

반복문과 switch문

while

do while

for

switch

반복문(loop)

동일한 코드를 여러 번 반복해야할 때 사용한다
조건(condition)이 참이면 본문의 코드가 실행된다

while



```
1  let i = 3;  
2  while (i) { // i가 0이 되면 조건이 falsy해져 반복을 멈춥니다.  
3      alert( i );  
4      i--;  
5  }
```

while

● ● ● → 조건엔 비교뿐만 아니라 모든 종류의 표현식, 변수 사용가능

```
1 let i = 3;
2 while (i) { // i가 0이 되면 조건이 falsy해져 반복을 멈춥니다.
3     alert( i );
4     i--;
5 }
```


while



반복문 본문이 한 번 실행되는 것 == 반복(iteration, 이터레이션)

```
1  let i = 3;  
2  while (i) { // i가 0이 되면 조건이 falsy해져 반복을 멈춥니다.  
3      alert( i );  
4      i--;  
5  }
```

do while

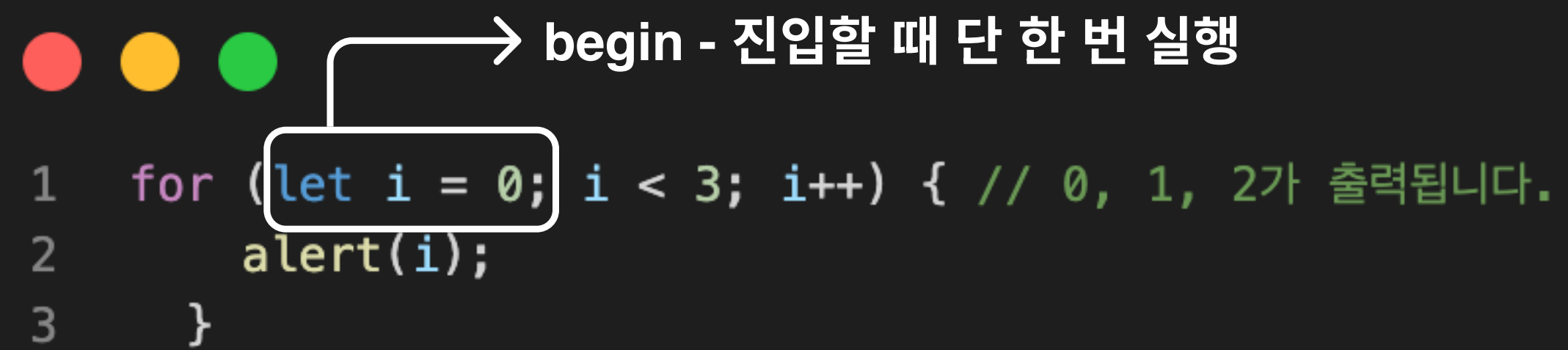


```
1  let i = 3;  
2  do {  
3      alert( i );  
4      i--;  
5  } while (i);
```

조건문이 끝에 가는 do while문

본문이 먼저 실행되고, 조건을 확인한 후
조건이 truthy인 동안엔 본문이 계속 실행된다.

조건에 상관없이, 본문을 최소한 한 번이라도
실행하고 싶을 때만 사용한다.

for

The diagram shows a code editor window with a dark background. At the top left of the editor are three colored circles: red, yellow, and green. To the right of these circles is a white arrow pointing to a white-bordered box that contains the text "begin - 진입할 때 단 한 번 실행". Below this, the code for a for loop is displayed with line numbers 1, 2, and 3 on the left. The code is: `1 for (let i = 0; i < 3; i++) { // 0, 1, 2가 출력됩니다.`, `2 alert(i);`, and `3 }`. The text `let i = 0;` is highlighted in blue, and the text `i < 3; i++` is highlighted in green.

```
1  for (let i = 0; i < 3; i++) { // 0, 1, 2가 출력됩니다.  
2      alert(i);  
3  }
```

for



condition - 반복마다 해당 조건이 확인



```
1  for (let i = 0; i < 3; i++) { // 0, 1, 2가 출력됩니다.  
2      alert(i);  
3  }
```


for



step - 각 반복의 body가 실행된 이후에 실행

```
1  for (let i = 0; i < 3; i++) { // 0, 1, 2가 출력됩니다.  
2      alert(i);  
3  }
```

for

‘인라인’ 변수 선언




```
1  for (let i = 0; i < 3; i++) { // 0, 1, 2가 출력됩니다.  
2      alert(i);  
3  }  
4  
5  alert(i); // Error: i is not defined
```

반복문

- 반복문 본문이 한 번 실행되는 것을 반복(iteration, 이터레이션)이라고 부른다.
- 조건에 상관없이, 본문을 최소한 한 번이라도 실행하고 싶을 때 do while문을 사용한다.

