

웹 앱 개발을 위한 Javascript 기초

3. 자바스크립트 사용자 정의 자료형 활용

학습내용

1. 조건문
2. 반복문

학습목표

1. if 조건문의 기본 사용 방법, if 조건문과 논리 연산자를 함께 사용하는 방법, switch 조건문에 대해 설명할 수 있다.
2. 배열을 생성하고 사용하는 방법, while 반복문과 for, for in 반복문과 for of 반복문, break 키워드와 continue 키워드에 대해 설명할 수 있다.

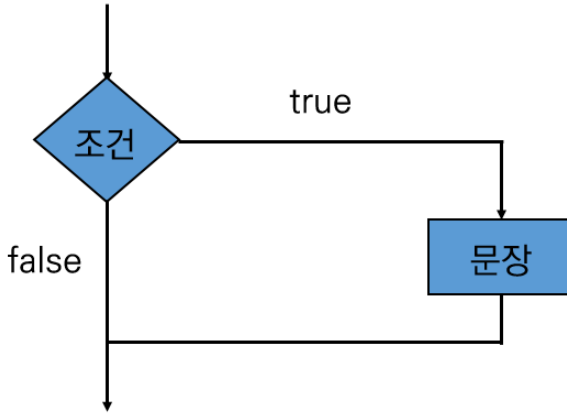
1. 조건문

if 조건문

* 기본 형태

```
if (⟨불 표현식⟩) {  
    
}
```

* 불 표현식이 true이면 문장을 실행, false이면 문장을 무시함



```
let date = new Date();  
if (date.getHours() < 12) {  
  console.log("오전입니다.");  
}  
if (12 <= date.getHours()){  
  console.log("오후입니다.");  
}
```

오후입니다.

undefined

```
> let input = 32;  
if (input % 2 == 0) {  
  console.log("짝수입니다!");  
}  
if (input % 2 == 1) {  
  console.log("홀수입니다!");  
}
```

짝수입니다!

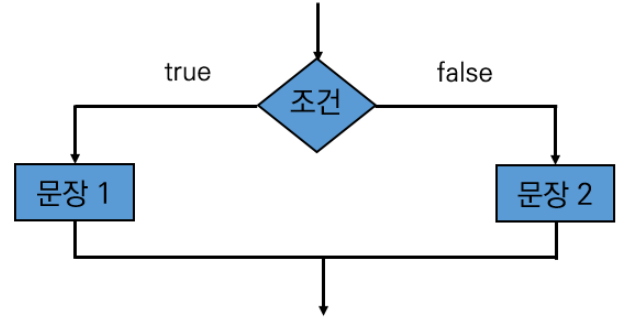
< undefined

1. 조건문

if else 조건문

* 기본 형태

```
if (불 표현식) {
  // 불 표현식이 참이 때 실행할 문장
} else {
  // 불 표현식이 거짓일 때 실행하는 문장
}
```



```
let input = 32;
if (input % 2 == 0){
  console.log("짝수입니다!");
} else {
  console.log("홀수입니다!");
}
짝수입니다!
undefined
```

```
let date = new Date();
if (date.getHours() < 12) {
  console.log("오전입니다.");
} else {
  console.log("오후입니다.");
}
오전입니다.
undefined
```

중첩 조건문

* 기본 형태

```
if (불 표현식) {
  if (불 표현식) {
    문장;
  } else {
    문장;
  }
}
```

```
} else {
  if (불 표현식) {
    문장;
  } else {
    문장;
  }
}
```

1. 조건문

중첩 조건문

- * `DateTime.Now.Hour < 11` 조건을 비교
- * `false`이면 `DateTime.Now.Hour < 15` 조건을 한 번 더 비교

```
let date = new Date();
let hours = date.getHours();
if (hours < 11) {
  console.log("아침 먹을 시간입니다.");
} else {
  if (hours < 15) {
    console.log("점심 먹을 시간입니다.");
  } else {
    console.log("저녁 먹을 시간입니다.");
  }
}
```

아침 먹을 시간입니다.
undefined

if else if 조건문

- * 중복되지 않는 세 가지 이상의 조건을 구분할 때 사용
- * 기본 형태

```
if (⟨불 표현식⟩) {

} else if (⟨불 표현식⟩) {

} else if (⟨불 표현식⟩) {

} else {

}
```

1. 조건문

if else if 조건문

```
let date = new Date ();
let hours = date.getHours();
if (hours < 11) {
  console.log("아침 먹을 시간입니다.");
} else if (hours < 15) {
  console.log("점심 먹을 시간입니다.");
} else {
  console.log("저녁 먹을 시간입니다.");
}
```

