

[실전 1] 숫자 맞추기 게임

숫자 맞추기 게임은 컴퓨터가 만든 숫자를 사용자가 범위를 좁혀가면서 맞추는 게임입니다. 아마 친구들과 모여서 한 사람이 제시한 숫자를 나머지 친구들이 차례로 돌아가며 맞추는 (일명 up & down 게임) 게임을 해 본 적도 있을 것입니다.

프로그램 흐름 미리 그려보기

1. 빈 노트를 꺼내놓고 게임이 어떤 식으로 진행될지를 단계별로 미리 그려보겠습니다. 우선 컴퓨터에서 숫자를 정합니다. 이 숫자는 사용자가 알 수 없지만 여기에서는 '23'이라는 숫자를 예로 들어보겠습니다.

1	컴퓨터가 숫자 정함	예) 23
---	------------	-------

2. 이번에는 사용자가 숫자를 제시합니다. 숫자 범위는 정해주는 게 좋겠죠? 여기에서는 1 에서 100 사이의 숫자만 사용하겠습니다. 예를 들어, '55'라는 숫자를 제시해 보겠습니다.

1	컴퓨터가 숫자 정함	예) 23
2	사용자가 숫자 정함	예) 55

3. 컴퓨터가 정한 숫자와 사용자가 제시한 숫자를 비교해 더 큰지, 더 작은지, 혹은 맞았는지 알려줍니다. 사용자가 제시한 숫자보다 컴퓨터가 정한 숫자가 더 크다면 사용자에게 좀 더 큰 숫자를 제시하라고 'UP'이라고 알려주고, 컴퓨터가 정한 숫자가 더 작다면 좀더 작은 숫자를 제시하라고 'DOWN'이라고 알려주겠습니다. 맞았다면 '맞았습니다'라고 알려주고 게임이 끝납니다. 여기에서는 사용자가 제시한 숫자 '55'를 입력하면 'DOWN'이라고 표시되겠네요.

1	컴퓨터가 숫자 정함	23
2	사용자가 숫자 제시	55
3	컴퓨터 숫자와 사용자 숫자 비교	'55' 입력 결과 : DOWN
	3-1 '컴퓨터 숫자 > 사용자 숫자'라면 'UP' 표시	
	3-2 '컴퓨터 숫자 < 사용자 숫자'라면 'DOWN' 표시	
	3-3 '컴퓨터 숫자 = 사용자 숫자'라면 '맞힘' 표시. 게임 끝.	

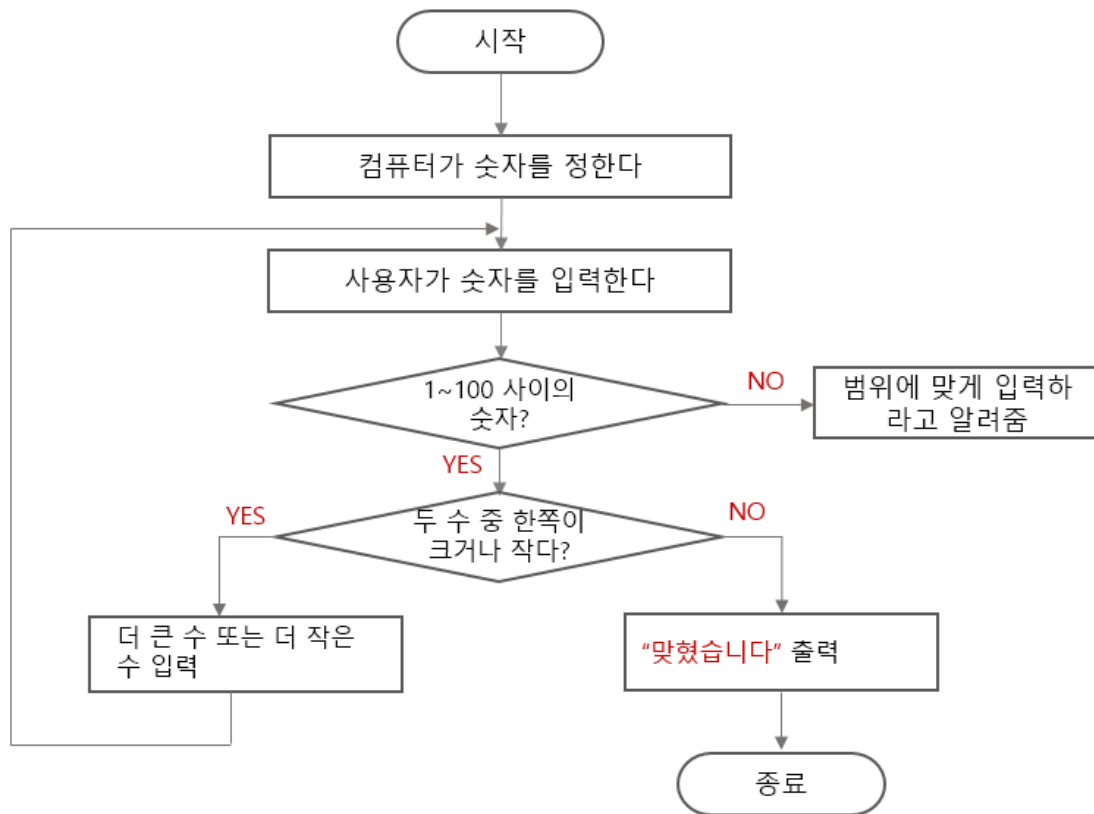
4. 숫자를 맞추지 못했다면 사용자가 다시 다른 숫자를 제시합니다. 우리는 2 번 단계에서 '55'라는 숫자를 제시했었는데, 컴퓨터가 'DOWN'이라고 했으므로 아까 입력했던 값보다 작은 값을 제시합니다. 여기에서는 '10'을 제시해 보겠습니다.

