지난 시간 프리뷰 퀴즈

변수란 ?

변수를 만드는 방법은 ?

선언 이란 ?

할당 이란 ?

재 할당이란 ?

상수와 변수의 차이는 ?

문자열을 만드는 방법 3가지 .

불리언이란 ?

대표적인 식별자 표기법2가지를 설명 해보시오.

문자열 ‘10’을 담은 변수와 숫자 10을 담은 변수 두가지를 만들어 보시요.

조건문

지난주에는 변수를 만들어보고, 짤막하게 나마 자바스크립트의 콘솔.로그 기능을 통하여 변수를 사용 해 보았습니다. 오늘부터는 조건문을 배우고 조건에 따라 웹 페이지를 동적으로 변화 시키는 행위들을 해보도록 하겠습니다.

먼저 조건 문을 배우기 전 익혔던 불리언이라는 타입을 기억하시나요 ? 불리언은 어떤 논리식이나 비교식, 그리고 값의 결과를 true , false로 나타내주는 데이터 타입입니다. 예를 들어 부등호를 사용하여 console.log(10 > 20)를 해보면 우리의 콘솔에는 거짓이 찍히게 되죠. 반대로 부등호를 < bigger than left를 하게 되면 true라고 우리의 콘솔창에 출력되게 됩니다.

부등호처럼 왼쪽항과 오른쪽항의 비교를 하여 식을 만드는 것을 프로그래밍언어에서 비교 연산자라고 합니다. 비교 연산자에는 어떤 항이 더 크다는 > , < 부등호가 있고, 같음을 표시하기위한 == 동등 연산자와 === 일치 연산자가 있습니다. 그리고 크거나 같음을 표시하기 위한 >= . <= 같은 표시 형식도 있구요.

10 > 20

30 < 50

10 >= 10

20 >= 30

40 == 40

50 === 50

그렇다면 왜 비교연산자에서 동등하다는 표현을 = 한개로 표현하여 사용하지 않을까요 ? 그 이유는 이퀄이 한 개 있으면, 왼쪽 변수에 오른쪽 값을 대입한다는 대입연산자의 역할이 이미 있기 때문입니다.

그렇다면 왼쪽과 오른쪽 항이 일치한다는 == 표시는 이해 하겠는데, ===표시는 무엇인가요 ? == 동등연산자는 10 == ’10’ 처럼 문자열과 숫자 타입이 값이 같을 경우 이를 참으로 해석합니다. 그러나 일치연산자는 왼쪽항과 오른쪽항의 데이터 타입이 일치하고, 값이 같을 경우에만 참으로 표시합니다.

다음 식들의 결과를 참과 거짓으로 말해보시오

10 > 20

150 < 200

120 >= 120

70 <= 75

10 == 10

250 === ‘250’

‘170’ === ‘170’

‘안녕하세요’ == ‘안녕하세요’

우리는 지금 까지 비교 연산자를 통하여, 비교식을 불리언으로 변환하는 과정을 익혀보았습니다.

이제 불리언 타입의 특성을 사용하여 우리는 조건문이라는 것을 만들어 우리의 프로그래밍을 조금 더 다이나믹하고 역동적으로 만들어 주는 것을 연습하고 익혀 봅시다.

조건문은 조건에 따라 순차적인 프로그래밍의 흐름을 바꾸어주는 문장입니다. 사람에 따라 조건문, 제어문, 분기문으로 불리우는데 거의 비슷한 말이니 우리는 조건문이라고 합시다. 자바스크립트를 포함한 다른 프로그래밍에서도 조건문의 사용방법은 비슷하며, 조건문은 if문과 switch문, 삼항연산자가 존재합니다.

조건문의 사용방법은 매우 간단하며, 효과는 정말 엄청납니다.

먼저 if문의 경우 다음과 같이 작성합니다.

If문을 사용하기위해 if를 선언해주시고, 옆에 소괄호에는 조건을 작성해 줍니다. 그리고 중괄호 블럭을 열고 이 곳에는 조건문의 조건이 참일 경우 실행할 코드를 작성해줍니다.