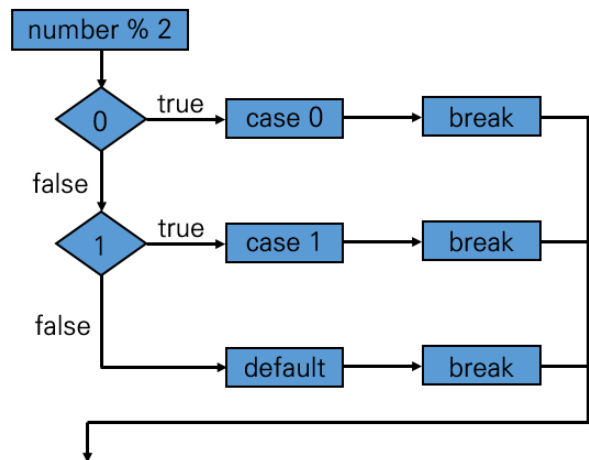
아침 먹을 시간입니다.

undefined

switch 조건문

* 기본 형태

```
switch {<비교할 값>} {
  case <값> :
    <문장>
    break;
  case <값> :
    <문장>
    break;
  default :
    <문장>
    break;
}
```



1. 조건문

switch 조건문

```
let input = 32;
switch (input % 2) {
  case 0:
    console.log("짝수입니다.");
    break;
  case 1 :
    console.log("홀수입니다.");
    break;
}
```

짝수입니다.

undefined

switch 조건문

* break 키워드를 사용하지 않는 switch 조건문


```
let date = new Date( );
switch (date.getMonth( )+1) {
  case 12:
  case 1:
  case 2:
    console.log("겨울입니다.");
    break;
  case 3:
  case 4:
  case 5:
    console.log("봄입니다.");
    break;
  case 6:
  case 7:
  case 8:
    console.log("여름입니다.");
    break;
  case 9:
  case 10:
  case 11:
    console.log("가을입니다.");
    break;
  default:
    console.log("대체 어떤 행성에 살고 계신가요?");
    break;
}
```

겨울입니다.

undefined

2. 연산자

배열

- * 여러 개의 자료를 한꺼번에 다룰 수 있는 자료형
- * 대괄호 내부의 각 자료는 쉼표로 구분
- * 배열에는 여러 자료형이 섞여 있을 수 있음
- * 요소 : 배열 안에 들어 있는 각 자료
- * 배열 선언 형태 Ex) `let 이름 = [자료, 자료, 자료, 자료, 자료]`
- * 배열 요소 Ex) `배열[인덱스]`


배열

- * 인덱스의 시작 숫자는 0임에 주의

```
// 배열을 생성합니다.
let array = [52, 273, '아침밥', '점심밥', true, false]
// 배열의 요소를 변경합니다.
array[0] = 0
// 요소를 출력합니다.
console.log(array[0]);
console.log(array[1]);
console.log(array[2]);
console.log(array[3]);
console.log(array[4]);
```

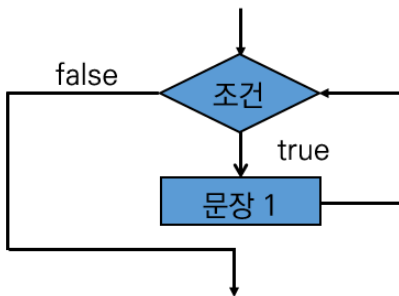
0
273
아침밥
점심밥
true
undefined

2. 연산자

📖 while 반복문

* 기본 형태

```
While (<불 표현식>) {
    // 불 표현식이 참인 동안 실행할 문장
}
```



📖 while 반복문

* 특정한 숫자를 증가시켜 불 표현식을 false로 만들어 반복문을 벗어남

```
// 변수를 선언합니다.
let i = 0;
let array = [52, 273, 32, 65, 103];
// 반복을 수행합니다.
while (i < array.length) {
    // 출력합니다.
    console.log(i + "번째 출력:" + array[i]);
    // 알출하려고 변수를 더합니다.
    i++;
}
0번째 출력:52
1번째 출력:273
2번째 출력:32
3번째 출력:65
4번째 출력:103
4
```

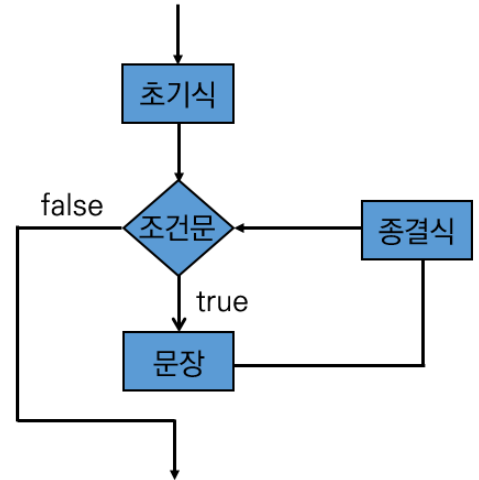
2. 연산자

for 반복문

* 기본 형태

```
for (let i = 0; i < <반복 횟수>>; i++) {  
  }  
}
```

1. 초기식을 비교
2. 조건식을 비교 - 조건이 false이면 반복문을 종료
3. 문장을 실행
4. 종결식을 실행
5. 2단계로 이동



for 반복문

* 0부터 100까지 더하기

```
// 변수를 선언합니다.  
let output = 0;  
// 반복을 수행합니다.  
for (let i = 0; i <= 100; i++) {  
  output += i;  
}  
// 출력합니다.  
console.log(output);  
5050  
undefined
```

2. 연산자

for 반복문

* 1부터 20까지 곱셈하기

```
//변수를 선언합니다.
let output = 1;

//반복을 수행합니다.
for (let i = 1; i <= 20; i++) {
    output *= i;
}

//출력합니다.
console.log(output);
```

```
// 변수를 선언합니다.
let output = 1;
// 반복을 수행합니다.
for (let i = 1; i <= 20; i++){
    output *= i;
}
// 출력합니다.
console.log(output);
2432902008176640000
undefined
```

초깃값을 0으로 놓으면 무엇을 곱해도 0이 됩니다.
따라서 이번에는 1로 설정합니다.

