

HTML → 정적임. (만들면 끝. 바꾸는거.)

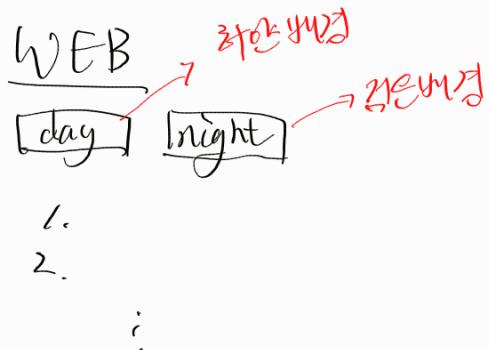
JavaScript ⇒ 사용자와 상호작용 할 수 있게 해줌.

(Ex) 배경색 키자가 선택 등.

태그로 넣어가보자.
蕴傳

* day - night button ⇒ 상호작용 예시. 이런식이다~
만들고 넘어가.

(Ex)



유저의 "day" / "night" 버튼
⇒ 마우스 클릭 (상호작용) 으로
페이지 배경색 및 툴트식 변경

<body>

<h1>...</h1>

<input type="button" value="day" onclick = "

```
document.querySelector('body').style.backgroundColor = 'black';
document.querySelector('body').style.color = 'white';
">
```

<input type="button" value="night" onclick = "

```
document.querySelector('body').style.backgroundColor = 'white';
document.querySelector('body').style.color = 'black';
">
```

• 그림, HTML과 별개의 독립체계를 가진 JavaScript를 이렇게

HTML에서 이어서 처리하는 방식인?

⇒ <script> ~ </script>

② JAVA SCRIPT

$1+1=2$
~~
HTML

$1+1=1+1$
~~

HTML처럼 단순 입력값

⇒ 흐름이 아님.

JS는 동적으로 작동하여

이처럼 내부에서 계산값 출력

```
<body>  
  <h1>JAVA SCRIPT</h1>
```

```
  <script>  
    document.write("1+1=", 1+1);  
  </script>
```

```
  <h1>HTML</h1>
```

$1+1=1+1$

```
</body>
```

만약 write(1+1);

이면 2만 출력.

* 웹 브라우저에서 발생하는 “이벤트”를 받아서 처리하는 방법
(page)

⇒ <input> 태그 속성 !! (on~ 속성.)

↳ 웹상호작용의 기획

• 버튼 생성 → <input> 태그에서 type=button 지정, 클릭 이벤트
= onclick

```
<input type="button" value="k" onclick="alert('SJ')">
```

⇒ 버튼 클릭 시 알림메세지로 SJ 출력.

ex)

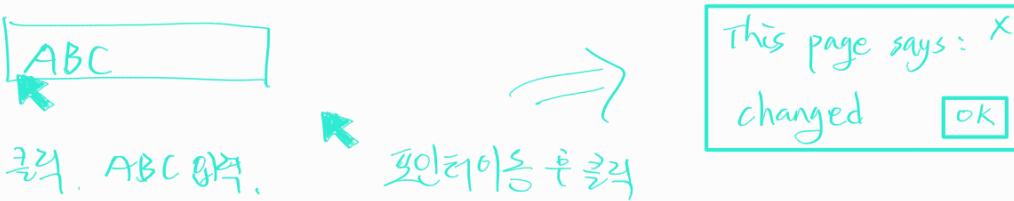


- 텍스트 입력 칸 생성) → <input> 태그 type = text 지정. 입력값 변경 이벤트: onchange

```
<input type="text" onchange="alert('changed')">
```

⇒ 텍스트 입력창 클릭 & 입력값(글자) 변경 후 마우스로 이동을
입력창 바깥으로 빼면 알림메세지 changed 출력.

ex)



- 텍스트 입력 시 알림 끄기하고 싶다면 → onkeydown 특성
(기 입력 이벤트)

```
<input type="text" onkeydown="alert('KEY DOWN')">
```

⇒ 텍스트 입력창 클릭 & 입력 시 키 하나 입력될 때마다
KEY DOWN 출력.

ex)



A, B, C 각 문자 입력할 때마다

ABC



This page says: X

KEY DOWN

OK

크리. ABC 알약.

* Console과 JavaScript의 관계

- 웹페이지에서 F12로 개발자 도구 열면 "console" 터미널 존재
- 일반 프로그래밍 언어 console 창처럼, 이것도 JavaScript 편집
- 현재 페이지를 대상으로 입력하는 코드 수행 !!