1	컴퓨터가 숫자 정함	23
2	사용자가 숫자 제시	55
3	컴퓨터 숫자와 사용자 숫자 비교	'55' 입력 결과 : DOWN
	3-1 '컴퓨터 숫자 > 사용자 숫자'라면 'UP' 표시	
	3-2 '컴퓨터 숫자 < 사용자 숫자'라면 'DOWN' 표시	
	3-3 '컴퓨터 숫자 = 사용자 숫자'라면 '맞힘' 표시. 게임 끝.	
4	사용자가 숫자 제시	10

5. 다시 한번 컴퓨터 숫자와 사용자 숫자를 비교해서 'UP'인지, 'DOWN'인지, 아니면 맞혔는지 확인합니다. 여기에서는 컴퓨터가 정한 숫자 '23'보다 사용자 숫자가 작으므로 'UP'이 표시되겠네요.

1	컴퓨터가 숫자 정함	23
2	사용자가 숫자 제시	55
3	컴퓨터 숫자와 사용자 숫자 비교	DOWN
	3-1 '컴퓨터 숫자 > 사용자 숫자'라면 'UP' 표시	
	3-2 '컴퓨터 숫자 < 사용자 숫자'라면 'DOWN' 표시	
	3-3 '컴퓨터 숫자 = 사용자 숫자'라면 '맞힘' 표시. 게임 끝.	
4	사용자가 숫자 제시	10
5	컴퓨터 숫자와 사용자 숫자 비교	'10' 입력 결과 : UP
	5-1 '컴퓨터 숫자 > 사용자 숫자'라면 'UP' 표시	
	5-2 '컴퓨터 숫자 < 사용자 숫자'라면 'DOWN' 표시	
	5-3 '컴퓨터 숫자 = 사용자 숫자'라면 '맞힘' 표시. 게임 끝.	

6. 여기까지 하면 2~3 단계와 4~5 단계가 반복되는 걸 알 수 있습니다. 사용자가 컴퓨터 숫자를 맞추기 전까지는 계속 반복할 것입니다. 이것을 순서도로 그려보면 다음과 같이 그릴 수 있습니다. 이 순서도를 참고해 프로그램을 만들어 보세요.