break / continue

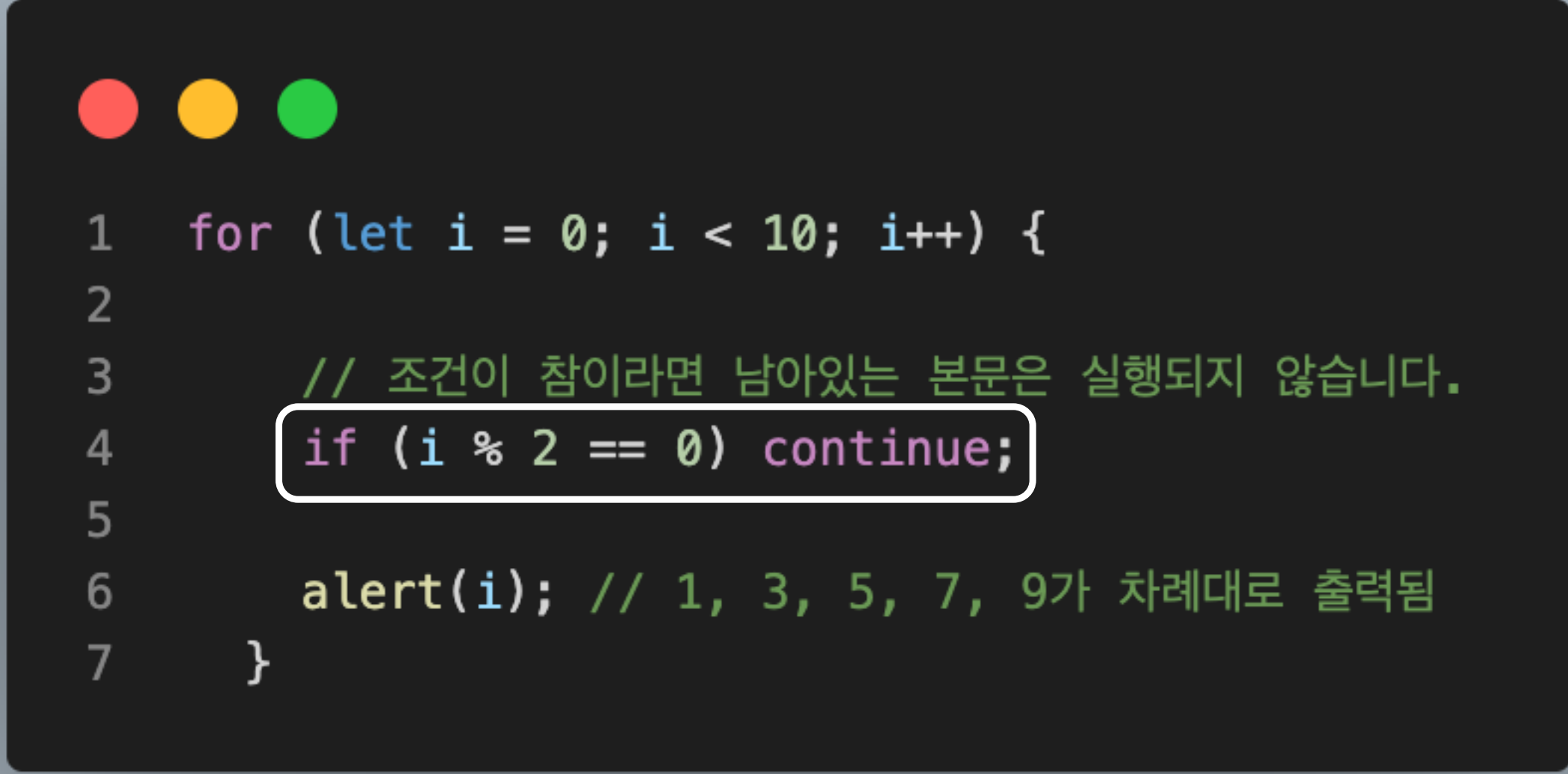


```
1  let sum = 0;
2
3  while (true) {
4      let value = +prompt("숫자를 입력하세요.", '');
5      if (!value) break; // (*)
6      sum += value;
7  }
8  alert('합계: ' + sum);
```

break

조건과 상관없이 break를 사용하면 언제든지 원하는 때에 반복문을 종료시킬 수 있다.

break / continue



```
1  for (let i = 0; i < 10; i++) {  
2  
3      // 조건이 참이라면 남아있는 본문은 실행되지 않습니다.  
4      if (i % 2 == 0) continue;  
5  
6      alert(i); // 1, 3, 5, 7, 9가 차례대로 출력됨  
7  }
```

continue

break의 '가벼운 버전'

break는 전체 반복문을 멈추지만,
continue의 경우 현재 실행 중인 이터레이션을 멈추고
다음 반복으로 넘어간다.