(ex)

```

<div class="li" id="u_0_s" style="display: flex; align-items: center; justify-content: space-between; gap: 10px;">
    <div>
        <script type="text/javascript"></script>
        <script></script>
        <script></script>
        <script></script>
        <!-- BigPipe construction and first response -->
        <script></script>
        <script>bigpipe.beforePageletArrive("first_response")</script>
        <script></script>
        <script></script>
        <div class="hidden_elem"></div>
        <script>bigpipe.beforePageletArrive("pagelet_group_permalink")</script>
    </div>

```

→ 랜덤 추출 (4명)

→ 이런식으로 해당 페이지 내 정보 추출 & 분석 가능.

* JS의 Data type - 문자열 & 숫자

• Boolean

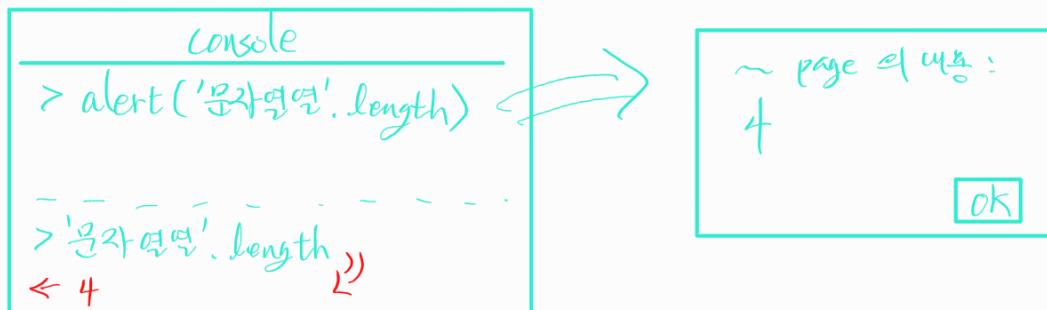
- Null
- Undefined
- Number
- String
- Symbol
- + Object

* Console 창에서 숫자 & 문자 입력 테스트.

- 어떤 문자의 글자수 추출하는 경우

- console 창에서 alert 으로 (원래는 텁) '문자'.length 입력.

ex)



바로처도 콘솔창에 나옴!

- 특정 문자의 첫 등장 위치

- '문자열'.indexOf ('찾을문자')

ex)



```

> 'Hello World'.indexOf('o')
← 4
> 'Hello World'.indexOf('x') ~ 찾는값 -> return
← -1
> 'Hello World'.indexOf('orld') ~ 문자열은 찾을 수 있다.
← 7

```

• 공백 제거기

- '문자열' .trim()

②)

console
> ' space here '.trim() ← 'space here'

• 문자열 vs 숫자

- "1" + "1" 와 1 + 1,

③)

console	
> 1+1 ← 2	
> "12" + "12" ← "1212"	~ 이어붙여짐

* 변수와 대입연산자.

\sim \sim

$x =$

• ES에서 변수를 선언하는 3가지 방법

언어를 선언하는 3가지 방법.

- ✓ const → 변하지 않을 변수, (상수화)
- ✓ var → * 호이스팅을 하는 변수선언.
- ✓ let → 블레이드된 선언. 예쓰면 일반 언어와
동일하게 작동.
- ✗ 선언x. (그냥 $x=3$ 해도 선언됨.)

* 호이스팅 ?

Hoisting

→ interpreter가 변수와 함수의 미로리 공간을
선언전에 미리 할당하는 것.

⇒ 문제점 1.

```
> console.log(a)  
var a=3  
< undefined
```

· a를 선언도 전에 사용했는데 'Error' 가 아닌

'undefined' 출력

⇒ 문제점 2.

```
> for(var i=1; i<5; i++) {  
    console.log(i)  
}  
  
console.log(i)  
  
< 12345
```

- 지역변수 개념이 없어, for 뒤에서 바로 출력.
입력하는, 함수 내에서 선언해야 지역변수로
인정. 나머지는 전부 전역변수 취급.

⇒ 문제점 3.

```
> var a=1  
    console.log(a)  
  
var a=3  
    console.log(a)  
  
< 1 3
```

- 줄복 선언이 가능 함. (자신 이전에 등장x..)

⇒ 결론: let 오로 선언하자.

* 웹 브라우저 제어 (day-night 처리)

- CSS 기초. style 속성.
- 디자인을 변경하고자 하는 태그에 style 속성 사용

ex) <h2 style="color: powderblue"> Java-CSS-Test </h2>
→ Java-CSS-Test ↓
색변경!

