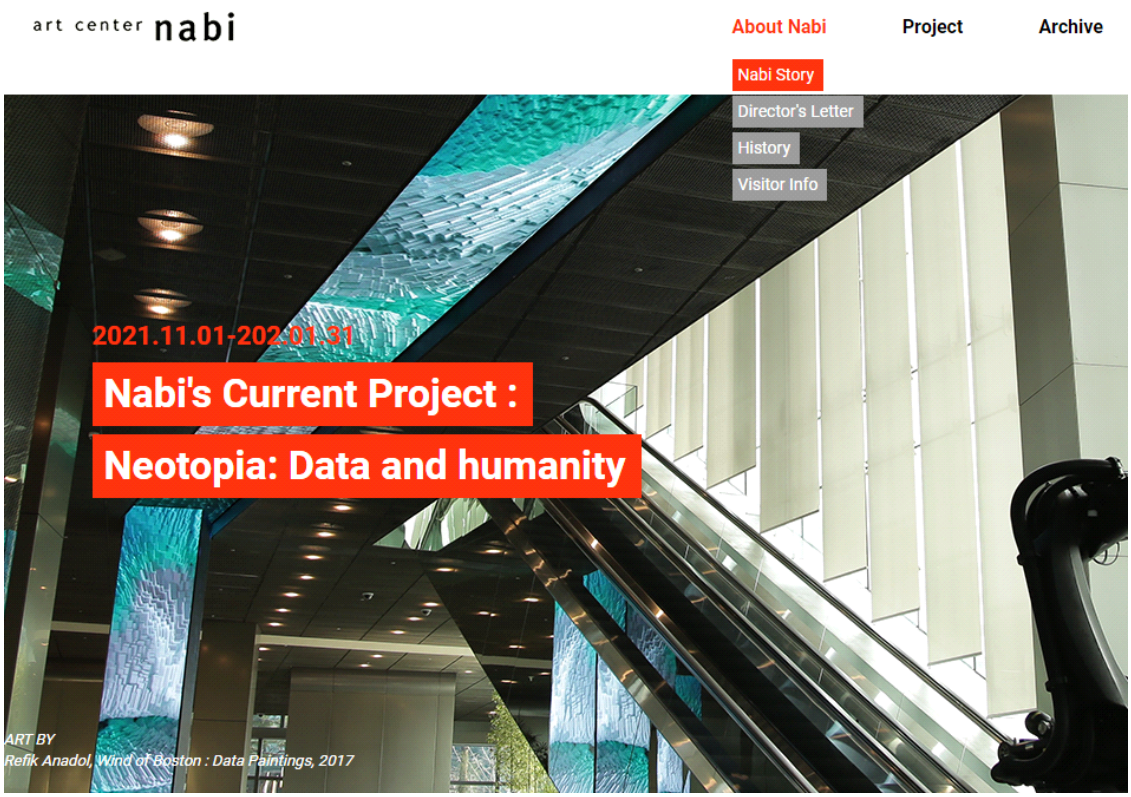
Chapter1. Desktop Type



1) JavaScript 문법

시작 파일

part5 javascript/javascript5.html ~ part5 javascript/javascript9.html

참고 URL

https://www.w3schools.com/jsref/dom_obj_document.asp

1-1) 하위 노드가 있는지 점검

hasChildNodes() 메서드는 지정된 노드에 자식 노드가 있으면 true를 반환하고 그렇지 않으면 false를 반환합니다.

```
node.hasChildNodes()

Return Value : A Boolean
```

1-2) 노드 클래스

classList 속성은 요소의 클래스 이름을 DOMTokenList 객체로 반환합니다.

```
element.classList

Return Value : A DOMTokenList
```

DOMTokenList는 DOM 속성을 구분짓는 하나의 리스트 형식입니다. 배열 모양의 객체이며 목록에 인덱스를 만들어 개별 속성을 가져올 수 있습니다.

1-3) 노드 클래스 추가

요소에 하나 이상의 클래스를 추가합니다.

```
element.classList.add(class1, class2, ...)

Return Value : No return value
```

1-4) 노드 클래스 제거

요소에 하나 이상의 클래스를 제거합니다.

```
element.classList.remove(class1, class2, ...)

Return Value : No return value
```

1-5) 노드 클래스 유무

요소에 지정된 클래스가 있는지 여부를 나타내는 부울 값을 반환합니다.

```
element.classList.contains(class)

Return Value : A Boolean
```

1-6) 노드 클래스 추가, 제거 토글

요소의 클래스를 토글 형식으로 추가 제거합니다.

```
element.classList.toggle(class, true|false)

Return Value : A Boolean
```

만일 지정된 클래스가 있다면, 지정된 클래스를 요소에서 제거하고 false를 반환합니다. 반대로 지정된 클래스가 없다면 추가하고 true를 반환합니다.