Up & Down 게임 순서도

[해설]

1. 무작위 수 만들기

1. 비주얼 스튜디오 코드에서 upAndDown.html 문서를 불러와서 간단히 소스를 분석해 보겠습니다. 사용자가 숫자를 입력한 후 #check 버튼을 클릭하면 숫자를 비교하고 계산한 후 #display 영역에 결과를 표시합니다. #reset 버튼을 클릭하면 웹 문서를 다시 불러와 처음부터 다시 시작하게 되죠.

```
<div id="wrapper">
  <h1>숫자 맞추기 게임</h1>
  <p>1에서 100 사이의 수를 입력하세요.</p>
  <form action="">
    <label><input type="text" id="try" autocomplete="off" autofocus></label>
    <input type="button" value="확인" id="check" class="btn btn-1">
    <input type="button" value="다시" id="reset" class="btn btn-2">
  </form>
  <div id="display" class="output"></div>
  <div id="counter" class="footer"></div>
</div>
```



2. #check 버튼을 클릭했을 때 숫자를 비교하고 결과를 표시해야 합니다. 우선 #check 부분에 `finding()` 함수를 연결하겠습니다. 그리고, [취소] 버튼을 클릭했을 때는 현재 페이지를 다시 로딩 하도록 할 것입니다. 다음과 같이 두 개의 버튼에 함수를 연결합니다.

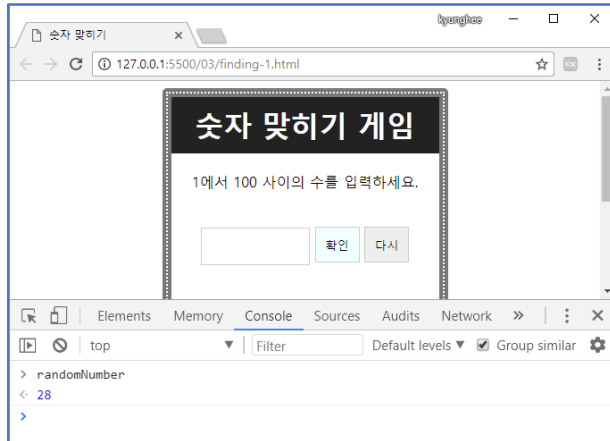
▶ `window.location.reload()` 명령은 현재 문서를 다시 불러오는 함수로 자바스크립트 안에 미리 만들어져 있습니다.

```
<label><input type="text" id="try" autocomplete="off" autofocus></label>
<input type="button" value="확인" id="check" class="btn btn-1" onclick="finding( )">
<input type="button" value="다시" id="reset" class="btn btn-2"
onclick="window.location.reload()">
```

3. js 폴더에 `upanddown.js` 파일을 만들고 `upAndDown.html` 문서에 연결합니다. 이제부터 `upanddown.js` 파일에 소스를 작성해 보겠습니다. 우선 몇 번 만에 숫자를 맞히는지 확인하기 위해 `count` 라는 변수를 선언하고 초깃값을 0으로 지정합니다. 그리고, 무작위 수를 만들기 위해 자바스크립트 안에 이미 만들어져 있는 `Math.random()` 이라는 함수를 사용하는데, `Math.random()` 함수 결과값은 0과 1 사이의 숫자이기 때문에 1에서 100 사이의 무작위 수를 구하기 위해 결과값에 100을 곱하고 정수로 바꾼 후 1을 더해 줍니다.

```
var count = 0;
var randomNumber = Math.floor(Math.random()*100) + 1;
```

4. 문서를 저장한 후 웹 브라우저에서 확인해 볼까요? [Ctrl + Shift + J]를 눌러 콘솔 창을 엽니다. 그리고 randomNumber라고 입력해 보세요. 1과 100 사이의 무작위 수가 표시됩니다.



2. 사용자가 입력한 숫자 가져오기

컴퓨터에서 무작위 수를 만들었으니 이제 사용자가 숫자를 입력할 차례군요. 사용자가 텍스트 필드에 값을 입력하고 [확인] 버튼을 클릭하면 텍스트 필드의 값을 가져옵니다. 그리고, 이 프로그램은 1에서 100 사이의 숫자만 사용할 것이기 때문에 사용자가 입력한 값이 1과 100 사이의 숫자인지도 확인해 보겠습니다.

