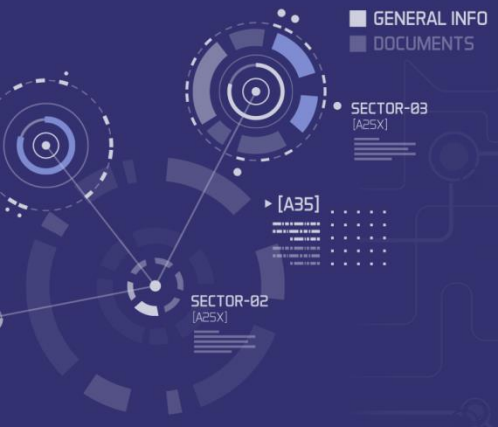
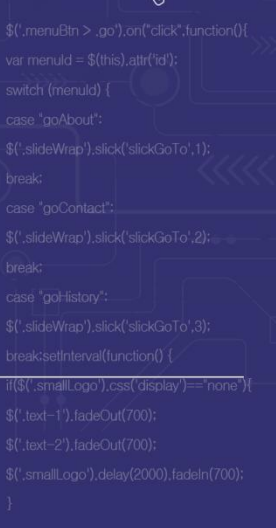
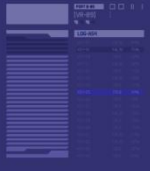


화면 구현 part 2





자바스크립트

KOREATECH
온라인평생교육원

학습목표

- 자바스크립트에 대한 개념을 설명할 수 있다.
- 함수와 제어문을 이해하고, 활용할 수 있다.

학습내용

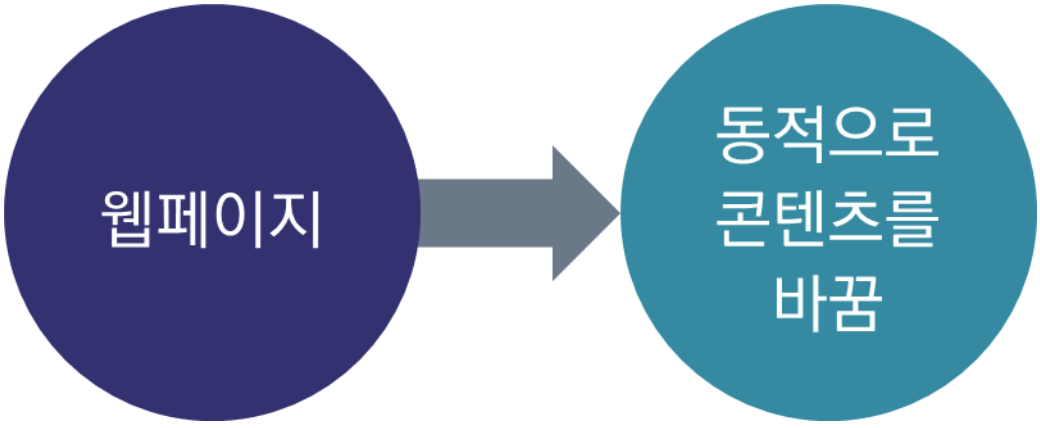
- 자바스크립트(JavaScript)
- 함수와 제어문

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

1) 자바스크립트란?

자바스크립트란?



프로그래밍적으로 제어하기 위해 고안된 언어

- 객체 기반의 스크립트 언어
- 타입을 명시할 필요가 없는 인터프리터 언어
- 객체 지향형 프로그래밍과 함수형 프로그래밍을 모두 표현 가능

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

1) 자바스크립트란?

- 객체 기반의 스크립트 언어
- 타입을 명시할 필요가 없는 인터프리터 언어
- 객체 지향형 프로그래밍과 함수형 프로그래밍을 모두 표현 가능

* 인터프리터: 프로그램을 해석하는 방법 중 하나로, 사람이 이해할 수 있는 고급언어로 작성된 코드를 한 단계씩 해석하여 실행시키는 방법

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

2) 문법

[실행문은 세미콜론 ' ; ' 으로 구분됨]

```
var x = 10;
var y = 13;
```

[대소문자를 구분함]

```
var js = 10;
var JS = 20;
```

→ js와 JS는 서로 다른 변수로 인식함

[상수(Literal)는 직접 표현되는 값 그 자체를 의미함]

```
12.    // 숫자 상수(Literal)
"자바script" // 문자열 상수(Literal)
true   //  boolean 상수(Literal)
```

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

2) 문법

[식별자 작성 방식]

