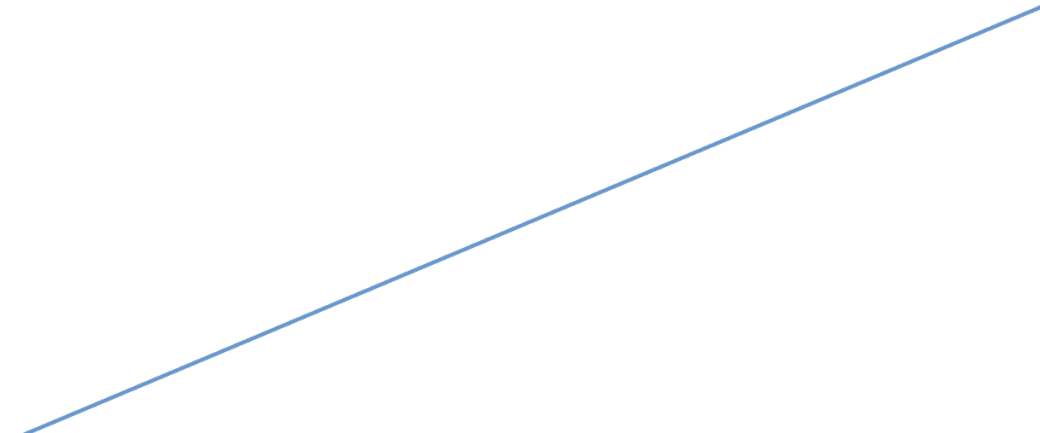


프론트엔드 개발 실무

[6장] JavaScript 기초

디노웍스
이 시 영

목차

- JavaScript 개요
 - JavaScript Core
 - JavaScript DOM
 - JavaScript BOM
 - JavaScript Class
 - UI 개발 실습
- 

➤ JavaScript란?

- 웹의 프로그래밍 언어로 웹의 **동작**을 담당
- 데이터를 계산, 조작, 검증할 수 있으며, **HTML과 CSS를 변경**할 수 있음

➤ 어디에 작성하는가?

- HTML 문서의 <script> 태그 내부에 작성
- <head>, <body> 요소의 내부에 직접 작성
- 외부파일(.js)로 작성 후 HTML 문서로 연결
 - HTML과 코드를 분리하여 코드를 읽고 **유지 관리**하기가 쉬워짐
 - **캐시**된 JavaScript 파일은 페이지 로드 **속도**를 높일 수 있음

➤ 주요 기능 구분

- **CORE** : 변수, 함수, 제어문, 클래스 등 프로그래밍 언어 기반 제공
- **DOM**(Document Object Model) : HTML 문서를 조작
- **BOM**(Browser Object Model) : 브라우저(window)를 조작

➤ JavaScript의 작성 위치

코드 예제

```
<!DOCTYPE html>
<html>
  <head>
    ...
    <script>
      alert('Hello World!');
    </script>
    <script src="./js/script.js"></script>
  </head>
  <body>
    <h1>My First Web Page</h1>
    <p>My First Paragraph</p>
    <p id="demo"></p>
    ...
  </body>
</html>
```

➤ JavaScript의 작성 위치

코드 예제

```
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    <h1>My First Web Page</h1>
    <p>My First Paragraph</p>
    <p id="demo"></p>

    <script>
      alert('Hello World!');
    </script>
    <script src="./js/script.js"></script>
  </body>
</html>
```

➤ 출력문

- `innerHTML` / `innerText` : HTML 요소 내부에 출력
- `document.write()` : HTML 문서 출력 (실행되는 시점의 문서 위치에 작성)
- `window.alert()` : 브라우저 알림창 출력
- `console.log()` : 브라우저 콘솔에 출력

코드 예제

```
<script>

document.getElementById('test').innerHTML = 'Hello World';

document.write('Hello World');

alert('Hello World!');

console.log('Hello World!');

</script>
```

➤ 문법

- 리터럴(Literals) : 숫자, 문자열 등
- 변수(Variable)
- 키워드(Keyword)
- 식별자(Identifier) : 변수명, 함수명 등 저장공간을 지정하는 이름
 - 문자, _, \$로 시작되며 2번째 글자 이후에는 숫자도 포함
 - 예약어(reserved keyword)는 사용할 수 없음
 - 대소문자를 가림(case-sensitive)
- 연산자(Operators)
- 표현(Expressions) : 값, 변수, 연산자 등의 조합으로 이루어진 문장
 - `let lastName = "Doe";`
- 대소문자는 다른 것으로 간주
 - Lower Camel Case 방식으로 기술하는 것이 관례 : firstName, interCity 등
- 주석 : `//`, `/* */`