break / continue

삼항연산자에는 사용할 수 없다



```
1  if (i > 5) {  
2      alert(i);  
3  } else {  
4      continue;  
5  }  
6  
7  (i > 5) ? alert(i): continue; // 여기에 continue를 사용하면 안 됩니다.
```

break / continue와 레이블

```
1  outer: for (let i = 0; i < 3; i++) {  
2  
3      for (let j = 0; j < 3; j++) {  
4  
5          let input = prompt(`(${i},${j})의 값`, '');  
6  
7          // 사용자가 아무것도 입력하지 않거나 Cancel 버튼을 누르면 두 반복문 모두를 빠져나옵니다.  
8          if (!input) break outer; // (*)  
9  
10         // 입력받은 값을 가지고 무언가를 함  
11     }  
12 }  
13 alert('완료!');  
14
```

레이블

중첩된 반복문을 한번에 탈출할 수 있는 방법

반복문 앞에 레이블을 붙이고, break/continue에 이 레이블을 함께 사용할 수 있다.

break / continue와 레이블

소수*를 찾는 예제

```
1 nextPrime:
2   for (let i = 2; i <= n; i++) { // 각 i에 대하여 반복문을 돌림
3
4       for (let j = 2; j < i; j++) { // 제수(나눗수)를 찾음
5           if (i % j == 0) continue nextPrime; // 소수가 아니므로 다음 i로 넘어감
6       }
7
8       alert(i); // 소수
9   }
```

* 1과 그 수 자신 이외의 자연수로는 나눌 수 없는 자연수

break / continue와 레이블

- break와 continue를 통해 조건이 참이더라도 반복문을 탈출할 수 있다.
- 반복문 앞에 레이블을 이용해 여러 개의 중첩 반복문을 한 번에 빠져나올 수 있다.


switch문

복수의 if 조건문을 switch문으로 바꿀 수 있다

가독성을 높다는 장점이 있다

대개 default문도 있지만, 이는 필수는 X

switch



```
1  let a = 2 + 2;
2  switch (a) {
3      case 3:
4          alert( '비교하려는 값보다 작습니다.' );
5          break;
6      case 4:
7          alert( '비교하려는 값과 일치합니다.' );
8          break;
9      case 5:
10         alert( '비교하려는 값보다 큼니다.' );
11         break;
12     default:
13         alert( "어떤 값인지 파악이 되지 않습니다." );
14 }
```

switch

```
1 let arg = prompt("값을 입력해주세요.");
2 switch (arg) {
3     case '0':
4     case '1':
5         alert('0이나 1을 입력하셨습니다. ');
6         break;
7
8     case '2':
9         alert('2를 입력하셨습니다. ');
10        break;
11
12    case 3:
13        alert('이 코드는 절대 실행되지 않습니다! ');
14        break;
15    default:
16        alert('알 수 없는 값을 입력하셨습니다. ');
17 }
```

여러 개의 case문 묶기

case '0'이 참인 경우에 아래줄 코드인 case '1'이 실행되기 때문에 case '0'과 '1' 동일한 메시지를 보여준다

switch

```
1  let arg = prompt("값을 입력해주세요.");
2  switch (arg) {
3      case '0':
4      case '1':
5          alert('0이나 1을 입력하셨습니다. ');
6          break;
7
8      case '2':
9          alert('2를 입력하셨습니다. ');
10         break;
11
12     case 3:
13         alert('이 코드는 절대 실행되지 않습니다! ');
14         break;
15     default:
16         alert('알 수 없는 값을 입력하셨습니다. ');
17 }
```

자료형의 중요성

비교하려는 값과 case문의 값의 형과 값이 같아야 해당 case문이 실행된다.

이때 prompt에 3을 입력하면 어떤 창이 뜰까?

switch

```
1  let arg = prompt("값을 입력해주세요.");
2  switch (arg) {
3      case '0':
4      case '1':
5          alert('0이나 1을 입력하셨습니다. ');
6          break;
7
8      case '2':
9          alert('2를 입력하셨습니다. ');
10         break;
11
12     case 3:
13         alert('이 코드는 절대 실행되지 않습니다! ');
14         break;
15     default:
16         alert('알 수 없는 값을 입력하셨습니다. ');
17 }
```

자료형의 중요성

3을 입력하더라도

case 3:에 해당하는 세 번째 alert문은 실행되지 않는다!

default에 해당하는 알 수 없는 값을 입력하셨습니다.는 창이 뜬다

prompt 함수 - 필드에 기재한 값을 **문자열**로 변환해 반환

3과 '3'은 다르다

switch

- switch는 가독성을 높인 조건문.
- break와 실행코드를 작성하지 않음으로 여러 case문을 묶을 수 있다.
- case문의 인수에서 자료형을 구분한다.

THANK YOU