Camel Case 방식

- 식별자가 여러 단어로 이루어질 경우에 첫 번째 단어는 모두 소문자로 작성
- 그 다음 단어부터는 첫 문자만 대문자로 작성하는 방식

Underscore Case 방식

- 식별자를 이루는 단어들을 소문자로만 작성
- 그 단어들은 언더스코어(_)로 연결하는 방식

```
var firstVar = 10;          // Camel Case
var  second_var = 20;       // Undersore Case
```

[주석: 코드 내에 삽입된 일종의 설명문(실행은 되지 않음)]

- 한 줄 주석

```
// 주석문
```

- 여러 줄 주석

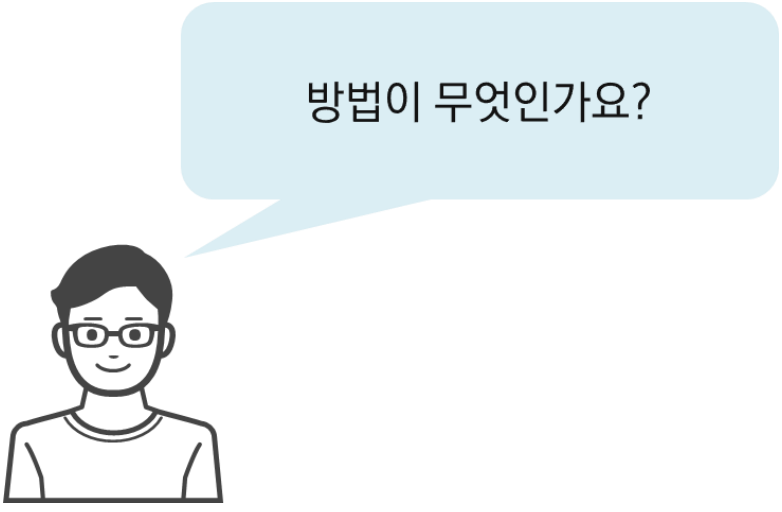
```
/* 주석문 */
```

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

- HTML 페이지에 출력을 할 수 있음



자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[window.alert() 메소드]

입력

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Output</title>
</head>

<body>

    <h1>Window 객체의 alert() 메소드</h1>
    <button onclick="alertDialogBox()">alert 대화 상자
</button>

    <script>
        function alertDialogBox() {
            alert("확인을 누를 때까지 다른 작업을 할 수 없어요!");
        }
    </script>