1. 계속해서 upanddown.js 파일에 작성합니다. #try 필드에 사용자가 입력한 값을 가져와 userNumber라는 변수에 저장해 보겠습니다. 텍스트 필드의 값을 가져올 때는 텍스트 필드 요소(#try)의 value 속성을 이용합니다. 사용자 입력 값은 [확인] 버튼을 클릭한 후에야 가져올 수 있으므로 finding() 함수 안에 선언합니다.

```
var count = 0;

var randomNumber = Math.floor(Math.random() * 100) + 1;

function finding() {
    var userNumber = document.querySelector('#try').value;
}
```


2. 사용자가 1과 100 사이의 숫자를 입력했는지 확인하려면 if 문과 else 문을 사용합니다. 조건에 맞는다면 if 다음의 명령을 실행하고 조건에 맞지 않는다면 else 다음의 명령을 실행합니다. 제대로 동작하는지 확인하기 위해 if 문에서 console.log를 사용해 사용자 입력 값을 표시해 보겠습니다.

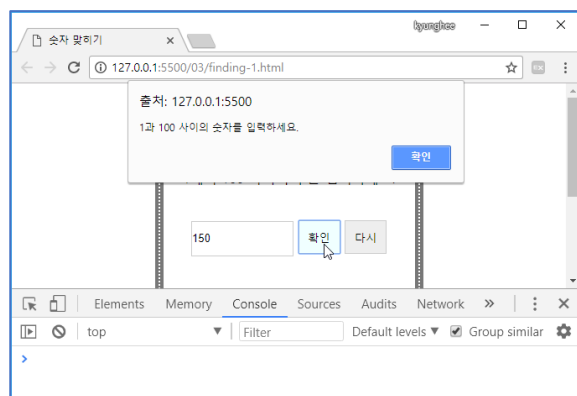
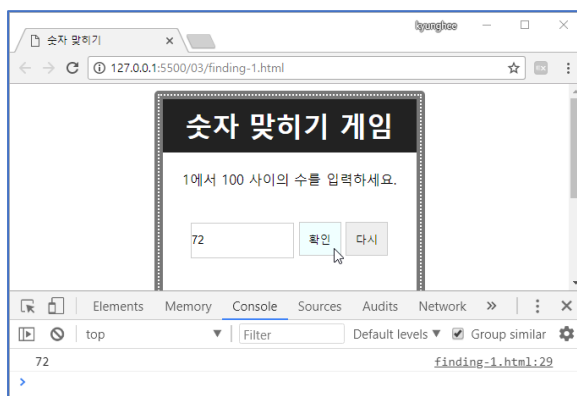
```
var count = 0;

var randomNumber = Math.floor(Math.random()*100) + 1;

function finding() {
  var userNumber = document.querySelector('#try').value;

  if (userNumber >= 1 && userNumber <= 100) {
    console.log(userNumber);
  }
  else {
    alert("1과 100 사이의 숫자를 입력하세요.");
  }
}
```

3. 문서를 저장한 후 웹 브라우저로 확인해 보겠습니다. 텍스트 필드에 1과 100 사이의 아무 숫자나 입력한 후 [확인] 버튼을 클릭하면 콘솔 창에 방금 입력한 숫자가 표시되어야 합니다. 100 이상의 숫자를 입력하면 알림 창이 뜨야 하고요. 혹시 제대로 동작하지 않는다면 지금까지 입력한 소스 중에 오타는 없는지 확인하세요.



3. 사용자 숫자와 컴퓨터 숫자 비교하기

사용자가 입력한 1과 100 사이의 숫자를 가져왔다면 이제 컴퓨터 숫자와 비교해 봐야겠죠? 컴퓨터 숫자보다 사용자 숫자가 작다면 'UP'이라고 표시하고, 컴퓨터 숫자보다 사용자 숫자가 크다면 더 작은 수를 입력하라고 'DOWN'이라고 표시할 것입니다. 컴퓨터 숫자와 사용자 숫자가 같다면 '맞았습니다'라고 표시할 것입니다.

1. 앞에서 if 문 안에 있는 console.log() 문을 삭제합니다. 그리고 그 자리에 다음과 같은 소스를 추가합니다. 조건을 체크할 것이므로 if...else 문을 사용해야겠죠? 그런데 조건이 하나가 아니라 여러 개입니다. 이럴 경우에는 if...else 문을 중첩해서 사용합니다. 그리고 1과 100 사이의 숫자를 가져와서 처리했다면 count 값도 1 증가시켜 줍니다.