➤ 변수(Variables)

- 값을 저장
- 변수의 종류 및 특징

구분	Scope	Redeclare	Reassign	Hoisted	Binds this
var	Function	O	O	O	O
let	Block	X	O	X	X
const	Block	X	X	X	X

코드 예제

```
let carName = "Volvo";

const price1 = 5;
const price2 = 10;
let total = price1 + price2;
```


➤ 변수(Variables)

○ 데이터 타입

- **String** : 문자열
- **Number** : 숫자
- **Bigint** : 큰 정수
- **Boolean** : true / false
- **Object** : key-value 쌍
- **Undefined** : 초기값이 할당되지 않은 변수의 값
 - 값이 할당된 적이 없음
- **Null** : 빈 객체
 - 값이 없음
- **Symbol** : 절대 중복되지 않는 값

○ typeof 연산자

- 변수의 데이터 타입을 확인

코드 예제

```
let color = "Yellow";
let lastName = "Johnson";
let length = 16;
let weight = 7.5;
```

```
let x = 1234567890123456789012345n;
let y = BigInt(1234567890123456789012345);
let x = true;
let y = false;
```

```
const person = {
  firstName:"John",
  lastName:"Doe"
};
```

```
const cars = ["Saab", "Volvo", "BMW"];
const date = new Date("2022-03-25");
```

```
let x;
let y;
```

```
let x = null;
let y = null;
```

➤ 연산자(Operators)

○ 산술연산자

- `+, -, *, **, /, %, ++, --`

○ 할당연산자

- `=, +=, -=, *=, **=, /=, %=, :`

○ 비교연산자

- `==, ===, !=, !==, >, <, >=, <=`

코드 예제

```
let x = 5;
let y = 2;
let z = x + y;

let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;

let age = 20;
if (age < 18) text = "Too young to buy alcohol";
```

➤ 배열(Array)

- 하나의 변수에 여러 개의 값을 저장
- 각각의 요소에 인덱스 번호로 접근 → **순서**가 중요

코드 예제

```
let car1 = "Saab";
let car2 = "Volvo";
let car3 = "BMW";

const cars = ["Saab", "Volvo", "BMW"];
const cars = new Array("Saab", "Volvo", "BMW"); // 권장되지 않음

cars[0] = "Saab";
cars[1] = "Volvo";
cars[2] = "BMW";

const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;
```

➤ 객체(Object)

- 하나의 변수에 여러 개의 값을 저장
- 각각의 요소에 속성명으로 접근 → **순서가 중요하지 않음**

코드 예제

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};  
  
const person = {};  
  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";  
  
person['firstName'] = "Lee";
```

➤ 객체(Object)

- 메서드 : 함수도 저장 가능 (변수는 **프로퍼티**)

코드 예제

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  id: 5566,  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
};  
  
name = person.fullName();
```

➤ 함수(Function)

- 재사용 가능한 코드 블록
- 구성요소
 - 매개변수, 코드블록, 반환값(return)

코드 예제

두 숫자의 곱을 계산하고 값을 반환하는 함수

```
function myFunction(p1, p2) {  
  return p1 * p2;  
}
```

```
let result = myFunction(3, 4);      # 함수 호출 및 반환값 사용
```

무한 개수의 인수를 배열로 처리

```
function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}
```

```
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

➤ 화살표 함수(Arrow Function)

- 함수 표현식에 더 짧은 구문 허용

코드 예제

```
let myFunction = function(a, b) {  
  return a * b  
}  
# arrow 함수  
let myFunction = (a, b) => a * b;  
  
let hello = function() {  
  return "Hello World!";  
}  
# arrow 함수  
let hello = () => {  
  return "Hello World!";  
}  
let hello = () => "Hello World!";  
let hello = (val) => "Hello " + val;
```

➤ 제어문

- 조건문 : if, else if, else, switch, case

코드 예제

```
let time = 15;
let greeting = '';

if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}

let text = (age < 18) ? "Minor" : "Adult";
```


➤ 제어문

- 조건문 : if, else if, else, switch, case

코드 예제