역 for 반복문

* 기본 형태

```
for (let i = length - 1; i >= 0; i--) {
}
```

* 역 for 반복문 - 배열의 요소를 뒤쪽부터 출력

```
//배열을 생성합니다.
let array = [1,2,3,4,5,6];
//요소의 길이를 출력합니다.
for (let i = array.length - 1; i >= 0; i--) {
    console.log(array[i]);
}
6
5
4
3
2
1
undefined
```

2. 연산자

📖 for in 반복문과 for of 반복문

- * 객체에 쉽게 반복문을 적용함
- * for in 반복문과 for of 반복문은 for 반복문 사용과 역할이 같음

```
for (let 인덱스 in 배열) {
}

for (let 요소 of 배열) {
}
```

```
for (let i = 0; i < 배열.길이; i++) {
  let 인덱스 = i;
  let 요소 = 배열[i];
}
```

📖 중첩 반복문

- * 반복문을 여러 번 중첩해서 사용

```
let output = "";
for (let i = 0; i < 10; i++) {
  for (let j = 0; j < i + 1; j++) {
    output += '*';
  }
  output += '\n';
}
console.log(output);

*
**
***
****
*****
*****
*****
*****
*****
*****
undefined
```

2. 연산자

중첩 반복문

- * 반복문을 여러 번 중첩해서 사용

```
> let output = "";
  for (let i = 0; i < 10; i++) {
    for (let j = 0; j < 10 - i; j++) {
      output += ' ';
    }
    for (let j = 0; j < i + 1; j++) {
      output += '*';
    }
    output += '\n';
  }
  console.log(output);

*
**
***
****
*****
*****
*****
*****
*****
*****
< undefined
```

break 키워드

- * 반복문을 벗어날 때 사용
- * 무한 반복문은 내부에서 break 키워드를 사용해야 벗어날 수 있음

```
while (true) {
}
```

2. 연산자

break 키워드

- * 짝수를 찾으면 break 키워드로 반복문을 벗어남

```
let i = 0;
let array = [1, 31, 273, 57, 8, 11, 32];
let output;

while (true){
  if(array[i] % 2 == 0){
    output = array[i];
    break;
  }

  i = i + 1;
}

console.log(`처음 발견한 짝수는 ${output}입니다`);
처음 발견한 짝수는 8입니다
undefined
```

처음 발견한 짝수는 8입니다.

continue 키워드

- * 반복문 내부에서 현재 반복을 멈추고 다음 반복을 진행함
- * 변수 i가 짝수일 때 continue 키워드로 현재 반복을 멈추고 다음 반복을 진행



따라서 코드를 실행하면 홀수만 출력

```
for (let i = 1; i < 10; i++) {
  if (i % 2 == 0) {
    continue;
  }
  console.log(i)
}
1
3
5
7
9
undefined
```

정리하기

1. 조건문

• IF 조건문

```
if (<<불 표현식>>){  
  
}
```

• if else 조건문

```
if (<<불 표현식>>){  
    // 불 표현식이 참이 때 실행할 문장  
} else {  
    // 불 표현식이 거짓일 때 실행하는 문장  
}
```

```
if (불 표현식) {  
    if (불 표현식) {  
        문장  
    } else {  
        문장;  
    }  
}
```

```
} else {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
}
```

• 중첩 조건문

```
if (불 표현식) {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
}
```

```
} else {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
}
```

정리하기

1. 조건문

- switch 조건문

```
if (⟨불 표현식⟩) {

}

case ⟨값⟩ :
    ⟨문장⟩
    break;
default :
    ⟨문장⟩
    break;
}
```

2. 연산자

- 배열 : 여러 개의 자료를 한꺼번에 다룰 수 있는 자료형
- while 반복문 : 특정한 숫자를 증가시켜 불 표현식을 false로 만들어 반복문을 벗어남
- for 반복문 : 초기식·조건식 비교, 조건이 false이면 반복문을 종료, 문장·종결식 실행, 2단계로 이동
- 역 for 반복문 : 배열의 요소를 뒤쪽부터 출력
- for in 반복문과 for of 반복문 : for in 반복문과 for of 반복문은 for 반복문 사용과 역할이 같음
- 중첩 반복문 : 반복문을 여러 번 중첩해서 사용
- break 키워드(반복문을 벗어날 때 사용), continue 키워드(반복문 내부에서 현재 반복을 멈추고 다음 반복을 진행함)