선택적인 두 번째 매개변수는 클래스가 이미 존재하는지 여부에 관계없이 클래스를 추가하거나 제거하도록 하는 부울 값입니다.

1-7) JavaScript 이벤트

참고 URL

https://www.w3schools.com/jsref/dom_obj_event.asp

1-7-1) 인라인 방식

초창기 인터넷 넷스케이프 당시부터 사용하던 이벤트 방식입니다. 전통적인 방법으로 오랫동안 사용해 와서 많은 사람들이 아직도 사용하고 있습니다.
하지만 HTML을 문서로서 최적화하기 위하여, HTML은 오로지 document를 표시해주는 역할을 담당하게 하기 위해서는 이벤트 핸들러를 연결하는 것이 좋은 방법입니다.

활용 구문입니다.

```
HTML ::

<div class="container">
    <button id="call" onclick="writeTxt()">event call</button>
    <textarea id="demo" class="note"></textarea>
</div>



JavaScript ::

function writeTxt(){
    var area=document.querySelector(".note");
    // area.textContent="Hello JavaScript!!";
    // area.innerText="Hello JavaScript!!";
    area.innerHTML="Hello JavaScript!!";
}
```

1-7-2) 속성 방식

MVC HTML 패턴(Model View Controller)에 맞게 함수를 콜백 함수 형태(Listener, 리스너)로 만듭니다.

콜백 함수는 대입 형식의 함수입니다. 이 타입의 이벤트를 지양하는 이유는 중복된 이벤트의 실행이 되지 않기 때문입니다.

일반적인 함수 형식입니다.

```
function myFn(){
    // console.log("function call!!");
}
myFn();
```

대입 형식의 함수 형식입니다.

```
var myFn=function(){
    // console.log("callback function call!!");
};
myFn();
```

이벤트의 속성 방식입니다.

```
window.onload=function(){
    // console.log("load event!!");

    var button=document.getElementById("call");

    button.onclick=function(){
        // console.log("Hello JavaScript!!");
    };
    button.onclick=function(){
        // console.log("Hello jQuery!!");
    };
};
```

1-7-3) addEventListener() 방식

위에서 언급한 콜백 함수 방식의 오류로 인해서, 일반적인 JavaScript에서의 이벤트는 대부분 addEventListener(), attachement() 방식을 사용합니다.

특히 load 이벤트의 중복 호출이 문제가 될 때가 많이 있습니다. 하지만 Internet Explorer 8.0과 Opera 6.0 브라우저에서는 지원되지 않으므로 이 경우에는 attachEvent() 이벤트 방식으로 작성합니다.

addEventListener() 메서드는 지정된 요소에 이벤트 핸들러를 연결합니다.

```
element.addEventListener(event, function, useCapture)

Return Value : No return value
```

활용 구문입니다.

```
window.addEventListener("load", function(){
    // console.log("load event!!");

    var button=document.getElementById("call");

    button.addEventListener("click", function(){
        // console.log("Hello JavaScript!!");
    });
    button.addEventListener("click", function(){
        // console.log("Hello jQuery!!");
    });
});
```

1-7-4) attachEvent() 방식

addEventListener() 이벤트 방식은 일반 브라우저에서 적용하며, Internet Explorer 8.0과 Opera 6.0 브라우저에서는 attachEvent() 방식으로 작성합니다.

IE 8.0 브라우저를 위한 조건부 주석으로 작성할 필요가 있습니다.

```
element.getAttribute(attributename)

Return Value : A String
```

활용 구문입니다.

```
// console.log(window.addEventListener);

if(window.addEventListener){
    window.addEventListener("load", function(){
        var button=document.getElementById("call");
        var note=document.getElementById("demo");

        button.addEventListener("click", function(){
            note.innerHTML="Hello JavaScript!!";
        });
    });
}
else{
    window.attachEvent("onload", function(){
        var button=document.getElementById("call");
        var note=document.getElementById("demo");

        button.attachEvent("onclick", function(){
            note.innerHTML="Hello JavaScript!!";
        });
    });
}
```