```
let text = '';

switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

➤ 제어문

- 반복문 : for, while, do-while

코드 예제

```
for (let i = 0; i < 5; i++) {  
  text += "The number is " + i + "<br>";  
}
```

```
while (i < 10) {  
  text += "The number is " + i;  
  i++;  
}
```

```
do {  
  text += "The number is " + i;  
  i++;  
}  
while (i < 10);
```

➤ 제어문

- 반복문 : for, while, do-while

코드 예제

```
for (let i = 0; i < 5; i++) {  
  text += "The number is " + i + "<br>";  
}
```

```
while (i < 10) {  
  text += "The number is " + i;  
  i++;  
}
```

```
do {  
  text += "The number is " + i;  
  i++;  
}  
while (i < 10);
```

➤ 제어문

- 반복문 : for ... in, foreach

코드 예제

```
const person = {fname: 'Jonh', lname: 'Doe', age: 25}
```

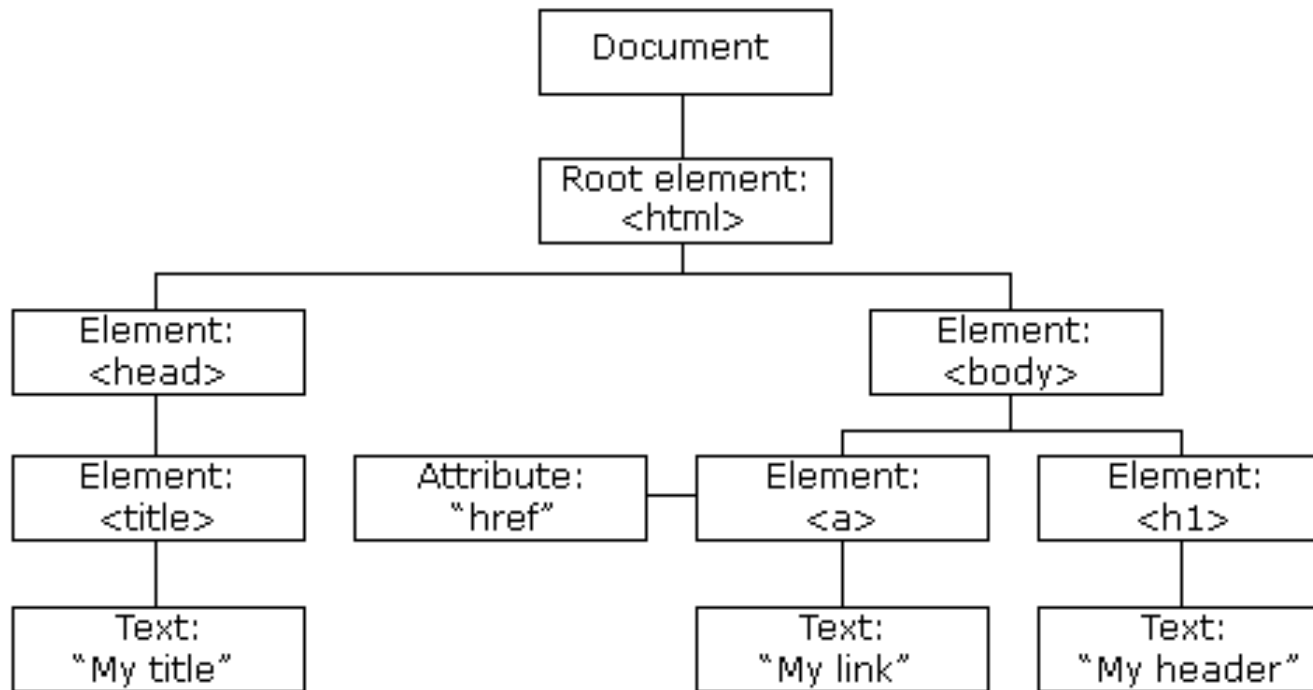
```
let text = '';  
for (let x in person) {  
  text += person[x];  
}
```

```
const testArray = [23, 42, 34, 23]
```

```
testArray.forEach((el, i) => {           // el : n번째 요소, i : 인덱스 번호(0 ~ )  
  console.log(el, i);  
});
```

➤ HTML DOM

- 웹사이트가 로드되면 브라우저는 해당 페이지의 DOM(Document Object Model)을 생성
- JavaScript의 **document** 객체에 관련 기능이 포함되어 있음