- div 태그.
- 일반적이. 가능도 X. CSS3 문법이 데다가 속성부여하고 싶을 때 사용.
- 하면 전체 사용. 줄바꿈이 default!

- Span 태그.
- div 태그와 마찬가지로 의미, 가능 X.
- 줄바꿈 없음 default!

선택자라고 함.
※※※

적용 대상 선택!

- ※
- style 태그.
 - 지금까지 본 디자인 관련 태그, 속성을 class, id처럼 써어서 통합 관리 ⇒ 본문에서 일일히 속성 지정할 필요 X !!

* Style 태그 - class 속성

- class 를 속성 변경 시 해당되는 파트 전부 변경

ex)

```

<head>
  <style>
    .js { font-weight: bold; color: red; }
  </style>
</head>
<body>
  <p>
    <span class="js">JavaScript </span> JS,
    CSS, HTML, what we learned before.
    Thanks <span class="js">JS </span> !!
  </p>
</body>

```

• 붙여야 class 를 의미함.
 # 붙이면 id 의미.

⇒ **JavaScript JS, CSS, HTML, what we learned before.**

Thanks JS !!

* Style 태그 - id 속성

- id는 분반 (Class) 내의 학번(id) 같은 개념.
- Class는 해당되는 모든 class 속성 태그를 전부 변경.
- id는 고유한 번호 개념으로, class 와 id 가 있다면 id의 속성이 우선으로 적용됨.

ex)

```

<head>
  <style>

```

```

#: id 속성부여 ←
  .js { font-weight: bold; color: red; }
  #first { color: orange; }

```

</style>

</head>

<body>

<p>

JavaScript JS,
 CSS, HTML, what we learned before.
 Thanks JS !!

</p>

</body>

⇒ JavaScript JS, CSS, HTML, what we learned before.

L! Thanks JS !!

여기서,

class의 "bold"에는 위치됨.

즉, class의 속성을 입력하고, 그 위에 id를 덮은 것 !!

* Style 태그 - 태그명 (ex. h1, h2, span...)

- class, id처럼 태그명 자체를 그늘침해서 속성 부여 가능.
- 우선순위는 떨어짐. style 태그 위에 class, id 속성이 존재할 경우 하위 순위로 밀림. (아무런 적용 X)

ex)

```

<head>
  <style>

```

만약 class 속성이
주석 처리되지 않았다면
class 가 발견되고
span은 물음표!!

```
<!--.js { font-weight: bold; color: red; } -->
span { color: pink; }
</style>
</head>
<body>
  <p>
    <span class="js">JavaScript </span> JS,
    CSS, HTML, what we learned before.
    Thanks <span class="js">JS </span>!!
  </p>
</body>
```

span 이전에 "js" 클래스.

⇒ JavaScript JS, CSS, HTML, what we learned before. Thanks JS !!

* JavaScript : 제어할 태그 선택

유저와 동적으로 상호작용

- Day, Night 버튼처럼 소파일 적용여부 설정 펼치기

ex)

```
<body>
  <h1>...</h1>
  <input type="button" value="day" onclick =
    document.querySelector('body').style.backgroundColor = 'black',
    document.querySelector('body').style.color = 'white',
  ">
  <input type="button" value="night" onclick =
    document.querySelector('body').style.backgroundColor = 'white',
    document.querySelector('body').style.color = 'black',
  ">
```

이부분 2중 body로 처리 가능.
(ex. document.body.style ~)

상황에 알맞게 속성 몰라 써야 함.

body 아닌 특정 클래스, Id 이용 시
직접 사용!

ex) document.querySelector('.sj-class').~
document.querySelector('#div-sj').~

* 프로그램 (Program) 이란?

- "순서"에 따라 실행되는 작업.
- 프로그래머는 이 작업을 만드는 사람.
- HTML : 작업 내용을 보여주는 언어. → 프로그래밍 언어 X
순서 필요 X. 입력된거 다 봐보아.
- JavaScript : 유저와 상호작용을 위해 시간 순서에 따라
입력 작동해야 함. → 프로그래밍 언어 O.

* 비교 연산자

Equal : ===

Less : < → "<" 문자 입력은 <

Greater : > >= <=

* if (조건문)

- if (true) 처럼 작동. (++ 사용, (괄호 필요))

* 리팩토링.