</body>

</html>
```

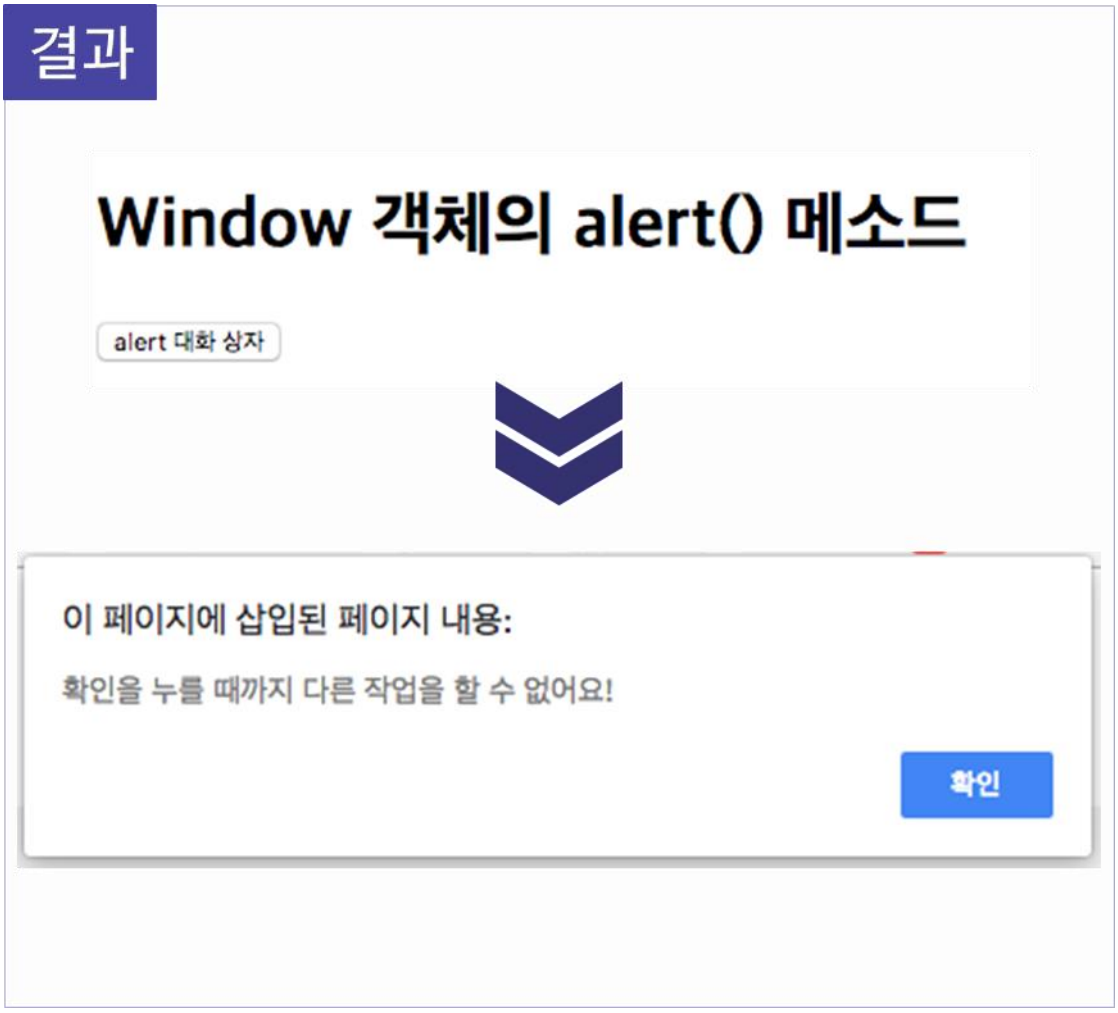
자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[window.alert() 메소드]

결과



자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[HTML DOM 요소를 이용한 innerHTML 프로퍼티]

입력

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Output</title>
</head>

<body>

    <h1>HTML DOM 요소를 이용한 innerHTML 프로퍼티</h1>
    <p id="text">이 문장을 바꿀 것입니다!</p>

    <script>
        var str = document.getElementById("text");
        str.innerHTML = "이 문장으로 바뀌었습니다!";
    </script>

</body>

</html>
```

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[HTML DOM 요소를 이용한 innerHTML 프로퍼티]

결과

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[document.write() 메소드]

입력

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Output</title>
</head>

<body>

    <h1>Document 객체의 write() 메소드</h1>

    <script>
        document.write(4 * 5);
    </script>

</body>

</html>
```

결과

Document 객체의 write() 메소드

20

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

3) 출력

[document.write() 메소드]

입력

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Output</title>
</head>

<body>

    <h1>Document 객체의 write() 메소드</h1>

    <button onclick="document.write(4 * 5)">버튼을 눌러보세
요!</button>

</body>

</html>
```

결과

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

4) 적용 방법

[내부 자바스크립트 코드로 적용]

- 〈head〉, 〈body〉 어디에 추가하던 상관없음

입력

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Apply</title>
    <script>
        function printDate() {
            document.getElementById("date").innerHTML = Date();
        }
    </script>
</head>

<body>

    <h3>head 태그 내의 자바스크립트</h3>
    <button onclick="printDate()">현재 날짜와 시간 표시하기!</button>
    <p id="date"></p>

</body>

</html>
```

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

4) 적용 방법

[내부 자바스크립트 코드로 적용]

- 〈head〉, 〈body〉 어디에 추가하던 상관없음

결과

자바스크립트(JavaScript)

1. 자바스크립트(JavaScript) 기초

4) 적용 방법

[외부 자바스크립트 파일로 적용]

- HTML 코드와 자바스크립트 코드를 분리하면 유지보수 및 가독성이 좋음

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>JavaScript Apply</title>
    <script src="example.js"></script>
</head>

<body>

    <h1>외부 자바스크립트 파일</h1>
    <p>자바스크립트를 이용하면 현재 날짜와 시간 정보에도 손쉽게 접근할 수 있어요!</p>
    <button onclick="printDate()">현재 날짜와 시간 표시하기!</button>
    <p id="date"></p>

</body>

</html>
```

자바스크립트(JavaScript)

2. 타입과 연산자

1) 타입

[원시 타입]

- 숫자 number

```
var num = 10;          // 숫자
```

- 문자열 string

```
var hello = “안녕”; // 문자열
```

- 불리언 boolean

```
var bool = false // boolean 타입 false
```

- 심볼 symbol

```
var sym = Symbol(“javascript”);  // symbol 타입
var symObj = Object(sym);        // object 타입
```

