

4_기초문법

조건문(if , switch)

if

```
if (condition) {  
    //수행코드  
}
```

- condition이 true 인경우 해당 블록의 명령문 수행

```
let number = 10;  
if (number > 5) {  
    console.log("5 초과 숫자 입니다.");  
}
```

if-else

```
const age = 17;  
const isAdult = false;  
if (age > 18) {  
    isAdult >= true;  
    console.log("18세 이상입니다. ");  
} else {  
    console.log("18세 미만 입니다. ");  
}
```

if...else if...else

```
const hour = 14;  
if (hour < 10) {  
    console.log("굿모닝");  
}
```

```

    } else if (hour < 18) {
        console.log("굿애프터눈");
    } else if (hour < 21) {
        console.log("굿이브닝");
    } else {
        console.log("굿나이트");
    }

```

switch

- 복수의 `if` 조건문은 `switch` 문으로 변경하여 사용 가능
- 특정 변수를 다양한 상황에서 비교할 수 있게 만들어 준다.
- switch 문은 하나의 표현식을 평가하고, 일치하는 항목의 case 절을 실행하는 조건문이다. 일치하는 항목이 없다면 default 절을 실행한다.
- break 키워드를 통해 switch 문을 벗어난다.

```

switch(x) {
  case 'value1': // if (x === 'value1')
    ...
    [break]

  case 'value2': // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}

```

```

let d = new Date();
let day = d.getDay();
// console.log(day);
let dayName = "";

switch (day) {
  case 4:

```

```

    dayName = "목요일";
    break;
case 1:
    dayName = "월요일";
    break;
case 2:
    dayName = "화요일";
    break;

default:
    dayName = "";
    break;
}

```

- break 키워드가 명시되지 않을 경우 switch 문을 벗어나지 못하고 아래의 case와 default 절까지 실행하게 된다.

```

const name = "John";
switch(name) {
    case "John":
        console.log("John");
    case "Jane":
        console.log("Jane");
    default:
        console.log("No name");
}

```

switch문



<https://ko.javascript.info/switch>



JAVASCRIPT.INFO
The Modern JavaScript Tutorial

참고

switch와 if else 중 어떤 것을 써야하는가?

switch와 if else 중 어떤 것을 써야하는가? switch구문은 변수를 입력 받아 미리 정해놓은 여러 값들과의 일치여부를 판단하여 switch문 내의 control flow를 결정한다. if else구문은 boolean의 결과 값을 내놓는 조건

☞ <https://aahc.tistory.com/6>



반복문

for

```
for (initialization; condition; expression) {  
  // code block to be executed  
}
```

- **initialization** : 변수 초기화 (선택, 최초 반복문 진입 시 1회만 실행)
- **condition** : condition 정의 (매 반복 시행 전 평가)
- **expression** : 반복 수행시 변수 증가(감소) 정의 (선택, 매 반복 시행 이후 평가)
- 사용할 변수 정의하고, 변수가 특정 조건에 대해 false가 될 때까지 연산하며 반복하는 반복문이다.
- 반복 가능한 객체(Array, Map, Set, String, TypedArray, arguments 객체 등)를 반복하는 기능 수행을 한다.
- 즉, 객체의 요소를(Data)를 순회하기 위한 구문이다.

```
for (let i = 0; i < 5; i++) {  
  if (i == 3) {  
    //break;  
    continue;  
  }  
  console.log("The number is : " + i);  
}
```

```
// expression2 만 정의 하는 방법
let i = 0;
const len = 10;
let total = 0;
for ( ; i <= len; ) {
    total += i;
    i++;
}
console.log(total);
```

```
const numbers = [43, 23, 5, 7];

const len = numbers.length;

for (let i = 0; i < len; i++) {
    let t = numbers[i];
    console.log(t);
}
```

JavaScript for Loop

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many

 https://www.w3schools.com/js/js_loop_for.asp



for ~ in

- Object의 enumerable한 non-Symbol **key(속성)** 들을 반복하는데 사용된다.
- array 의 경우 index가 반환 된다.

```
for (key in object) {
    // code block to be executed
}
```

```
const numbers = [43, 23, 5, 7];
```

```
for (let index in numbers) {
  console.log(numbers[index]);
}
```

```
const numbers = [43, 23, 5, 7];

for (let index in numbers) {
  console.log(index);
}
```

```
const numbers = {"a":43, "b":23, "c":5, "d":7};

for (let index in numbers) {
  console.log(index);
}
```

JavaScript For In

The JavaScript for in statement loops through the properties of an Object: Try it Yourself " The for in loop iterates over a person object Each iteration returns a key (x) The key is used to access the value
[w3schools.com/js/js_loop_forin.asp](https://www.w3schools.com/js/js_loop_forin.asp)



for ~ of

- **iterable** 한 Object(Array, Map, Set, String, TypedArray, arguments 등을 포함)에 대해 **속성값(value)**을 반복 순회
- `break`, `continue`, `return` 사용 가능

```
const numbers = [43, 23, 5, 7];

for (let num of numbers) {
  console.log(num);
}
```

- object type 은 동작하지 않는다.

```
const numbers = {"a":43, "b":23, "c":5, "d":7};

for (var value of numbers) {
  console.log(value);
}
```

Array.forEach()