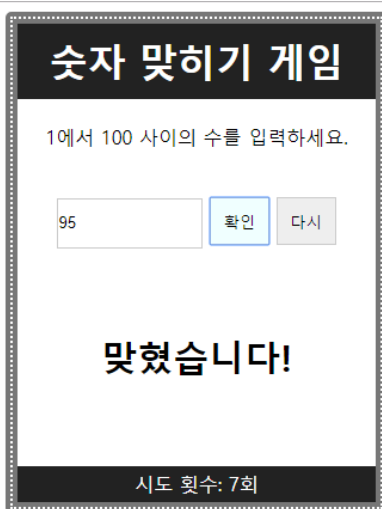
```
function finding() {  
    var userNumber = document.querySelector('#try').value;  
  
    if (userNumber >=1 && userNumber <= 100) {  
        if (randomNumber > userNumber) { // 컴퓨터 숫자가 클 경우 - 'UP' 표시  
  
        }  
        else if (randomNumber < userNumber) { //컴퓨터 숫자가 작을 경우 - 'DOWN' 표시  
  
        }  
        else { // 컴퓨터 숫자를 맞혔을 경우  
  
        }  
        count++;  
    }  
    else {  
        alert("1과 100 사이의 숫자를 입력하세요.");  
    }  
}
```

2. 이제 조건에 따라 그 결과를 화면에 표시해 보겠습니다. HTML 소스를 보면 #display인 요소에 결과를 표시해야겠군요. 앞에서 입력한 if 문과 else 문에 다음과 같은 소스를 추가합니다.

[팁] 화면에 표시할 내용이 텍스트뿐이라면 innerHTML 대신 innerText 메서드를 사용해도 됩니다.

```
function finding() {  
    var userNumber = document.querySelector('#try').value;  
  
    if (userNumber >=1 && userNumber <= 100) {  
        if (randomNumber > userNumber) {  
            document.querySelector('#display').innerHTML = "UP!"; // #display 영역에 'UP!' 표시  
        }  
        else if (randomNumber < userNumber) {  
            document.querySelector('#display').innerHTML = "DOWN!"; // #display 영역에 'DOWN!' 표시  
        }  
        else {  
            document.querySelector('#display').innerHTML = "<span style='color:red'>맞혔습니  
다!</span>"; // #display 영역에 빨간색으로 '맞혔습니다!' 표시  
        }  
        count++;  
        document.querySelector('#counter').innerHTML = "시도 횟수: "+ count + "회"; //  
#counter 영역에 시도 횟수(count) 표시  
    }  
    else {  
        alert("1과 100 사이의 숫자를 입력하세요.");  
    }  
}
```

3. 문서를 저장한 후 웹 브라우저에서 확인해 보세요. 숫자를 맞힐 때까지 다른 숫자를 입력하면서 게임을 진행할 수 있습니다. 검정색으로 표시된 영역에는 시도 횟수가 표시됩니다. 그런데 텍스트 필드에 숫자를 입력하고 [확인]을 클릭해도 입력했던 숫자가 지워지지 않기 때문에 새로운 숫자를 입력하려면 기존에 입력했던 숫자를 직접 삭제한 후 입력해야 합니다. 너무 번거롭지요? 소스를 좀 수정해야겠네요.



4. 사용자 숫자를 가져와 크기를 비교한 후에 그 결과를 화면에 표시했다면 사용자가 다시 숫자를 입력할 수 있도록 텍스트 필드를 지워야 합니다. else 문 다음에 다음과 같은 소스를 추가합니다. 소스 추가가 끝나면 [Ctrl+S]를 눌러 소스를 저장합니다.

```
else {  
    document.querySelector('#display').innerHTML = "<span style='color:red'>맞혔습니다!</span>";  
}  
document.querySelector('#try').value = "";  
count++;  
document.querySelector('#counter').innerHTML = "시도 횟수: " + count + "회";
```