- undefined / null

```
var str;              // undefined
var str = null;       // null
```

자바스크립트(JavaScript)

2. 타입과 연산자

1) 타입

[객체 타입]

- 객체 object

```
var person = { name: “김씨”, age: 20 }; // 객체의 생성
// 객체의 프로퍼티 참조
document.getElementById(“result”).innerHTML =
    “이름: ” + person.name + “나이: ” + person.age ;
```

[타입 변환]

- Number(“10”): 숫자로 타입 변환
- String(true): 문자열로 타입 변환
- Boolean(0): 불리언으로 타입 변환
- Object(2): new Number(3)와 동일한 결과로 숫자 2 객체 변환
- parseInt(): 정수형 타입으로 변환
- parseFloat(): 부동소수점 타입으로 변환

자바스크립트(JavaScript)

2. 타입과 연산자

1) 타입

- 변수 선언: var 키워드를 사용하여 변수를 선언

```
var month; // month라는 이름의 변수 선언
```

```
date = 25; // date라는 이름의 변수를 묵시적으로 선언
```

- 변수 초기화

```
var month;     // 변수의 선언
```

```
var date = 25; // 변수의 선언과 동시에 초기화
```

```
month = 12;    // 변수의 초기화
```

```
var month, date;            // 여러 변수를 한 번에 선언
```

```
var hours = 7, minutes = 15; // 여러 변수를 선언과 동시에 초기화
```

```
month = 10, date = 5;       // 여러 변수를 한 번에 초기화
```

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[산술 연산자]

+	▪ 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 더함
−	▪ 왼쪽 피연산자의 값에서 오른쪽 피연산자의 값을 뺌
*	▪ 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 곱함
/	▪ 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눔
%	▪ 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 나머지를 반환함

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[대입 연산자] 1/2

=	▪ 왼쪽 피연산자에 오른쪽 피연산자의 값을 대입함
+=	▪ 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 더한 후, 그 결과값을 왼쪽 피연산자에 대입함
-=	▪ 왼쪽 피연산자의 값에서 오른쪽 피연산자의 값을 뺀 후, 그 결과값을 왼쪽 피연산자에 대입함

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[대입 연산자] 2/2

*=	▪ 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 곱한 후, 그 결과값을 왼쪽 피연산자에 대입함
/=	▪ 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 결과값을 왼쪽 피연산자에 대입함
%=	▪ 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 나머지를 왼쪽 피연산자에 대입함

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[증감 연산자]

++x	▪ 먼저 피연산자의 값을 1 증가시킨 후에 해당 연산을 진행함
x++	▪ 먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 증가시킴
--x	▪ 먼저 피연산자의 값을 1 감소시킨 후에 해당 연산을 진행함
x--	▪ 먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 감소시킴

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[비교 연산자] 1/2

==	▪ 왼쪽 피연산자와 오른쪽 피연산자의 값이 같으면 참을 반환함
===	▪ 왼쪽 피연산자와 오른쪽 피연산자의 값이 같고, 같은 타입이면 참을 반환함
!=	▪ 왼쪽 피연산자와 오른쪽 피연산자의 값이 같지 않으면 참을 반환함
!==	▪ 왼쪽 피연산자와 오른쪽 피연산자의 값이 같지 않거나, 타입이 다르면 참을 반환함

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[비교 연산자] 2/2

〉	▪ 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크면 참을 반환함
〉=	▪ 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크거나 같으면 참을 반환함
〈	▪ 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작으면 참을 반환함
〈=	▪ 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작거나 같으면 참을 반환함

자바스크립트(JavaScript)

2. 타입과 연산자

2) 연산자

[논리 연산자]

&&	▪ 논리식이 모두 참이면 참을 반환함 (논리 AND 연산)
||	▪ 논리식 중에서 하나라도 참이면 참을 반환함(논리 OR 연산)
!	▪ 논리식의 결과가 참이면 거짓을, 거짓이면 참을 반환함(논리 NOT 연산)

함수와 제어문

1. 함수

1) 자바스크립트 함수

자바스크립트 함수란?

- 하나의 특별한 목적의 작업을 수행하도록 설계된 독립적인 블록
- 자바스크립트에서는 함수도 하나의 데이터 타입

자바스크립트 함수 특징은?

