# 데이터 조작을 위한 자바스크립트

## 빅데이터 오케스트레이션 및 시각화 실습 – 2일차

# 학습 내용

1. 배열과 JSON
2. Callback 함수
3. HTML 기본
4. UI와 이벤트
5. DOM
6. Ajax 통신

# _배열 - 기본

- **생성 하기**

. 배열 리터럴 대괄호([ ])를 사용하여 만드는 방법

. Array() 생성자 함수로 배열을 생성하는 방법

- **요소 다루기** : for-in, for-of 반복문 이용 *** forEach 콜백함수 사용**

```
var arr = ['zero', 'one', 'tow'];
var arr2 = new Array();
arr2.push('zero');
arr2.push('one');
arr2.push('two');

for (let i in arr) {
   console.log(arr[i]);
}

for (let i of arr) {
   console.log(i);
}

arr.forEach((item, index) => {
   console.log(item, index);
});
```

# _배열 - 주요 함수

## - 자신을 변화 시키는 함수들

| 메소드 | 설명 |
|---|---|
| pop()* | 배열의 마지막 요소를 제거하고 리턴합니다. |
| push()* | 배열의 마지막 부분에 새로운 요소를 추가합니다. |
| reverse()* | 배열의 요소 순서를 뒤집습니다. |
| slice() | 배열 요소의 지정한 부분을 리턴합니다. |
| sort()* | 배열의 요소를 정렬합니다. |
| splice()* | 배열 요소의 지정한 부분을 삭제하고 삭제한 요소를 리턴합니다. |

* 표시된 메소드는 자기 자신을 변화시킵니다.

**- 기존의 배열 요소를 제거하거나 새로운 배열 요소를 추가하여 배열의 내용을 변경 : splice()**

→ splice([제거할 요소 시작 인덱스], [제거할 개수], [삽입될 요소들 …] )
배열에서 제거된 요소를 배열의 형태로 반환

arr.**splice**(1, 2, false, "C언어") → 인덱스 1의 요소부터 2개의 요소를 제거한 후, false와 "C언어"를 그 자리에 삽입함.
var removedElement = arr.**splice**(1, 2, false, "C언어"); → 제거된 요소 리턴

# _배열 - 주요 함수

```
let array = [{name: "고구마", price: 1000}, {name: "감자", price: 500},
             {name: "바나나", price: 1500}];

let popped = array.pop();
console.log(popped);
console.log(array);

array.push(popped);
array.push({name: "사과",price: 2000});
console.log(array);

delete array[2];
console.log(array);

var arr = [1, true, "JavaScript", "자바스크립트"];
arr.reverse();
console.log(arr);

var removedElement = arr.splice(1, 2, false, "C언어");
console.log(arr);
console.log(removedElement);
```

# _배열 - 정렬 함수

- sort([정렬함수-콜백])
  . 기본은 오름차순. 내림차순으로 하려면 sort() 후 reverse() 해 준다.
  . 콜백함수를 정의하여 정렬 기준과 방법을 정할 수 있다.

```
let arrayA = ["고구마", "감자", "바나나"];
arrayA.sort();  // 기본 오름차순

let arrayB = [{name: "고구마", price: 1000}, {name: "감자", price: 500},
              {name: "바나나", price: 1500}];

arrayB.sort((itemA, itemB) => {
        if (itemA.name < itemB.name) {
          return -1; // 작은 것 판정시 -1 리턴
        } else if (itemA.name > itemB.name) {
          return 1;  // 큰 거 판정시 1 리턴
        } else {
          return 0;
        }
});
```

# _배열 - 주요 함수

- 자신을 참조만 하는 함수들

. **join**([구분자-없으면','로 구분]) : 배열의 모든 요소를 하나의 문자열로 반환합니다.

. **slice**([시작 인덱스], [종료 인덱스]) : 시작 인덱스부터 종료 인덱스 바로 앞까지의 배열 요소를 추출.
인수로 종료 인덱스가 전달되지 않으면 마지막 배열 요소까지 모두 추출

. **concat**([배열1], [배열2]) : 인수로 전달받은 배열들을 합쳐서 만든 새로운 배열을 반환합니다.

. **toString**() : 배열을 문자열로 반환. 요소간에는 쉼표로 구분

```
var arr = [1, true, "JavaScript", "자바스크립트"];
console.log(arr.join());
console.log(arr.join(" + "));

console.log(arr.slice(1, 3));
console.log(arr.slice(1));

console.log(arr.concat([2], [3, 4]));
console.log(arr.concat("다섯", [6, 7]));

console.log(arr.toString());
```

# _JSON 객체

- JSON 는 Douglas Crockford가 널리 퍼뜨린 Javascript 객체 문법을 따르는 문자 기반의 데이터 포맷
  JSON = Javascript Object Notation
- 개별 JSON 객체를 .json 확장자를 가진 단순 텍스트 파일에 저장할 수 있음. MIME 타입은 application/json

- JSON 객체 : 객체를 사용하는 데이터 표현방법
  . 문자열은 큰 따옴표로 만들어야 한다
  . 모든 키는 큰 따옴표로 감싸야 한다
  . 값에는 숫자, 문자열, 불 자료형, 배열, JSON 객체를 사용한다

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
}
```

# _JSON 데이터 (Collection 다루기)

- 내부 요소 접근
  . 배열형식 접근 : 객체명['요소키']
  . 객체지향형 접근 : 객체명.요소키

- 데이터 다루기
  . 값 사용 : let name = SuperHero.hometown;
  . 값 할당 : SuperHero.hometown = "Seoul";
  . 요소 삭제 : delete SuperHero.hometown;
  . 요소 확인 : in 연산자

SuperHero

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    }
  ]
}
```

SuperHero.hometown
SuperHero['hometown']

SuperHero.active
SuperHero['active']

SuperHero.members[0].age
SuperHero['members'][0]['age']

SuperHero.members[0].powers[1]
SuperHero['members'][0]['powers '][1]