If(조건){

실행될 코드

}

그럼 이제 조건문을 만들어보고 실행해보도록 할까요 ? 만약 프로그래머가 숫자를 담은 a와 b 변수의 값을 비교하여, a가 크다면 , a는 b보다 큽니다. 라는 조건문을 작성하고 싶으면 다음과 같이 작성하면 됩니다 .

Const A = 20;

Const B = 10;

If(A > B){

Console.log(` A 변수 : ${A}는 B 변수 : ${B}보다 큽니다. `);

}

그리고 노드로 파일을 실행시켜 보면 우리가 원하는 대로 조건문이 동작하게 됩니다. If문을 한번 소리내어 읽어 보도록 하겠습니다. ‘만약 A가 B보다 크다면 콘솔로그에 다음과 같은 문장을 출력하라. ‘ 라고 읽으면 되겠습니다. 그렇다면 만약 A가 B보다 크지 않은 상황에서는 어떻게 될까요?

A와 B의 값을 뒤 바꾸어 실행해보면 조건문의 조건식에서 참이 되지 않기 때문에 실행이 되지 않습니다. 만약 이 같이 조건식에 있는 조건이 참이 아닌 경우 다른 코드를 실행하고 싶으면 또 조건문을 만들고 조건식을 새롭게 정의 할 것 같습니다. 그러나 조건문에서 조건식이 참이 아닌 경우에 다른 코드를 실행하고 싶을 경우를 대비하여 else문과 else if문이 존재합니다.

If(A > B){

Console.log(` A 변수 : ${A}는 B 변수 : ${B}보다 큽니다. `);

}

이 식에서 만약 B가 A보다 클 경우 ‘B변수는 A변수보다 큽니다’ 를 출력하고 싶으면 if문 블럭의 말미에다가 else{}를 붙여 블럭안에 해당 조건식이 아닐 경우 실행하고 싶은 코드를 작성하면 됩니다.

If(A > B){

Console.log(` A 변수 : ${A}는 B 변수 : ${B}보다 큽니다. `);

}else{

Console.log(` B 변수 : ${B}는 A 변수 : ${A}보다 큽니다. `);

}

라고 작성하면 A가 B보다 큰 경우에는 첫번째 코드가 실행될 것이고, 아니라면 두번째 코드가 실행되게 됩니다. 하지만 아직 이 조건문에는 한가지 에러가 있죠. 그것은 A와 B가 같을 경우에도 2번째 조건문이 실행된다는 것입니다.

Else if문을 쓰면 처음 조건이 아닌 다른 조건에 반응하여 해당 블럭의 코드를 실행 시킬 수 있습니다. If문 블럭이나 else if문의 블럭 말미에 else if를 적고 소괄호 안에 조건을 작성합니다. 조건을 작성한 후 블럭을 열고 닫아 주면 해당 조건이 만족 했을 때 해당 블럭의 코드가 실행되게 됩니다.

If(A > B){

Console.log(` A 변수 : ${A}는 B 변수 : ${B}보다 큽니다. `);

}else if(A == B){

Console.log(` A 변수 : ${A}와 B 변수 : ${B}는 같습니다.. `);

}else{

Console.log(` B 변수 : ${B}는 A 변수 : ${A}보다 큽니다. `);

}

현재 까지는 우항과 좌황의 값들을 비교하는 간단한 비교문만 작성했는데, 조건을 조금 더 세밀하게 할 수 있는 논리연산자라는 것들이 있습니다. 논리연산자에는 AND 연산 OR 연산이 있고 부정 연산자가 있습니다.

AND연산은 && 표시를 사용하여 좌항과 우항의 식이 둘다 참일 때 이 식의 결과를 참으로 인식 하는 연산입니다. 사용은 다음과 같습니다. && 을 적고 왼쪽과 오른쪽에 비교식을 적으면 끝입니다. 예를 들어 A < B && A < 15 이 논리식의 결과는 참입니다. 그렇지만 A > B && A < 15 는 거짓입니다. 이유는 첫번째 논리식에서 &&의 양 옆의 비교식이 참을 반환하니 이 and 연산의 결과는 참이지만 , A > B && A < 15는 왼쪽 비교식이 A > B 거짓이기에 이 논리식은 오른쪽 항이 참이더라도 결과는 거짓이게 됩니다.