- 함수를 변수에 대입하거나 함수에 프로퍼티를 지정하는 것도 가능
- 다른 함수 내에 중첩되어 정의 가능

함수와 제어문

1. 함수

1) 자바스크립트 함수

function 키워드로 선언	함수의 이름
괄호 안에 쉼표(,)로 구분되는 함수의 매개변수(parameter)	중괄호({ })로 둘러싸인 자바스크립트 실행문

} 구성

```
function 함수이름(매개변수1, 매개변수2,...) {
    함수가 호출되었을 때 실행하고자 하는 실행문;
}
```

2) 반환문

반환문이란?

- **return 키워드** 사용
- 특정한 목적을 수행하고 그 결과를 반환문을 통해 실행된 결과를 전달받음
- 반환문은 함수의 실행을 중단할 수 있음
- 배열이나 객체를 포함한 모든 타입의 값을 반환할 수 있음

함수와 제어문

1. 함수

2) 반환문

입력

```
<!DOCTYPE html>
<html>
<body>
  <h1>함수 반환문</h1>
  <script>
    function multiNum(x, y) {
      return x * y;          // x와 y를 곱한 결과를 반환
    }

    var num = multiNum(3, 4);
    // multiNum() 함수가 호출된 후, 그 반환 값이 변수 num에 저장
    document.write(num);
  </script>
</body>
</html>
```

결과

함수 반환문

12

함수와 제어문

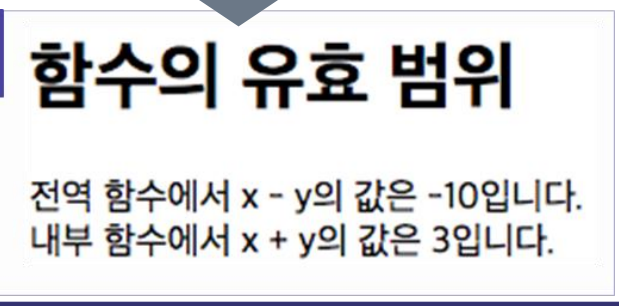
1. 함수

3) 함수의 유효 범위

입력

```
<!DOCTYPE html>
<html>
<body>
  <h1>함수의 유효 범위</h1>
  <script>
    // x, y를 전역 변수로 선언함.
    var x = 10, y = 20;
    // sub()를 전역 함수로 선언함.
    function sub() {
      return x - y;      // 전역 변수인 x, y에 접근함.
    }
    document.write("전역 함수에서 x - y의 값은 " + sub() + "입니다.<br>");

    // parentFunc()을 전역 함수로 선언함.
    function parentFunc() {
      var x = 1, y = 2; // 전역 변수와 같은 이름으로 선언하여 전역 변수의 범위를 제한함.
      function add() {  // add() 함수는 내부 함수로 선언됨.
        return x + y;   // 전역 변수가 아닌 지역 변수 x, y에 접근함.
      }

      return add();
    }
    document.write("내부 함수에서 x + y의 값은 " + parentFunc() + "입니다.<br>");
    </script>
</body>
</html>
```

결과



함수와 제어문

1. 함수

4) 함수 호이스팅

함수 호이스팅이란?

- 자바스크립트 함수 안에 있는 모든 변수의 선언은 함수의 맨 처음으로 이동한 것처럼 동작

입력

```
<!DOCTYPE html>
<html>
<body>
  <h1>함수 호이스팅</h1>
  <script>
    var globalNum = 10;      // globalNum을 전역 변수로 선언함.
    function printNum() {
      document.write("지역 변수 globalNum 선언 전의 globalNum의 값은 " + globalNum + "<br>");
      var globalNum = 20;    // globalNum을 지역 변수로 선언함.
      document.write("지역 변수 globalNum 선언 후의 globalNum의 값은 " + globalNum + "<br>");
    }
    printNum();
  </script>
</body>
</html>
```