# _JSON 데이터 (Collection 다루기)

- 요소 순회

```
for (key in obj) {
    console.log(obj[key]);
}
```

* if 문에서는 요소를 확인하는 연산자로 쓰임

```
const javascriptObject = [{name: '윤인성', region: '서울'}, {name: '윤명월', region: '도쿄'}];

for (let key in javascriptObject [0]) {
    console.log(typeof javascriptObject [0][key]);
    console.log(javascriptObject[0][key]);
}
```

# _Array, List, Map - 심화

- 리스트 구조와 Map 구조를 복합적으로 구성해서 쓰는 경우가 많음

**1) 자바스크립트 객체배열 : 배열 요소로 객체를 사용**

. 요소 접근 : 배열[인덱스].요소명

**2) 요소에 배열 사용 : 객체요소의 값으로 배열을 사용**

. 요소 접근 : 객체.요소명[인덱스]

SuperHero

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    }
  ]
}
```

SuperHero.hometown
SuperHero['hometown']

SuperHero.active
SuperHero['active']

SuperHero.members[0].age
SuperHero['members'][0]['age']

SuperHero.members[0].powers[1]
SuperHero['members'][0]['powers '][1]

# _JSON 객체와 자바스크립트 객체간 변환

- JSON 객체 → 자바스크립트 객체 : JSON.parse([JSON 객체])
- 자바스크립트 객체 → JSON 객체 : JSON.stringify([객체],null,[공백갯수])

```
const javascriptObject = [{name: '윤인성', region: '서울'}, {name: '윤명월', region: '도쿄'}];
const outputA = JSON.stringify(javascriptObject);
const outputB = JSON.stringify(javascriptObject, null, 2);
console.log(outputA);
console.log(outputB);

const outputC = JSON.parse(outputB)
console.log(outputC);
```

# _변수의 유효 범위 (variable scope)

변수의 유효 범위(scope)란 해당 변수가 접근할 수 있는 변수, 객체 그리고 함수의 집합을 의미

- scope는 블록 단위로 : 외부 변수 사용 가능, 내부 변수는 외부에서 사용 안 됨
- 지역 변수(local variable) : 함수 내에서 선언된 변수
- 전역 변수(global variable) : 함수의 외부에서 선언된 변수를 가리킵니다. (페이지 닫힐 때까지)

```
let num = 10;
function globalNum() {
  console.log("변수 num의 값은 " + num + "입니다.");
  num = 20;
  let num2 = 100;
}
globalNum();
console.log("변수 num의 값은 " + num + "입니다.");
console.log("변수 num2의 값은 " + num2 + "입니다.");
```

```
▶Uncaught ReferenceError: num2 is not defined
    at ex1.html:20:30
```

# _타이머 - 반복 실행, 시차를 두고 실행

- 반복 실행 : setInterval([함수], [지연시간 : 밀리세컨], <옵션 : 호출시 실인자> )
  . 실행을 멈추려면 clearInterval() 호출
- 지연 실행 : setTimeout([함수], [반복호출시간 : 밀리세컨] , <옵션 : 호출시 실인자> )
- 리턴값이 있는 함수 실행시 리턴값을 받을 수 없음

```
let num = 10;

function plusNum(value, plusnum = 20) {
  let num = value + 10;
  console.log("함수 호출 결과 값은 " + num + "입니다.");
}

setTimeout(plusNum, 2000, 10);

let timer = setInterval(plusNum, 2000, 10);
setTimeout(() => {
  clearInterval(timer);
  console.log("타이머 종료");
}, 6000);
```

# _콜백(Callback)

- 함수의 매개변수로 전달되며 이벤트발생시 호출되는 함수
- 이벤트 : 타이머, UI 조작 이벤트, 네트워크 통신 이벤트

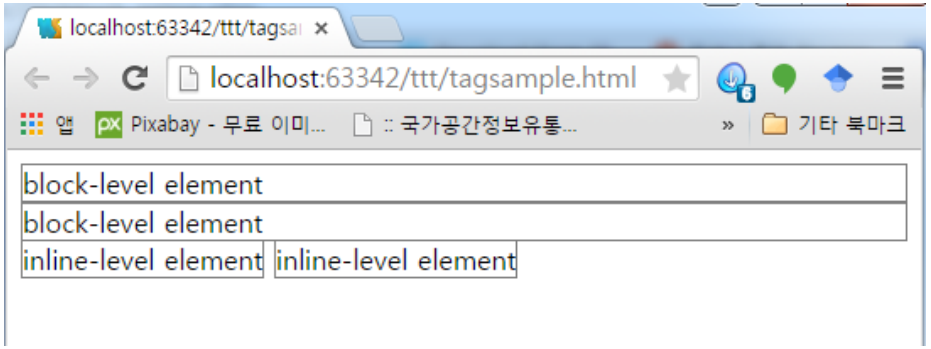
```
function callTenTimes(callback) {
   for (let i = 0; i < 10; i++) {
        // 매개 변수로 전달된 함수를 호출합니다.
        callback();
    }
}

callTenTimes(function () {   // 익명함수 정의
    console.log('함수 호출');
});
```

# _HTML5 기본

HTML 문법 기술 방법

- MIME 타입 : text/html
- HTML5 파일 확장자 : .html
- HTML5 파일의 시작부분에 DOCTYPE 선언
  <!DOCTYPE html>
- 문자 인코딩 지정 방법
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
- block-level 태그 inline-level 태그 : 한 라인을 차지 하느냐 vs 한 라인에 여러 개가 오느냐

```
<div style="border:1px gray solid;">block-level element</div>
<div style="border:1px gray solid;">block-level element</div>