Or연산은 AND연산과 달리 왼쪽식과 오른쪽 식중 하나만 참이면 이 논리식의 결과를 참으로 반환하는 논리연산자 입니다. || 표시를 사용하며 AND연산과 나머지는 동일합니다. A > B || A < 15 를 작성하였을 경우 왼쪽 항은 거짓이지만, 오른쪽 항이 참이기에 이 OR 연산은 참이 됩니다. 만약 두개가 전부 거짓이라면 OR연산은 거짓을 반환합니다.

그리고 != , !== 같은 부정연산자도 있는데, 이 부정연산자는 비교식이 거짓이라면 참으로 뒤바꾸어 줍니다. 예를 들어 10 != 20 은 10과 20이 같지 않기 때문에 참으로 불리언의 값이 리턴됩니다.

다음은 switch문입니다. Switch문은 if문과 같이 조건에 따라 다른 코드를 실행하는 조건문 입니다.

다만 switch문의 특성 때문에 처음 사용 할 때는 조금 어려울 수도 있습니다.

Switch문을 작성할 때는 다음과 같이 작성합니다. Switch문을 선언하고, if문과 같이 소괄호를 열고, 소괄호 안에는 조건 혹은 조건식이 들어갑니다. 그리고 블럭을 열어줍니다.

그리고 블럭내부에 분기를 결정 할 else if대신 case 라는 문장이 등장합니다. Case 옆에는 조건 혹은 조건식의 결과를 적고 콜론을 붙인 다음 실행할 코드를 작성해 줍니다. 실행할 코드는 블럭으로 감싸도 되고 안감싸도 됩니다. 대신 실행할 코드 끝에는 항상 break를 작성해야 하는데, 이유는 후에 알려드리도록 하겠습니다. 그리고 작성한 case에 해당하지 않는 경우 실행할 else와 비슷한 default라는 문장도 있습니다.

그렇다면 A의 값에 따라 다르게 작동하는 switch문을 작성해보도록 하겠습니다.

const A = 10;

const B = 20;

switch(A){

case 10 :

{

console.log(`A는 10입니다.`);

break;

}

case 20 :

{

console.log('A는 20입니다.');

break;

}

default :

console.log('에러가 발생하였습니다.');

break;

}

결과는 a는 10입니다. 가 출력되게 됩니다. 이 스위치문은 소괄호 안에 들어온 조건 혹은 조건식이 case에 명시되어있는 조건에 부합하면 해당 케이스의 블럭을 실행하는 제어문입니다. 그렇기 때문에 case A < B와 같은 식을 작성 할 수 없고 , 조건에 따른 결과를 case에 열거하여 사용하여야 합니다.

const A = 10;

const B = 20;

switch(A,B){

case A < B :

{

console.log(`A는 10입니다.`);

break;

}

case A> B :

{

console.log('A는 20입니다.');

break;

}

default :

console.log('에러가 발생하였습니다.');

break;

}

그리고 switch문을 살펴보면 break라는 단어가 있는데, 만약 break를 안 쓴다면 해당 케이스에 걸리고 그 아래에 있는 모든 케이스들을 실행하게 됩니다.

If문의 비해 조금 사용이 까다로워 왜 이런것을 만들었나 싶겠지만 switch문을 사용하는 경우 코드가 간결 해지며, if문의 경우 모든 조건식을 다 대조해봐야 하는 단점이 있지만, switch문의 경우 조건에 맞으면 case를 실행하고 조건문을 바로 끝내기 때문에 자원관리에 조금 더 효율적입니다.

마지막으로 삼항연산자입니다. 삼항 연산자는 간단한 식의 결과로 짧은 코드를 한 줄에 작성하기 위해서 사용하는 문법입니다.

사용은 매우 간단합니다. 조건 혹은 조건식을 적고 ? 물음표를 붙여줍니다. 그리고 물음표 바로 오른쪽에는 조건식이 참일 경우 실행할 코드를 적고 콜론을 붙여줍니다. 콜론 옆에는 조건이 거짓일 때 사용할 코드를 적으시면 됩니다.

A > B ? console.log(A는 B보다 큽니다.) : console.log(A는 B보다 작거나 같습니다.);

오늘은 3가지 조건문과 비교연산자, 논리연산자, 부정연산자에 대해서 배워 보았습니다.