➤ HTML 요소 선택

- `document.getElementById()`
- `document.querySelector()`
- `document.querySelectorAll()`

코드 예제

```
// HTML의 ID 속성을 이용하여 선택 (1개)
const header = document.getElementById('header')
console.log(header)

// CSS의 선택자를 이용하여 선택 (1개)
const btn = document.querySelector('div.image-slide div.control a')
console.log(btn)

// CSS의 선택자를 이용하여 선택 (여러 개 / 배열)
const btns = document.querySelectorAll('div.image-slide div.control a')
console.log(btns)
btns.forEach((btn, i) => {
  console.log(btn, i);
});
```

➤ 이벤트 설정

○ document.addEventListener()

- 이벤트 종류 : click, mouseenter, mouseleave, focusin, focusout, touchstart, touchend, ...

코드 예제

```
// 1개의 요소에 이벤트 설정
const btnPrev = document.querySelector('a.prev');
btnPrev.addEventListener('click', () => {
  alert('prev button');
}, false);

// 여러 개의 요소에 이벤트 설정
const btns = document.querySelectorAll('div.control a');
btns.forEach((btn, i) => {
  btn.addEventListener('click', () => {
    alert('button : ' + i);
  }, false);
});
```

➤ HTML 속성 다루기 (get / set)

- `element.attr / element.attr = value`
- `element.getAttribute()`
- `element.setAttribute()`

코드 예제

```
// 속성 프로퍼티로 get / set
const imgElement = document.querySelector('div.slide ul li:nth-child(2) img');
console.log(imgElement.src);
imgElement.src = '../img/main-visual-k2.jpg';
imgElement.style.border = '2px solid red';
imgElement.style.borderColor = 'blue';

// 속성관련 메서드 사용
const imgElement = document.querySelector('div.slide ul li:nth-child(2) img');
console.log(imgElement.getAttribute('src'));
imgElement.setAttribute('src', '../img/main-visual-k2.jpg');
```


➤ HTML class 속성 다루기

◦ `element.classList.xxx()`

- `add()`, `remove()`, `toggle()`, `contains()` ...
- 일반적인 속성과는 달리 여러 개의 값을 다양하게 다루어야 함

코드 예제

```
const imageSlide = document.querySelector('div.image-slide');
const btn = document.querySelector('div.control a.play');

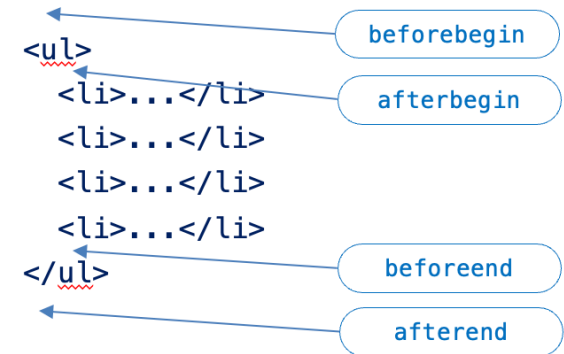
btn.addEventListener('click', () => {
  imageSlide.classList.add('play');

  // imageSlide.classList.remove('play');
  // imageSlide.classList.toggle('on');
  // console.log(imageSlide.classList.contains('play'));

}, false);
```

➤ HTML 구조 다루기

- `element.innerText`
- `element.innerHTML`
- `element.insertAdjacentHTML()`
 - `afterbegin`, `afterend`, `beforebegin`, `beforeend`
- `element.remove()`



코드 예제

```
const header = document.querySelector('#header');
header.innerText = 'website header';
header.innerHTML = '<h1><a href="#"></a></h1>';
header.innerText = '<h1><a href="#"></a></h1>';

const slideContainer = document.querySelector('div.slide ul');
slideContainer.insertAdjacentHTML('beforeend',
  '<li><a href="#"></a></li>');

slideContainer.remove();
```

➤ 새창 열기

- `window.open()` / `window.close()`;

코드 예제

```
const btn1 = document.querySelector('a.play');
const btn2 = document.querySelector('a.prev');
let myWindow = null;

btn1.addEventListener('click', () => {
  window.open("https://www.w3schools.com/");
  myWindow = window.open("", "popup1", "left=0, top=0, width=500, height=600");
}, false);

btn2.addEventListener('click', () => {
  myWindow.close();
}, false);
```