<span style="border:1px gray solid;">inline-level element</span>
<span style="border:1px gray solid;">inline-level element</span>
```




block level tags

p, h1~h6, ul, ol, pre, div, form, hr, table

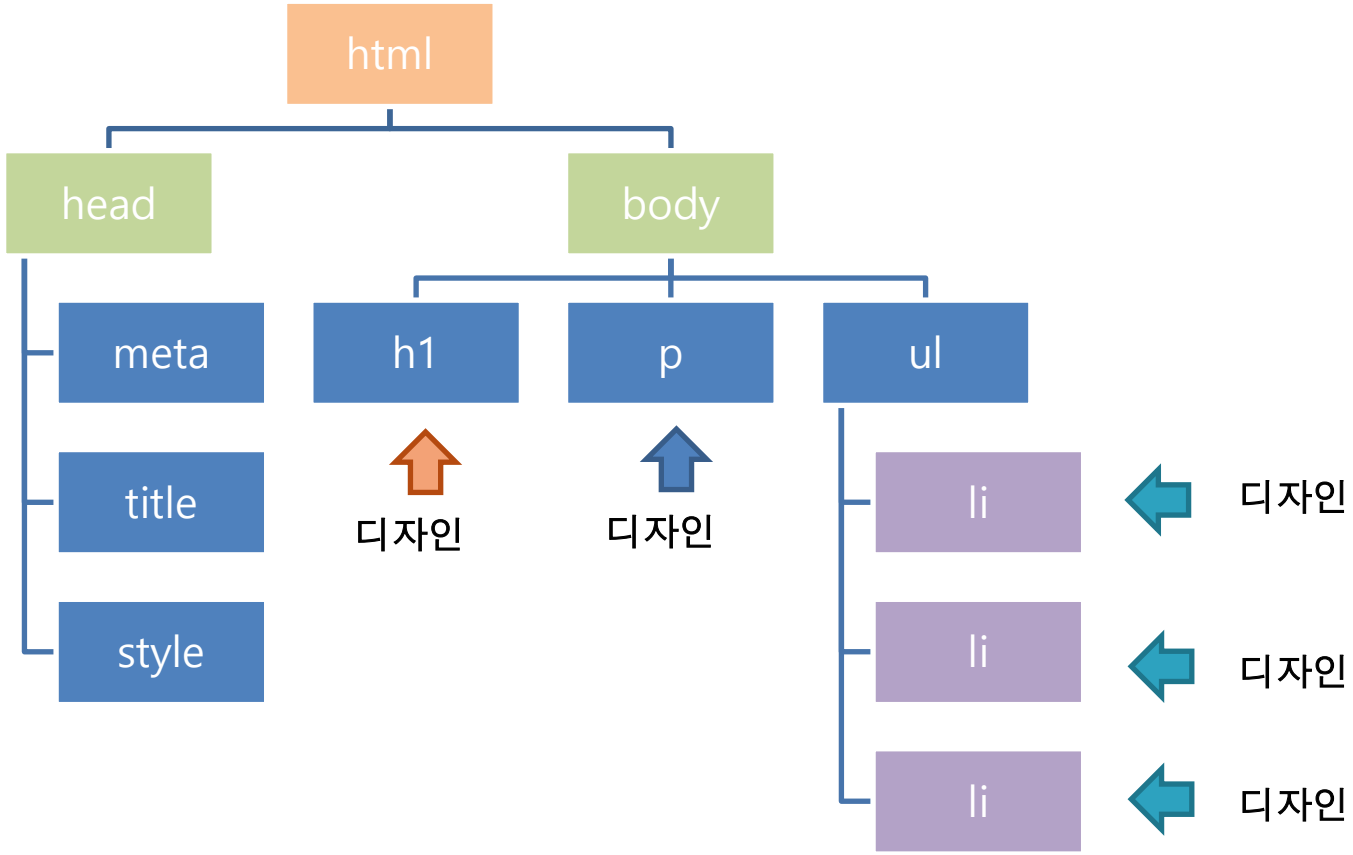
inline level tags

a, img, object, br, script, map, span, input, select, textarea, label, button

* 특정 컨테이너(부모 컨테이너) 내부에 있으면 컨테이너 크기 만큼 차지

# _CSS 기본

- 마크업 언어가 실제 표시되는 방법을 기술하는 언어
- W3C의 표준이며, 레이아웃과 스타일을 정의할 때의 자유도가 높다



# _CSS 기본

- 룰셋의 구성

- 전역적 기술 : 페이지 모든 태그에 영향 → rule set 정의

H1 { color : red; }

선택자 스타일속성 스타일값

```
#header {
    width: 800px; margin: 0 auto;
    background: red;
}
```

- 지역적 기술 : 해당 태그에 영향 → style 속성 지정

style="color : red;"

스타일속성 스타일값

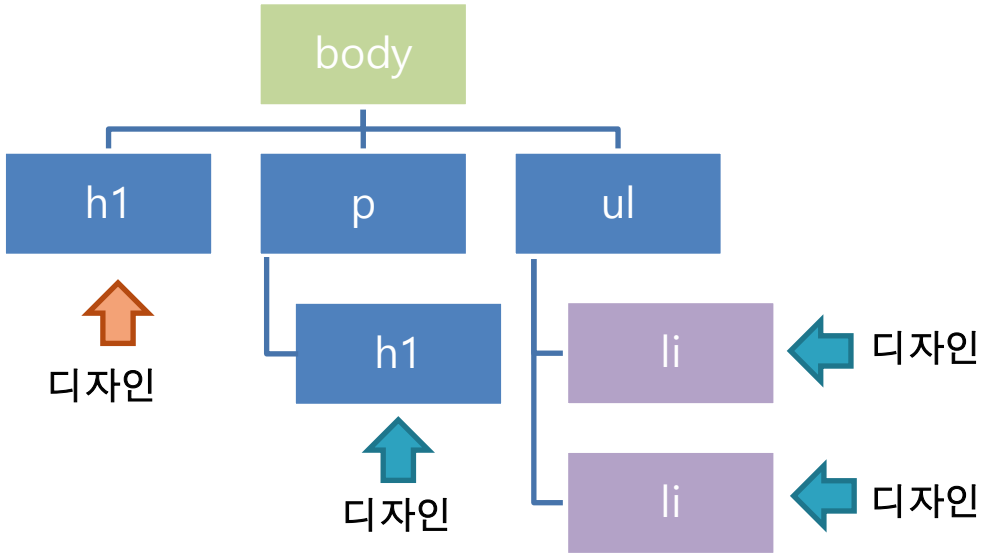
```
<div style="border:1px gray solid;">
    block-level element
</div>
<div style="border:1px gray solid;">block-level element</div>