결과

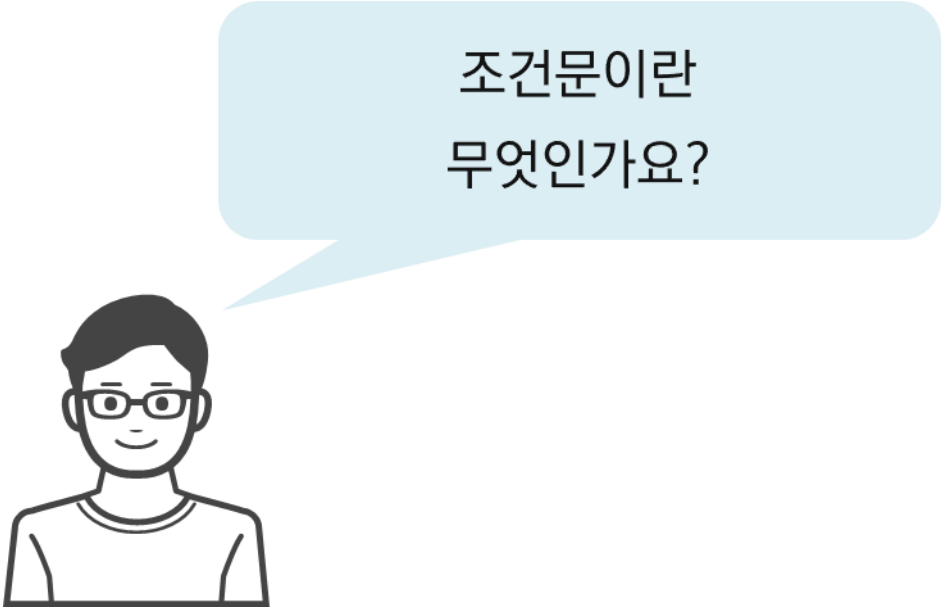
함수 호이스팅

지역 변수 globalNum 선언 전의 globalNum의 값은 undefined
지역 변수 globalNum 선언 후의 globalNum의 값은 20

함수와 제어문

2. 제어문

1) 조건문



프로그램 내에서 주어진 표현식의 결과에 따라
별도의 명령을 수행하도록 제어하는 실행문

함수와 제어문

2. 제어문

1) 조건문 | (1) if 문

- 표현식의 결과가 **참(True)이면**? → 주어진 **실행문을 실행**
- 표현식의 결과가 **거짓(False)이면**? → **아무것도 실행하지 않음**

```
if (표현식) {
    표현식의 결과가 참일 때 실행하고자 하는 실행문;
}
```

함수와 제어문

2. 제어문

1) 조건문 | (1) if 문

입력

```
<!DOCTYPE html>
<html>
<body>

    <h3>if 문</h3>

    <script>
        var x = 5, y = 4;
        if (x == y) {
            document.write("x와 y는 같습니다.");
        }
        if (x < y) {
            document.write("x가 y보다 작습니다.");
        }
        if (x > y) {
            document.write("x가 y보다 큽니다.");
        }
    </script>

</body>

</html>
```

결과

if 문

x가 y보다 큽니다.

함수와 제어문

2. 제어문

1) 조건문 | (2) If - else 문

- 표현식의 결과가 **참(True)이면**? → 주어진 **실행문을 실행**
- 표현식의 결과가 **거짓(False)이면**? → 주어진 **실행문을 실행**

```
if (표현식) {
    표현식의 결과가 참일 때 실행하고자 하는 실행문;
} else {
    표현식의 결과가 거짓일 때 실행하고자 하는 실행문;
}
```

함수와 제어문

2. 제어문

1) 조건문 | (2) If - else 문

입력

```
<!DOCTYPE html>
<html>

<body>

    <h1>if-else 문</h1>

    <script>
        var x = 5, y = 4;
        if (x == y) {
            document.write("x와 y는 같습니다.");
        } else {
            if (x < y)
                document.write("x가 y보다 작습니다.");
            else
                document.write("x가 y보다 큽니다.");
        }
    </script>

</body>

</html>
```

결과

함수와 제어문

2. 제어문

1) 조건문 | (3) If - else if - else 문

- else if 문을 이용하여 다양한 조건을 설정 가능

```
if (표현식1) {
   표현식1의 결과가 참일 때 실행하고자 하는 실행문;
} else if (표현식2) {
   표현식2의 결과가 참일 때 실행하고자 하는 실행문;
} else {
   표현식1의 결과도 거짓이고, 표현식2의 결과도 거짓일 때 실행하고자 하는 실행문;
}
```