➤ 타이머 다루기

- setTimeout() / clearTimeout()
- setInterval() / clearInterval()

코드 예제

```
const btn1 = document.querySelector('a.play');
const btn2 = document.querySelector('a.prev');
let timerId = null;

btn1.addEventListener('click', () => {
  timerId = setTimeout(() => {console.log('timeout');}, 2000);
  // timerId = setInterval(() => {console.log('interval');}, 1000);
}, false);

btn2.addEventListener('click', () => {
  clearTimeout(timerId);
  // clearInterval(timerId);
}, false);
```

➤ 클래스(Class)

- 객체 지향형 프로그래밍의 핵심
- 모든 대상을 프로퍼티(속성)과 메소드(동작)을 가진 객체로 정의
- 구성요소
 - 생성자(Constructor)
 - 프로퍼티(property) / 메소드(method)
 - public / private, static
 - 상속
 - 인스턴스

➤ 클래스(Class)

코드 예제

```
class Soldier {  
  hp = 100;  
  #name = '';  
  
  constructor(name) {  
    this.#name = name;  
  }  
  
  getName() {  
    return this.#name;  
  }  
  
  #heal() {  
    this.hp += 10;  
  }  
}
```

```
const soldier1 = new Soldier('John');  
const soldier2 = new Soldier('Jane');  
let hit1 = soldier1.hp;  
let name1 = soldier1.getName();  
let name1 = soldier1.#name; // 오류 (외부접근 안됨)
```

클래스



인스턴스

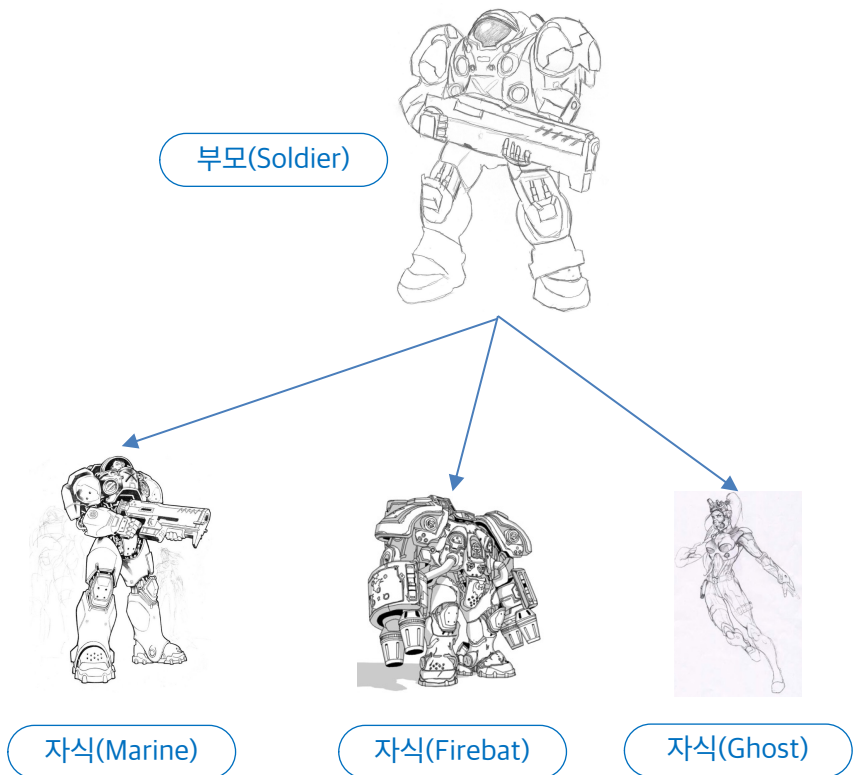
인스턴스



➤ 클래스(Class)

코드 예제

```
class Marine extends Soldier {  
  armor = 1000;  
  
  constructor(name, armor) {  
    super(name);  
    this.armor = armor;  
  }  
  
  attack() {  
    console.log('shot gun');  
  }  
  
  // overriding  
  #heal() {  
    this.hp += 20;  
  }  
}  
  
const marine1 = new Marine('Teddy', 100);  
console.log(marine1.armor);
```



➤ 이미지 슬라이드

이미지 슬라이드 UI



강사 연락처

➤ 디노웍스 대표 이시영

➤ 문의

- E-mail : lsy@dinoworks.kr
 - Mobile : 010-5179-6455
- 