<span style="border:1px gray solid;display: block">
    inline-level element
</span>
```

# _CSS 기본

- 클래스와 id

- **ID : 특정 태그에 대해서만 적용**

```
#myid { color:orange; }

<h1 id="myid"> ID : </h1>
```




- **class : 특정 태그들 (사용자가 정한 묶음)에 적용할 때**

```
.myclass { color:blue; }

<h1 class="myclass"> ID : </h1>
<li class="myclass"> Name : </li>
<li class="myclass"> phone : </li>
```

# _CSS Selector

- 선택자(Selector) 조건 지정 방법

| Selector | 설 명 |
|---|---|
| * | 모든 엘리먼트와 일치 |
| E | 태그명이 E인 모든 엘리먼트와 일치 ex) H1{ color:red; } |
| E,F | 태그명이 E 또는 F 인 모든 엘리먼트 ex) H1,H2{ color:red; } |
| E F | E의 자손이면서 엘리먼트의 이름이 F인 모든 엘리먼트 ex) H1 DIV{ color:red; } |
| .C | 클래스 속성값이 C인 모든 엘리먼트 ex) .myclass { color:red; } |
| #I | 아이디 속성값이 I인 모든 엘리먼트 ex) #myId { color:red; } |
| E[A=V] | 어트리뷰트 A의 값이 V인 모든 E엘리먼트<br>ex) div[myattr="hello"] { color:red; }<br>div[display=none] { color:red; } |

※ Selector는 document.querySelectorAll()에서 객체 지정시에도 사용된다.

# _CSS 우선순위

1. 속성값 뒤에 !important 를 붙인 속성

```
p {
    font-size: 40px !important;
    color: green !important;
}
```

2. 지역 스타일 지정 : 태그의 style 프로퍼티에 지정
3. #id 로 지정한 속성
4. .클래스로 지정한 속성
5. 태그이름 으로 지정한 속성
6. 나중에 지정된 속성

# _HTML CSS 예제

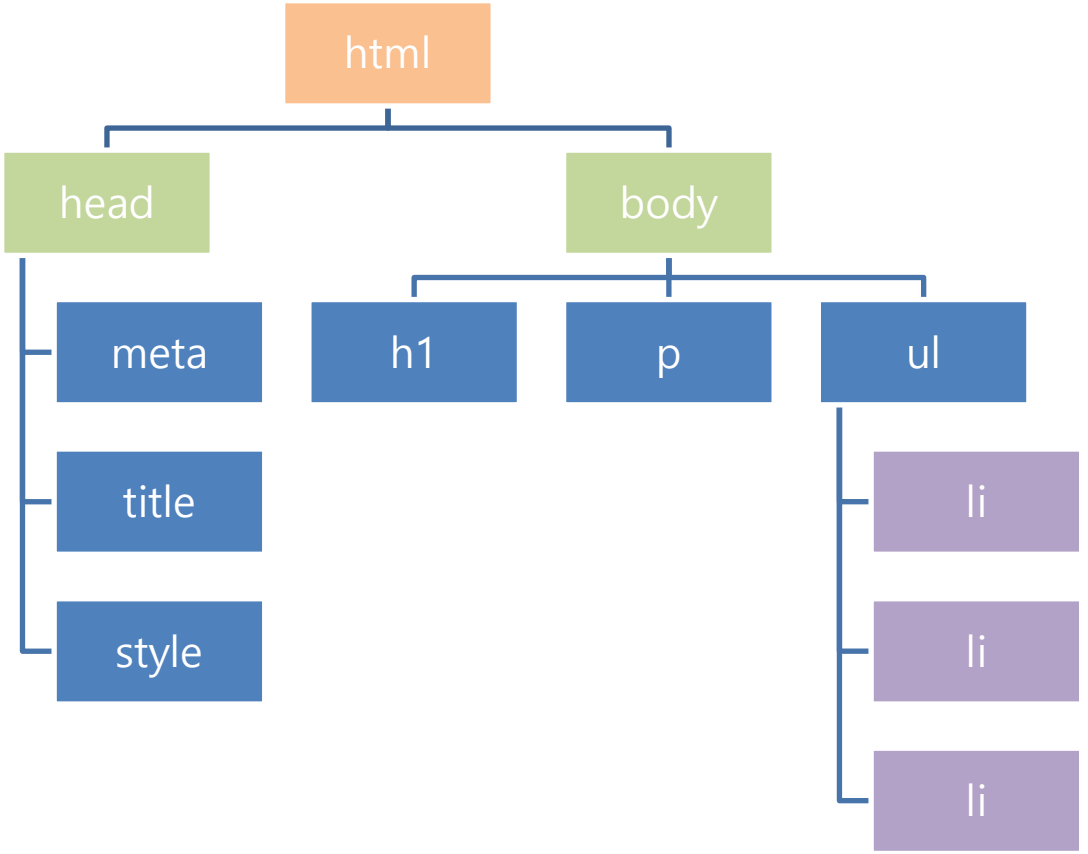
```
<style>
  .sports {
    font-size: 20px;
    font-weight: bold;
    color: blue;
  }

  .rpg {
    font-size: 15px;
    font-weight: bold;
    color: crimson;
    border: 3px solid rgb(7, 72, 194);
  }

  #game6 {
    font-size: 25px;
    font-weight: bold;
    color: crimson;
    border: 3px dashed rgb(12, 247, 247);
  }

</style>
```

```
<body>
 <div>
   <div id="game1" class="sports">NBA 2022</div>
   <div id="game2" class="sports">FIFA 2022</div>
   <div id="game3" class="sports">NFL 2022</div>