- 코드 간결화를 위한 단순화 작업.
- 변수 지정 : 동일 항목을 여러 번 쓸 경우
변수로 지정하여 대체.
- this 함수 : 만약 어떤 태그 (<input>)의
id를 기준으로 자신의 속성을 변경할
경우 this로 대체 가능.

&

<input type="button" value="Dark" onclick="

let body_v = document.body;

if (this.value === 'Light') {

this.value = 'Dark';

body_v.color = 'white';

```
body.v.backgroundColor = 'black';  
{"<"}
```

* 배열.

- 대괄호로 표현. 변수로 지정 가능.

Ex)

```
<body>  
  <h2> Syntax </h2>  
  <script>  
    let arr = ["apple", "orange"]  
  </script>  
  <h2> output </h2>  
  <script>  
    document.write(arr[0]);  
    document.write(arr[1]);  
    arr.push('kiwi');  
  </script>  
  <script>  
    document.write(arr.length);  
  </script>
```

out

: appleorange ↪ { document.write(arr[0]);
 document.write(arr[1]);

 친도 추가. ↪ arr.push('kiwi');

 </script>

 <script>

out
: 3

 ↪ document.write(arr.length);
 </script>

- Array 함수 매우 잘 구성되어 있음!

push, pop, concat, find, fill, indexOf

* 반복문

- for loop : python 처럼 in 사용.

Ex) let arr = ['a', 'b', 'c', 'd']
for(let i in arr) {
 document.write(arr[i]);
}

← abcd

- while loop : while() 조건 조건 부여.

Ex) let i = 0
while(i < arr.length) {
 document.write('' + arr[i] + '');
}

← • a
• b
• c
• d

* 합수

• function (function name) () { } (function body)

Function 험수 (인자) {} 연체도 사용

- 인자에 변수(함수) 지정 X (≈ python)

* 객체 (object)

- Variable에 dict처럼 지정!! 특이함.
- 타입변수에 Function을 지정 가능!

(x)

<script>

```
let user = {  
    "id": "sj",  
    "name": "sjlee"  
};  
showAll: function() {  
    for (let k in user) {  
        if (k == 'showAll') continue;  
        document.write(k + ': ' +  
            user[k] + <br>);  
    }  
};  
};  
// define and
```

user라는 변수에
— (객체)
 모든 속성 (property)

지정

임의

```
> document.write("id : " + user.id + "<br>");  
document.write("name : " + user.name + "<br>");
```

{ User.age = "30";]

 user["phone number"] = "010-11";

 document.write("PH : " + user["phone number"
 + "
"]);

 document.write('-----
');

function ← user.showAll();

call

← id : sj

name : sjlee

PH : 010-11

id : sj

name : sjlee

age : 30

phone number : 010-11

function

call 결과.

* js 파일로 속성 배포

- 위에서 본 함수, 객체 등의 코딩 내용을

, js 파일로 처리해서 넘기도록 하면

- <script> 안의 내용을 볼 수 있음. (script 포함)

(x)

~~<script>~~

```
function sum(first, second) {  
    return (first + second);  
}
```

```
function multiply(first, second) {  
    return (first * second);  
}
```

~~</script>~~



<head>

```
<script src = "calc.js"></script>  
</head>
```

src (source) 속성으로 호출.

<body>

<script>

→ custom module

import 가능 있음.

```
    sum(1,5);
```

```
    document.write(<br>);
```

```
    multiply(1,5);
```

</script>

import 가능
할 때 function
사용 가능!

</body>

← 12
35

* 라이브러리 / 프레임워크

- library : 만들어진 기능을 뺏겨와서 사용.
- framework : 만들고자 하는 방향의 공통부분.
즉, 이 위에서 나만의 방향을
찾아가는 느낌.

library

```
<head>
  <script src = "library">
</head>
{
+
  +
```

framework

```
  <head>
    style
    <body>
    +
```

+ editing

내 코드

내 코드

- 라이브러리

- jQuery: JS 대포라이브러리. DOM을 쉽게 다루도록 도와주는 기능 아주 많음.

- Lodash

- > jQuery 이후 대포 라이브러리

- Moment

- 아웃로드 방법

- 1) 직접 아웃로드

- 아웃로드 후 내 풀더에 옮기기.

- 2) CDN (Content Delivery Network)

- 제작자 서버에 저장한걸

- 링크 형식으로 코드에서 쓸 수 있음 !!

↳ jQuery CDN 사용.

- jQuery 사이트에서 Google CDN → 최신 snippet 링크

복사해서 <head>에 붙여넣기.