함수와 제어문

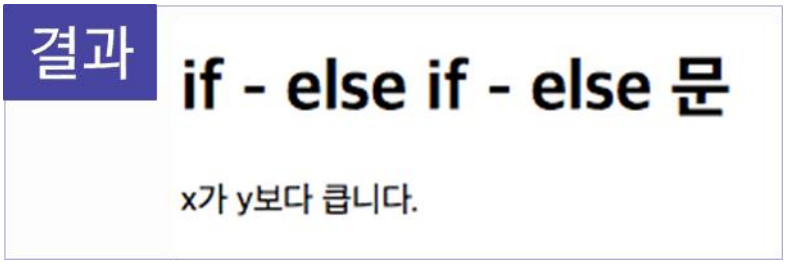
2. 제어문

1) 조건문 | (3) If - else if - else 문

입력

```
<!DOCTYPE html>
<html>

<body>

    <h1>if - else if - else 문</h1>

    <script>
        var x = 5, y = 4;
        if (x == y) {
            document.write("x와 y는 같습니다.");
        }
        else if (x < y) {
            document.write("x가 y보다 작습니다.");
        }
        else { // x > y인 경우
            document.write("x가 y보다 큽니다.");
        }
    </script>

</body>

</html>
```

결과



함수와 제어문

2. 제어문

1) 조건문 | (4) 삼항 연산자에 의한 조건문

- If – else 문을 삼항 연산자를 이용하여 간단하게 표현 가능

표현식 ? 반환값1 : 반환값2

입력

```
<!DOCTYPE html>
<html>

<body>

    <h1>삼항 연산자</h1>

    <script>
        var x = 5, y = 4;
        document.write( (x > y) ? "x가 y보다 큽니다":"y가 x 보다 큽니다");

    </script>

</body>

</html>
```

결과

함수와 제어문

2. 제어문

1) 조건문 | (5) switch 문

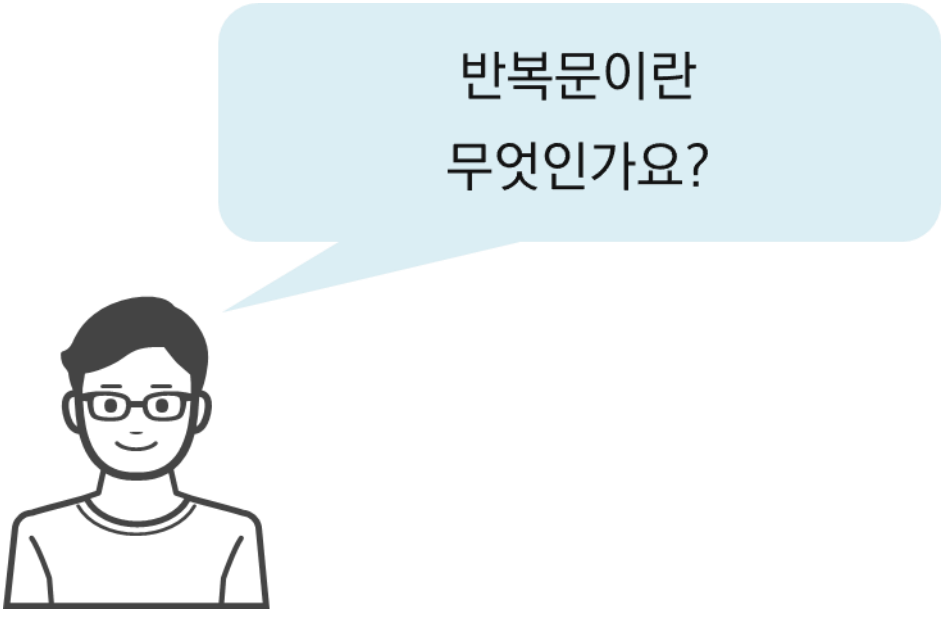
- If – else 문과 같이 주어진 조건 값에 따라 프로그램이 다른 명령을 수행

```
switch (조건 값) {
   case 값1:
      조건 값이 값1일 때 실행하고자 하는 실행문;
      break;
   case 값2:
         조건 값이 값2일 때 실행하고자 하는 실행문;
         break;
   ...
   default:
      조건 값이 어떠한 case 절에도 해당하지 않을 때 실행하고자 하는 실행문;
      break;
}
```

함수와 제어문

2. 제어문

2) 반복문



프로그램내에서 **같은 명령을 일정 횟수만큼 반복**하여
수행하도록 제어하는 실행문

함수와 제어문

2. 제어문

2) 반복문 | (1) while 문

- 특정 조건을 만족할 때까지 계속해서 주어진 실행문을 반복 실행

```
while (표현식) {
    표현식의 결과가 참인 동안 반복적으로 실행하고자 하는 실행문;
}
```

입력

```
<!DOCTYPE html>
<html>

<body>

    <h3>while 문</h3>

    <script>
        var i = 1;

        while (i < 10) {
            document.write(i + "<br>");
            i++;
        }
    </script>

</body>

</html>
```