   <br />
   <br />

   <div id="game4" class="rpg">WoW</div>
   <div id="game5" class="rpg">Jelda</div>
   <div id="game6" class="rpg">diablo</div>
 </div>
</body>
```

# _DOM (Document Object Model)

- 문서 객체 모델 : document 객체와 관련된 객체의 집합을 나타냄
- HTML 태그를 자바스크립트에서 사용할 수 있는 객체로 만듦. 문서 객체를 조작한다는 말은 태그를 조작한다는 말
- 웹 브라우저는 웹 페이지를 실행 시 HTML 코드를 위에서 아래로 실행함.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="EUC-KR">
        <title>DOM</title>
        <style>
            body {
             color:white;
             background-color:black;
            }
        </style>
    </head>

    <body>
        <h1 title="want">원하는 것</h1>
        <p title="book">보고싶은 책</p>
        <ul id="list">
            <li>멈추면 비로소 보이는 것들</li>
            <li>무지개 원리</li>
            <li>은교</li>
        </ul>
    </body>
</html>
```



# _문서 객체의 선택

- 문서 객체 선택 : 문서 객체를 선택하면 자바스크립트로 실행 중에 내부 글자를 변경하거나 스타일을 변경할 수 있음
- document.querySelector( [선택자] )
  document.querySelectorAll( [선택자])
  document.getElementById( [선택할 Id ] )

| Selector | 설 명 |
|---|---|
| .C | 클래스 값이 C인 모든 엘리먼트<br>ex) .myclass |
| #I | 아이디가 I인 엘리먼트<br>ex) #myId |
| E[A=V] | 어트리뷰트 A의 값이 V인 모든 E엘리먼트<br>ex) div[myattr="hello"]<br>div[display=none] |
| .aaa[A=V] | 클래스가 'aaa'인 것 중 어트리뷰트 A의 값이 V인 모든 엘리먼트<br>ex) aaa[myattr="hello"] |

# _문서 객체의 조작

1. 객체 내부 (태그 사이) 조작 : [요소객체].innerHTML [요소객체].textContent

2. 스타일 조작 (style 속성) : [요소객체].style.[스타일명]

3. 일반 속성 조작
   - setAttribute([속성이름], [속성 값]), getAttribute([속성이름])
   - 웹표준 속성은 '[요소객체].[속성명]' 으로 접근 가능 ex) image.src

4. 클래스 조작 : 자바스크립트에서는 클래스명으로 요소를 묶어서 처리하는 일이 많으므로 매우 중요
   - 클래스 추가 : [요소객체].classList.add([추가클래스]);
   - 클래스 삭제 : [요소객체].classList.remove([삭제클래스]);

5. UI 값 조작 : UI에 입력된 값을 대상
   - 값 얻기 : [요소객체].value
   - 값 수정 : [요소객체].value = [새값];

# _DOM 기본 예제

```
<div>
    <div id="game1" class="sports" year="2022">NBA 2022</div>
    <div id="game2" class="sports" year="2022">FIFA 2022</div>
    <div id="game3" class="sports" year="2022">NFL 2022</div>

    <br />
    <br />

    <div id="game4" class="rpg" year="2008">WoW</div>
    <div id="game5" class="rpg" year="1990">Jelda</div>
    <div id="game6" class="rpg" year="1998">diablo</div>

    <div id="game7" year="2022">Marvel Fighter</div>
</div>
```

# _DOM 기본 예제

```
const game1Obj = document.querySelector("#game1");

game1Obj.innerHTML = "NBA 2000";
// game1Obj.textContent = "NBA 2000";

game1Obj.setAttribute("year", "2000");
console.log(`game1Obj year attribute = ${game1Obj.getAttribute("year")} `);

const gameEls = document.querySelectorAll(".sports");
gameEls.forEach((el) => {
        el.style.border = "1px solid blue";
        el.style.padding = "24px";
        el.style.width = "48px";
});
const game7Obj = document.getElementById("game7");
game7Obj.classList.add("rpg");

const rpggameEls = document.querySelectorAll(".rpg");
rpggameEls.forEach((el) => {
        console.log(el);
});

const hiddenObjs = document.querySelectorAll(".sports[year='2022']");
hiddenObjs.forEach((el) => {
        el.style.display = "none";
});
```

# _DOM 기본 예제

```
const game1Obj = document.querySelector("#game1");

game1Obj.innerHTML = "NBA 2000";
// game1Obj.textContent = "NBA 2000";

game1Obj.setAttribute("year", "2000");
console.log(`game1Obj year attribute = ${game1Obj.getAttribute("year")} `);

const gameEls = document.querySelectorAll(".sports");
gameEls.forEach((el) => {
        el.style.border = "1px solid blue";
        el.style.padding = "24px";
        el.style.width = "48px";
});
const game7Obj = document.getElementById("game7");
game7Obj.classList.add("rpg");

const rpggameEls = document.querySelectorAll(".rpg");
rpggameEls.forEach((el) => {
        console.log(el);
});

const hiddenObjs = document.querySelectorAll(".sports[year='2022']");
hiddenObjs.forEach((el) => {
        el.style.display = "none";
});
```