→ 그 다음 함수 (function (clr) { 여기 }) 끝에서

$\$('a')$, $.css('color', clr)$ 처럼 사용. 끝! 간편!

jQuery의 선택자.

선택할 객체명

접근하는 식별자.

body, a, h1 또는 id 등 입력.

jQuery를 의미하는 함수.

※ $\$('a') = \$(\text{jQuery}('a'))$ \$가 그 자체로 함수임.

→ 이 line의 의미 : 페이지에 'a' 태그 'color'를
모두 clr로 지정한다.

> <head>

import jQuery ← <script src="https://code.jquery.com/jquery-1.12.4.min.js">
</script>

<script>
\$(document).ready(function () {
 \$('#doc_id').text("문서 로딩완료");
});

\$(window).load(function () {
 \$('#win_id').text("창 로딩완료");
});

</script>

</head>

<body>

<p id="doc_id">제이쿼리</p>

<p id="win_id">자바스크립트</p>

</body>

* UI vs API

- UI (User Interface)

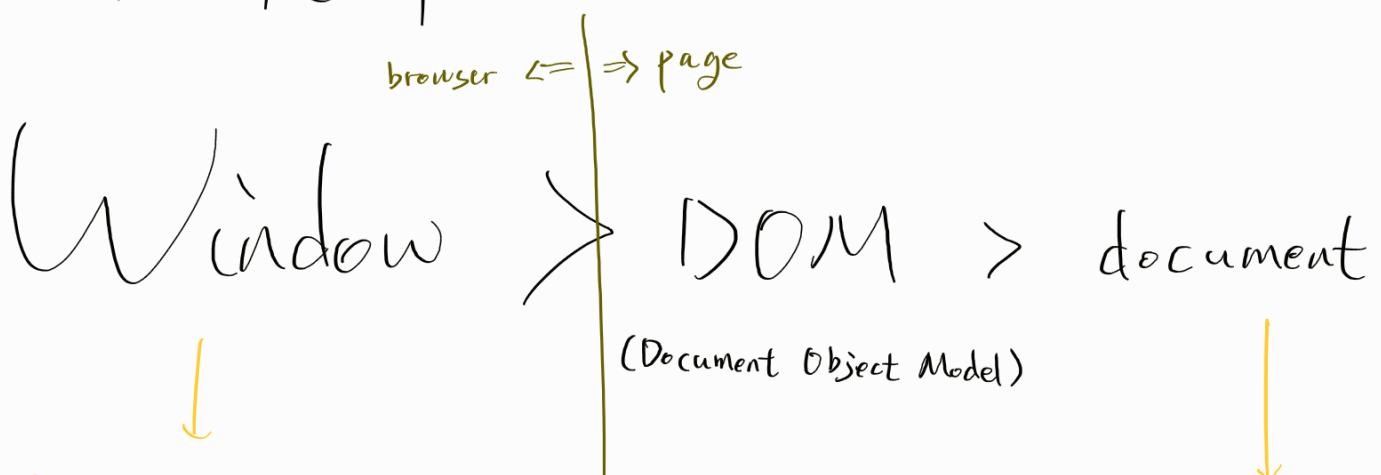
- 버튼 (`button`) 처럼 페이지를 사용하는 사용자가 시스템을 제어하고 쓰는 장치.

- API (Application Programming Interface)

- JS의 alert (`onclick="alert('...')"`) 처럼 코드를 하는 프로그래머가 웹브라우저가 갖고 있는 기능(경고창, 링크접속 등)을 사용하기 위해 쓰는 장치. (액세스 방법)

→ 입력 순서에 따라 작동. API는 순서대로 실행.
매우 친한 단어

* 용어 상하관계



웹 브라우저 (웹페이지)

태그 삭제

자체를 다루어야 할 경우

자식태그 추가 등

(page 편집, 크기, 색 창 열기 등)

태그 관련 조작

* 그 외 검색 기획

reload

- ajax → 웹페이지 리로드 없이 정보 변경
페이지 개별에 액션 적용!!
- cookie → 반대로
웹페이지 리로드 후에도 미리 상태 유지 원한다면.
선택보기 + 사용자 별 서비스 제공 가능!
- offline web application → 인터넷 끊겨도 볼 수 있는 웹
원한다면.
- webrtc → 화상통신 웹 개발 학습으면.
- speech → 음성 전송 원한다면.
- webgl → 3D 그래픽, 게임 개발 관련.
- webvr → 가상현실 관련.