- 주어진 함수를 배열 요소들에 대해 반복 작업을 실행한다.

```
const items = [43, 23, 5, 7];
items.forEach(function(item) {
  console.log(item);
});


// arrow 사용
items.forEach(item=> console.log(item));
```

```
const array1 = ['a', 'b', 'c'];
let txt = "";

array1.forEach(myFunction);
function myFunction(value, index, array) {
  txt = txt + value + "<br>";
}

console.log(txt)
```

JavaScript For In

The JavaScript for in statement loops through the properties of an Object: Try it Yourself " The for in loop iterates over a person object Each iteration returns a key (x) The key is used to access the value
 https://www.w3schools.com/js/js_loop_forin.asp



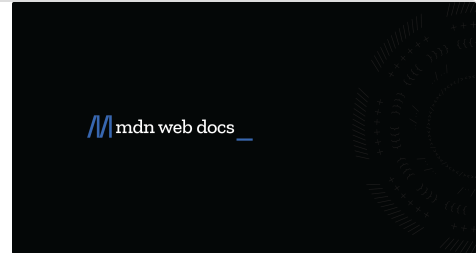
참고

- JavaScript 에서 함수는 왜 밑에서 선언되어도 되는걸까?

호이스팅 - MDN Web Docs 용어 사전: 웹 용어 정의 | MDN

JavaScript에서 호이스팅(hoisting)이란, 인터프리터가 변수와 함수의 메모리 공간을 선언 전에 미리 할당하는 것을 의미합니다. var로 선언한 변수의 경우 호이스팅 시 undefined로 변수를 초기화합니다. 반면 let과

 <https://developer.mozilla.org/ko/docs/Glossary/Hoisting>



while

- condition이 true 인 동안 반복 시행

```
while (condition) {
  // code block to be executed
}
```

```
const i = 1;
let total = 0;
while (i <= 10) {
  total = total + i;
  i++;
}

console.log(total);
```

do~while

- 구문이 실행된 뒤에 테스트 조건이 평가됨으로 구문은 무조건 한 번은 실행

```
do {
  // code block to be executed
```



```
}  
while (condition);
```

```
const i = 11;  
let total = 0;  
  
do {  
    total = total + i;  
    i++;  
} while (i <= 10);  
  
console.log(total);
```

JavaScript while Loop

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many

 https://www.w3schools.com/js/js_loop_while.asp



<https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Statements/do...while>

함수

- JavaScript에서는 2가지의 방식으로 함수를 정의한다.

- 함수 선언식
- 함수 표현식

▼ 참고

- JavaScript의 함수는 일급객체(First-class-citizen)에 해당
- 일급객체 : 다음의 조건들을 만족하는 객체를 의미함
 - 변수에 할당 가능
 - 함수의 매개변수로 전달 가능

- 함수의 반환 값으로 사용 가능

Functions

- Function 선언식

```
function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}
```

```
function add(num1, num2) {  
  return num1 + num2  
}  
  
add(1, 2)
```

- Function 표현식

```
const name = function (args) {  
  // do something  
}
```

```
const add = function (num1, num2){  
  return num1 + num2  
}
```


```
function calc(income) {  
  const vat = 0.1;  
  const tax = income * vat;  
  // console.log(tax);  
  return tax;  
}  
let tax = calc(10);  
console.log(tax);
```

```
function sum(x, y, oper) {
  if (oper == "+") {
    return x + y;
  } else if (oper == "-") {
    return x - y;
  }
}
let plus = sum(5, 7, "+");
console.log(plus);

let minus = sum(5, 7, "-");
console.log(minus);
```

JavaScript Functions


A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it). Try it Yourself " A JavaScript function is defined


 https://www.w3schools.com/js/js_functions.asp



함수 - JavaScript | MDN

보통 함수란 자신의 외부(재귀 함수의 경우 스스로) 코드가 '호출'할 수 있는 하위 프로그램입니다. 프로그램과 마찬가지로, 함수 역시 명령문의 시퀀스로 구성된 함수 본문을 가집니다. 함수에 값을 '전달'하면, 함수는 값을

 <https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Functions>

 mdn web docs

Arrow Function

- 함수를 비교적 간결하게 정의할 수 있는 문법으로 `function` 키워드 생략 가능
- 함수의 parameter가 단 하나 뿐이라면, `()` 도 생략 가능
- 함수 Body가 표현식 하나라면 `{}` 과 `return` 도 생략 가능

```

const arrow1 = function (name) {
  return `hello, ${name}`
}

// 1. function 키워드 삭제
const arrow2 = (name) => { return `hello, ${name}` }

// 2. parameter가 1개일 경우에는 () 생략 가능
const arrow3 = name => { return `hello, ${name}` }

// 3. 함수 body가 return을 포함한 표현식 1개일 경우에 { } & return 삭제 가능
const arrow4 = name => `hello, ${name}`

```

```

//방식 1
function hello() {
  console.log("hello hi?");
  return "hello hi";
}
hello();

//방식 2
const hello = function () {
  console.log("hello hi2");
  return "hello hi2";
};

//방식 3 arrow func
const hello = () => {
  console.log("hello hi3!!");
  return "hello hi3!!!";
};

//1개 표현식인 경우
const hello = () => "hello world";

const hello = (firstName, lastName) =>
"hello " + firstName + " " + lastName;

console.log(hello("hi", "ho"));

```