# _기본 UI 태그

## 1. 버튼

### ① HTML

<INPUT TYPE=BUTTON VALUE=[버튼 위 텍스트] onClick=[이벤트 핸들러]>


### ② JavaScript

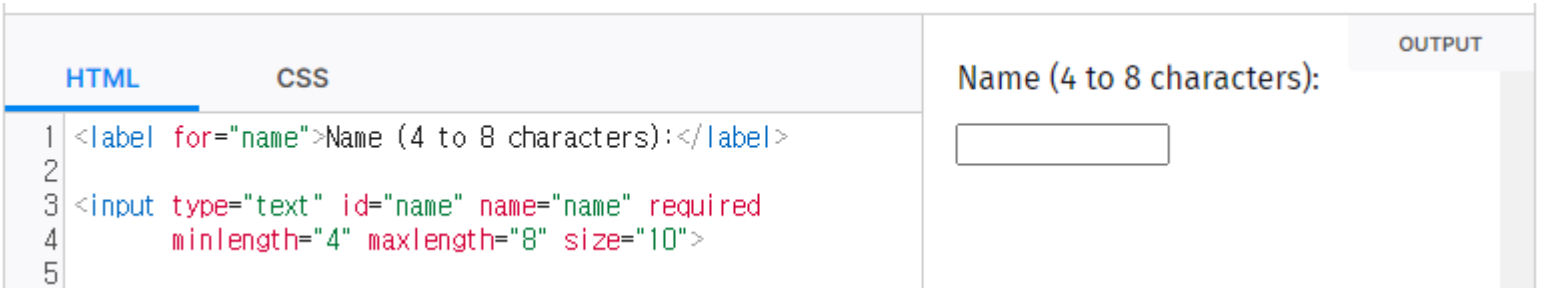
| | | |
|---|---|---|
| **속성** | form | 버튼의 상위 폼 객체 |
| | name | 버튼의 이름 |
| | type | 버튼의 타입 |
| | value | 버튼의 값 (라벨 텍스트) |
| **메소드** | blur() | 포커스 해제 |
| | click() | 버튼을 클릭 |
| | focus() | 버튼에 포커스를 줍니다 |

# _기본 UI 태그

## 2. text

### ① HTML

<INPUT TYPE="TEXT" VALUE=[값] SIZE=[입력크기]>




* password : 텍스트 상자 객체 입력내용이 *로 보이게 함(1라인 입력)

<INPUT TYPE="PASSWORD" VALUE=[값] SIZE=[입력크기]>

### ② JavaScript

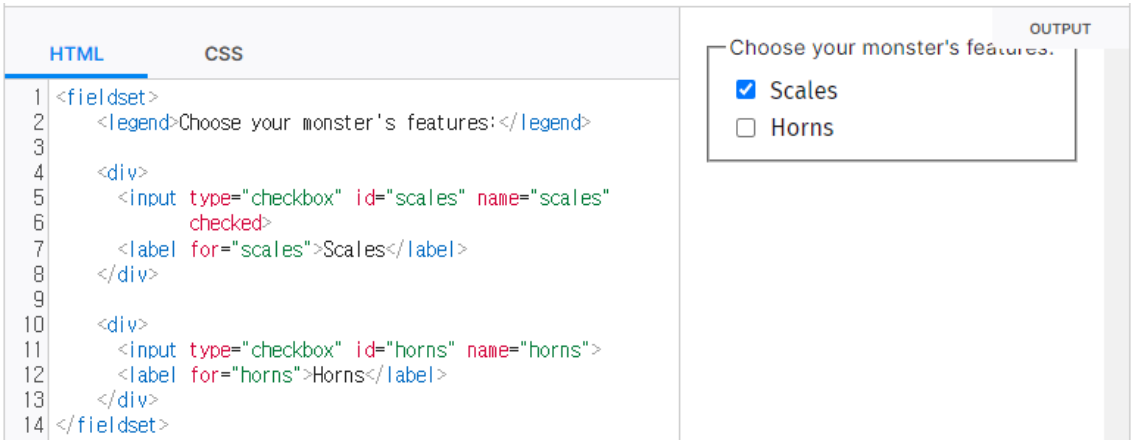
| | | |
|---|---|---|
| 속성 | defaultValue | text box 의 초기문자열 |
| | form | 상위 폼 객체 |
| | name | name 속성 |
| | type | type 속성 |
| | value | value 속성 |
| 메소드 | blur() | 포커스를 해제합니다 |
| | select() | 텍스트박스 입력란의 문자열을 선택 |
| | focus() | 포커스를 줍니다 |

# _기본 UI 태그

## 3. checkbox

### ① HTML

〈INPUT TYPE="checkbox" NAME=[이름] VALUE=[값] [CHECKED] 〉




### ② JavaScript

| | | |
|---|---|---|
| 속성 | checked | 체크되어 있는지 여부(true/false) |
| | defaultChecked | 기본값으로 체크되어 있는지 여부(true/false) |
| | form | 상위 폼 객체 |
| | name | 체크박스의 이름 속성 |
| | type | 체크박스의 타입 속성 |
| | value | 체크박스의 value 속성 |
| 메소드 | blur() | 포커스를 해제합니다 |
| | click() | 체크박스를 클릭합니다 |
| | focus() | 포커스를 줍니다 |

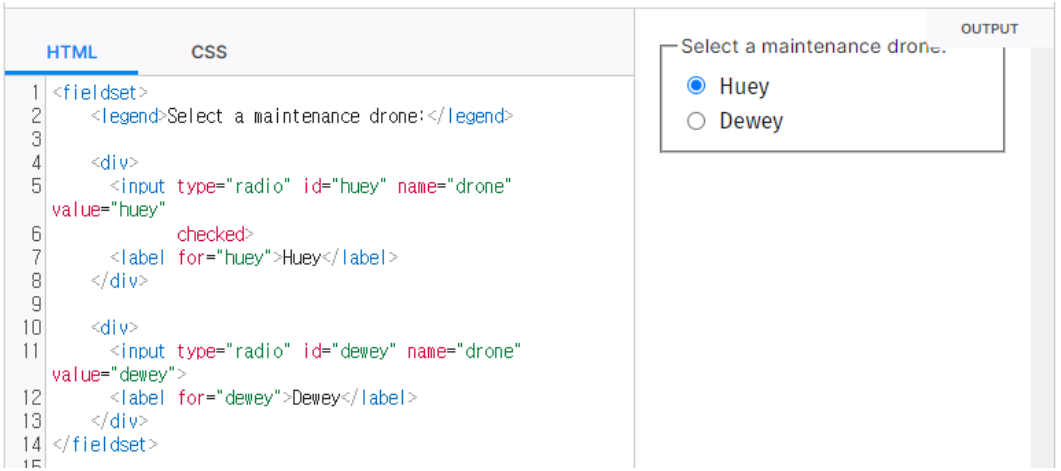
# _기본 UI 태그

## 4. radio

### ① HTML

〈INPUT TYPE="radio" NAME=[이름] VALUE=[값] [CHECKED] 〉

* NAME을 같게 지정해야 한 묶음이 된다.