결과

while 문

1
2
3
4
5
6
7
8
9

함수와 제어문

2. 제어문

2) 반복문 | (2) do - while 문

- 먼저 루프를 한번 실행한 후 표현식을 검사

```
do {
    표현식의 결과가 참인 동안 반복적으로 실행하고자 하는 실행문;
} while (표현식);
```

입력

```
<!DOCTYPE html>
<html>

<body>

    <h3>do - while 문</h3>

    <script>
        var i = 5, j = 1;

        while (i < 10) {
            document.write("i : " + i + "<br>");
            i++;
        }

        do {
            document.write("j : " + (j++) + "<br>");
        } while (j > 3);
    </script>

</body>

</html>
```

결과

do - while 문

i : 5
i : 6
i : 7
i : 8
i : 9
j : 1

함수와 제어문

2. 제어문

2) 반복문 | (3) for 문

- 초기식, 표현식, 증감식 모두 포함하고 있는 반복문

```
for (초기식; 표현식; 증감식) {
    표현식의 결과가 참인 동안 반복적으로 실행하고자 하는 실행문;
}
```

입력

```
<!DOCTYPE html>
<html>

<body>

    <h3>for 문</h3>

    <script>
        for (var i = 1; i < 10; i++) {
            document.write(i + "<br>");
        }
    </script>

</body>

</html>
```

결과

for 문

1
2
3
4
5
6
7
8
9

함수와 제어문

2. 제어문

2) 반복문 | (4) for - in 문

- 반복문 루프마다 객체의 열거할 수 있는 프로퍼티(enumerable properties)의 이름을 지정한 변수에 대입

```
for (변수 in 객체) {
    객체의 모든 열거할 수 있는 프로퍼티의 개수만큼 반복적으로 실행하고자 하는 실행문;
}
```

함수와 제어문

2. 제어문

2) 반복문 | (4) for - in 문

입력

```
<!DOCTYPE html>
<html>

<body>

    <h3>for - in 문</h3>

    <script>
        var arr = [1, 2, 3];

        for (var i = 0; i < arr.length; i++) {
            document.write(i + " ");
        }
        document.write("<br>");

        for (var i in arr) {
            document.write(i + " ");
        }
    </script>

</body>

</html>
```

결과

for - in 문

0 1 2
0 1 2

함수와 제어문

2. 제어문

2) 반복문 | (5) for - of 문

- 반복할 수 있는 객체(iterable objects – Array, Map, Set, arguments)를 순회

```
for (변수 of 객체) {
    객체의 모든 열거할 수 있는 프로퍼티의 개수만큼 반복적으로 실행하고자 하는 실행문;
}
```

입력

```
<!DOCTYPE html>
<html>

<body>

    <h3>for - of 문</h3>

    <script>
        var arr = new Set([1, 1, 2, 2, 3, 3]);

        for (var value of arr) {
            document.write(value + " ");
        }
    </script>

</body>

</html>
```

결과

for - of 문

1 2 3

핵심정리

1. 자바스크립트(JavaScript)

- 개념
  - ✓ 웹페이지를 동적으로 콘텐츠를 바꾸고, 프로그래밍적으로 제어하기 위해 고안된 언어로서 객체기반의 스크립트 언어
- 자바스크립트의 실행문은 세미콜론(;) 구분하며 대소문자를 구분함
- 자바스크립트로 HTML 페이지에 출력을 할 수 있음
- 방법
  - ✓ window.alert()
  - ✓ HTML DOM 요소를 이용한 innerHTML프로퍼티
  - ✓ document.write(),console.log()
- HTML 어디서든 자바스크립트를 사용할 수 있지만 외부 파일로 따로 관리하는 것이 유지보수 및 가독성에 좋음

핵심정리

2. 함수와 제어문

- 자바스크립트에서는 함수도 하나의 데이터 타입이며 함수를 변수에 대입하거나 함수에 프로퍼티를 지정하는 것도 가능함
- function 키워드로 선언하며 중괄호({ }) 안에 자바스크립트 내부 실행문을 넣어서 작성함
- 조건문은 프로그램 내에서 주어진 표현식의 결과에 따라 별도의 명령을 수행하도록 제어함
  - ✓ If 문
  - ✓ else if - else 문
- 반복문은 특정 조건을 만족할 때까지 계속해서 주어진 실행문을 반복 실행함
  - ✓ while 문
  - ✓ do-while, for 문