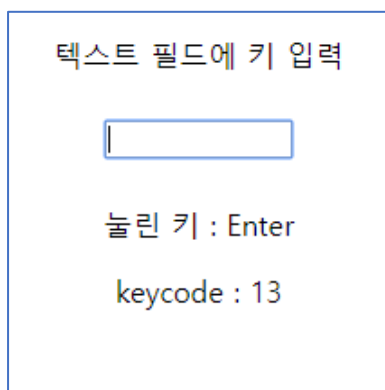
5. 웹 브라우저에서 다시 한번 확인해 볼까요? 사용자가 입력한 숫자와 컴퓨터 숫자를 비교한 결과가 화면에 표시되면서 텍스트 필드는 다음 숫자를 입력할 수 있게 비워질 것입니다. 숫자 맞추기 게임의 기본 기능이 완성되었습니다.

The screenshot shows a web browser window with a game interface. At the top, a dark header contains the title '숫자 맞추기 게임' in white. Below the header, a light gray area contains the instruction '1에서 100 사이의 수를 입력하세요.' in black. Underneath the instruction is a text input field with a blue border, followed by two buttons: a green '확인' button and a gray '다시' button. In the center of the page, the word 'DOWN!' is displayed in large, bold, black letters. At the bottom, a dark footer contains the text '시도 횟수: 3회' in white.

4. [Enter] 키 눌러서 함수 실행하기

지금까지는 [확인] 버튼을 눌러야 finding() 함수를 실행할 수 있었습니다. 키보드에서 값을 입력하고 마우스로 손을 옮겨 [확인] 버튼을 눌러야 하기 때문에 약간 번거롭습니다. 지금처럼 [확인] 버튼을 눌러 실행할 수도 있지만, 마우스까지 손을 뻗지 않고 키보드의 [Enter] 키를 눌러 함수를 실행한다면 좀 더 편리할 것입니다. 자바스크립트에서 [Enter] 키를 알아내는 방법을 알아보겠습니다.

1. 웹 브라우저에서 upAndDownWkeycode.html 문서를 열고 텍스트 필드에 키보드의 아무 키나 눌러보세요. 어떤 키가 눌렸는지, 그리고 키코드 값은 얼마인지 표시됩니다. 예를 들어, [Enter] 키를 누르면 키코드가 '13'이라고 표시됩니다.



2. 웹 문서에서 마우스나 키보드를 조작하거나 웹 문서를 불러오는 등 웹 문서에서 발생하는 모든 동작을 '이벤트(event)'라고 하는데, 키보드의 키를 눌렀을 때는 keypress라는 이벤트가 발생합니다. #try 요소에 keypress 이벤트가 발생했을 때(즉, 텍스트 필드에서 어떤 키를 눌렀을 때) 함수를 실행하려면 다음과 같이 추가합니다. 이 소스는 finding() 함수를 선언하기 전에 추가합니다.

```
var count = 0;
var randomNumber = Math.floor(Math.random()*100) + 1;

document.querySelector('#try').onkeypress = function(e) {
}
}
```

3. 눌린 키의 키코드 값이 '13'인지 확인하고 함수를 실행해야겠죠? 키코드 값은 `event.keycode` 속성을 가져오면 되는데, 여기에서는 이벤트 `e`로 표기했으므로 `e.keycode` 값을 가져와 확인하면 되겠네요. 그리고 파이어폭스의 경우 `keypress` 이벤트에서 `e.keycode` 대신 `e.which` 속성을 사용해야 키코드 값을 알 수 있기 때문에 두 가지 속성을 함께 체크합니다. 다음과 같은 소스를 추가하세요. [Ctrl+S]를 눌러 문서를 저장합니다.

```
var count = 0;

var randomNumber = Math.floor(Math.random()*100) + 1;

document.querySelector('#try').onkeypress = function(e) {
  if (e.keycode == 13 || e.which == 3) { // 눌린 키가 [Enter]키라면
    finding(); // finding() 함수 실행
    return false; // keypress 이벤트가 발생했을 때 브라우저가 기본으로 하는 동작 취소
  }
}
```

4. 웹 브라우저에서 다시 확인해 보겠습니다. 텍스트 필드에 숫자를 입력하고 [Enter] 키를 눌러보세요. [확인]을 누르지 않더라도 입력한 숫자를 체크해서 'UP'인지 'DOWN'인지 표시됩니다.