### ② JavaScript

| | | |
|---|---|---|
| 속성 | checked | 체크되어 있는지 여부(true/false) |
| | defaultChecked | 기본값으로 체크되어 있는지 여부(true/false) |
| | form | 상위 폼 객체 |
| | length | 라디오버튼의 개수 |
| | name | name 속성 |
| | type | type 속성 |
| | value | value 속성 |
| 메소드 | blur() | 포커스를 해제 |
| | click() | 버튼을 클릭 |
| | focus() | 포커스를 줍니다 |

# _이벤트와 이벤트 핸들러

## 1. 이벤트 핸들러 등록

1) 정의 : function을 정의한다. → function 이벤트핸들러 이름 (..) {...}

2) 이벤트 핸들러 등록

- 태그 속성에서 등록 : onclick="myFunction()"
- document.getElementById("demo").**onclick** = function() {myFunction()} ;
- **document.getElementById("demo").addEventListener("click", myFunction);**
  (이벤트 핸들러 복수 등록 가능)

## 2. 이벤트 종류

| Mouse Events | Keyboard Events | Form Events | Document/Window Events |
|---|---|---|---|
| click<br>dblclick<br>mouseenter<br>mouseleave<br>mousedown<br>mouseup<br>hover | keypress<br>keydown<br>keyup | submit<br>change<br>focus<br>blur | load<br>resize<br>scroll<br>unload |

## 3. 이벤트 주요정보

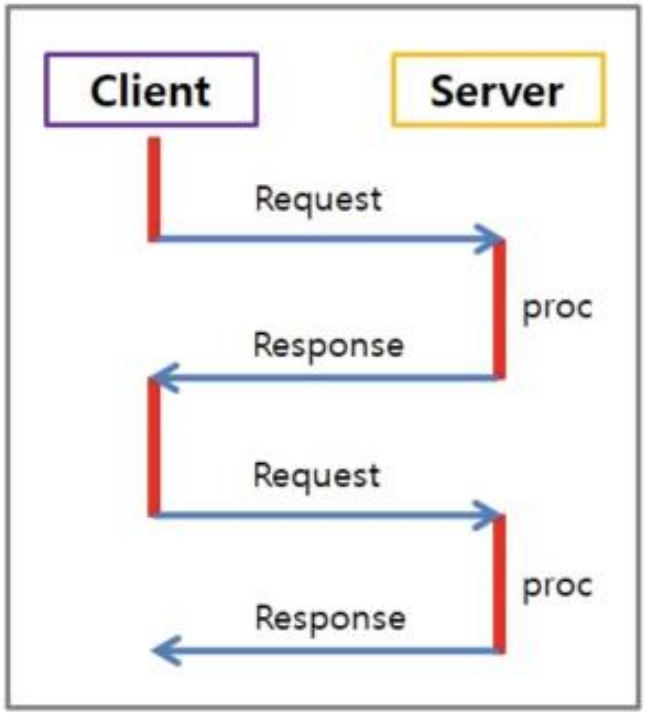
| Mouse Events | Keyboard Events |
|---|---|
| 마우스 위치 정보<br>event.offsetX, event.offsetY<br>event.clientX, event.clientY<br>event.screenX, event.screenY | 키에 대한 정보<br>event.keyCode |

# _이벤트와 이벤트핸들러 예제

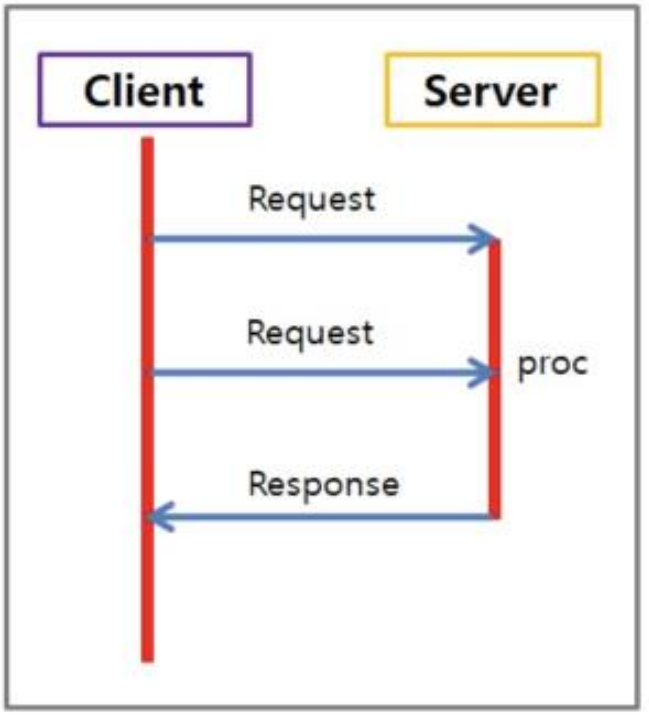
```
<div id="divbox" style="width:200px;height:200px; background: yellowgreen;">
</div>
<div id="mouseXYText"> </div>
<br><br>
<input id="inputbox" type="text" style="width:200px;"/>
<div id="keyCodeText"> </div>

<script>
    function ready() {
        document.getElementById("divbox").addEventListener("click", function(event) {
           document.getElementById("mouseXYText").innerHTML =
            "mouse >>>>  " + event.offsetX + " , " +  event.offsetY ;
        });

        document.getElementById("inputbox").addEventListener("keyup", function(event) {
           document.getElementById("mouseXYText").innerHTML =
              "keyCode >>>>  " + event.keyCode;
        });
    };
</script>
```

# _동기 통신과 비동기 통신

- 동기 통신 : 요청 후 응답을 받아야 다음 동작이 이루어짐
- 비동기 통신 : 요청 후 응답을 기다리지 않고 바로 다음 동작이 이루어지며
  응답이 완료되면 이벤트를 받아서 처리 (콜백 처리)



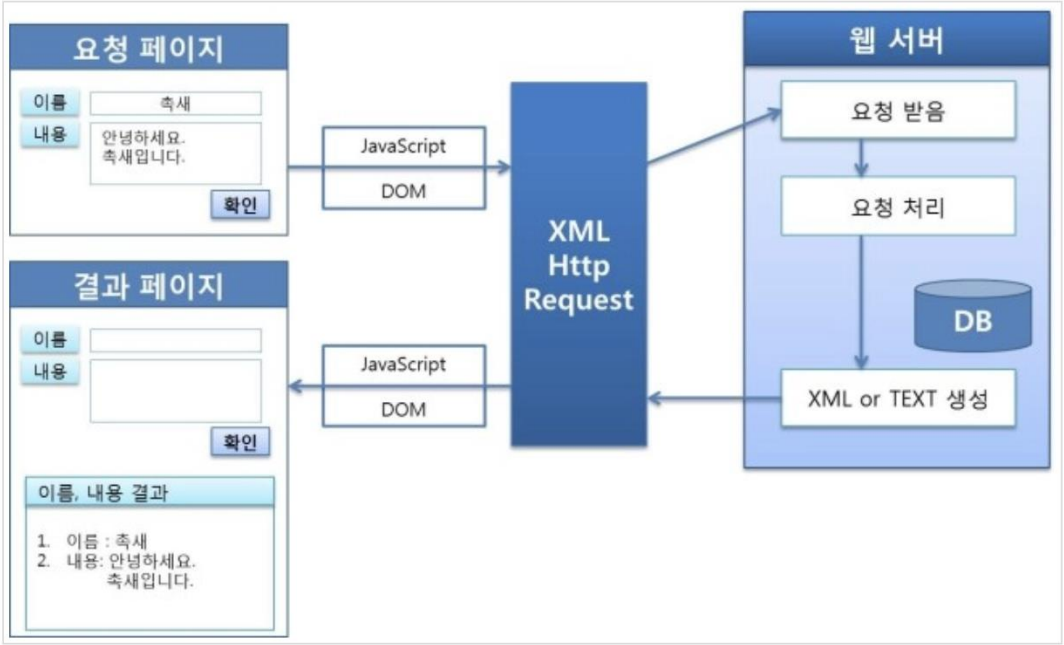

<동기식>




<비동기식>

# _AJAX

- Ajax는 JavaScript의 라이브러리 중 하나이며 Asynchronous Javascript And Xml의 약자
- 브라우저가 가지고있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고도 페이지의 일부만을 위한 데이터를 로드하는 기법
- JavaScript를 사용한 비동기 통신, 클라이언트와 서버간에 XML 데이터를 주고받는 기술
- 사용이유 : 기본적으로 HTTP프로토콜은 클라이언트쪽에서 Request를 보내고 Server쪽에서 Response를 받으면 이어졌던 연결이 끊기게 되어있음. 화면의 내용을 갱신하기 위해서는 다시 request를 하고 response를 하면서 페이지 전체를 갱신하여야 함.
  ajax는 html 페이지 전체가 아닌 일부분만 갱신할수 있도록 XML HttpRequest객체를 통해 서버에 request를 해서 Json이나 xml형태로 필요한 데이터만 받아 갱신하므로 자원과 시간을 아낄 수 있음



# _AJAX 기본 코드

```
var url = "http://dev2.soxcorp.co.kr:17104/mytest";
var request = new XMLHttpRequest();

request.open("GET", url, true);    // 비동기 호출
request.send(null);

request.onreadystatechange = function () {
      if (request.readyState === 4 && request.status === 200) {
        result = request.responseText;
        result = JSON.parse(result);
        console.log(result);
      }
};
```

# _AJAX JQuery 활용 코드

- 형식이 쉽고 직관적이어서 많이 사용
- POST/GET 동일한 방식으로 호출 (파라미터를 싣는 방식이 동일)

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

$.ajax({
      type : "GET",                                          //전송방식 (POST,GET)
      url: "http://dev2.soxcorp.co.kr:17104/mytest",     //호출 URL
      data: { name: "soxcorp", age: 4 },                  // 파라미터 지정
      dataType: "text", // 페이지 형식
      success: function (data) {
        console.log(JSON.parse(data));                   // 받아온 값
      },
      error: function () {
        alert("통신실패!!!!");
      }
});
```

* http://make-random.com/MakeRandom/api/userInfo.get

# 응용 예제

# 실습

0~9 임의의 수 5개를 생성해서 배열에 넣되 중복된 수를 넣지 않도록 하는 프로그램

# 실습

다음 CSV 문자열을 파싱하여 평균을 구하시오.

"100,60,78,33,60,87,65,88,90,100"

# 실습

**배열로 된 학생들의 성적을 객체 배열로 전환 하시오.**

```
let name = ["kim", "lee", "park", "song", "hong"];
let kor = [100, 60, 78, 33, 60];
let math = [60, 70, 79, 63, 61];
let eng = [60, 70, 70, 60, 60];
```

# 실습

이전 예제에서 만든 객체 배열을 이름 순으로 출력하시오.

{name: 'kim', kor: 100, math: 60, eng: 60}
{name: 'lee', kor: 60, math: 70, eng: 70}
{name: 'park', kor: 78, math: 79, eng: 70}
{name: 'song', kor: 33, math: 63, eng: 60}
{name: 'hong', kor: 60, math: 61, eng: 60}

# 실습

**1초에 한번씩 현재 시간을 출력하도록 타이머를 만드시오.**

현재 시간 구하기

```
let now = new Date();
let timeStr = now.toLocaleString();
```

# 실습

AJAX로 다음 API의 결과를 받아와 출력하는 프로그램을 작성하시오.

http://make-random.com/MakeRandom/api/userInfo.get


THANK
YOU!