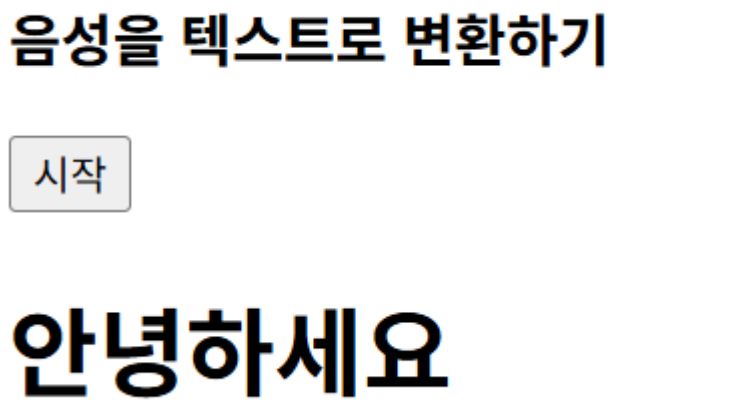
#1 음성인식 시스템 수정

음성을 텍스트로 바꿔주는 프로그램이 개발되었는데,
고객님께서 사용하시다가 추가 요구사항이 들어왔습니다.
요구사항을 모두 반영하세요. "고객 감동!!"

음성인식원본.html (1)

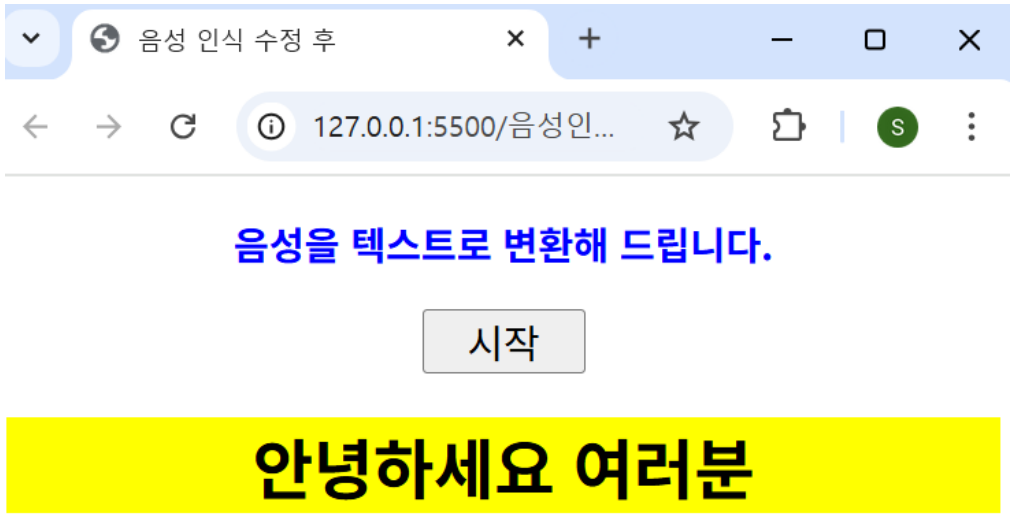
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"> <title>음성 인식 원본</title>
</head>
<body>
    <h3>음성을 텍스트로 변환하기</h3>
    <button id="startButton">시작</button>
    <h1 id="output">여기에 인식된 텍스트가 표시됩니다.</h1>

    <script>
        const output = document.getElementById("output");

        // 음성 인식을 지원하는지 확인
        const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
        if (!SpeechRecognition) {
            alert("이 브라우저는 음성 인식을 지원하지 않습니다.");
        }
```



음성인식원본.html (2)

```
    else {
        const recognition = new SpeechRecognition(); /* SpeechRecognition 객체 생성 */
        recognition.lang = 'ko-KR'; // 언어 설정

        document.getElementById("startButton").onclick = () => {     /* 클릭이벤트 */
            recognition.start();
        };

        recognition.onresult = (event) => {    /* 음성 인식 결과 보여줌  */
          output.textContent = event.results[0][0].transcript;
        };

        recognition.onerror = (event) => {   /* 오류 처리 */
            console.error("오류가 발생했습니다:", event.error);
            output.textContent = "오류가 발생했습니다: " + event.error;
        };

        recognition.onend = () => {     /* 인식 끝날 때마다 */
            console.log("음성 인식이 종료되었습니다.");
        };
    }
  </script>
</body></html>
```

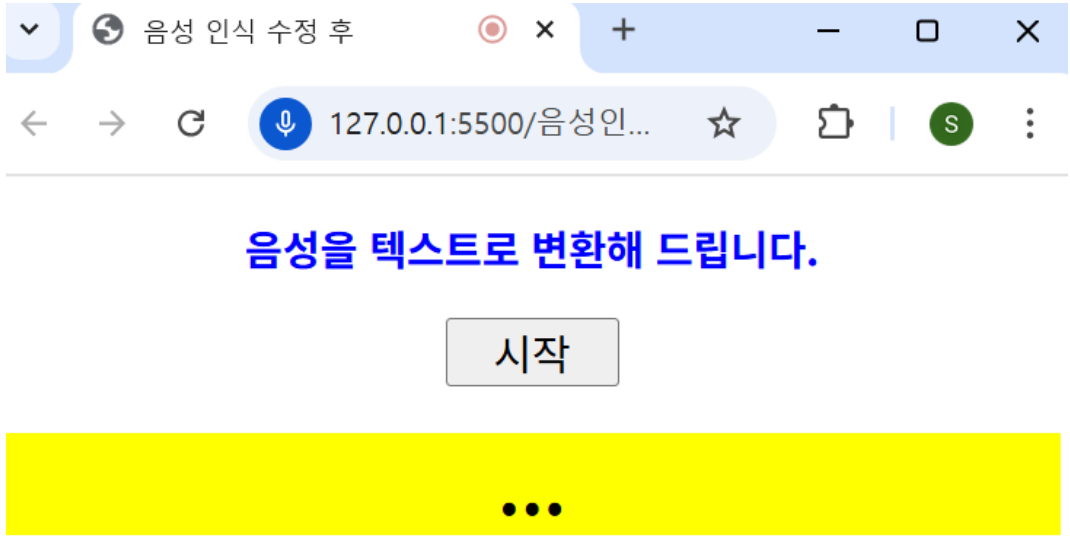
고객 요구 사항 (1)
- 화면 디자인



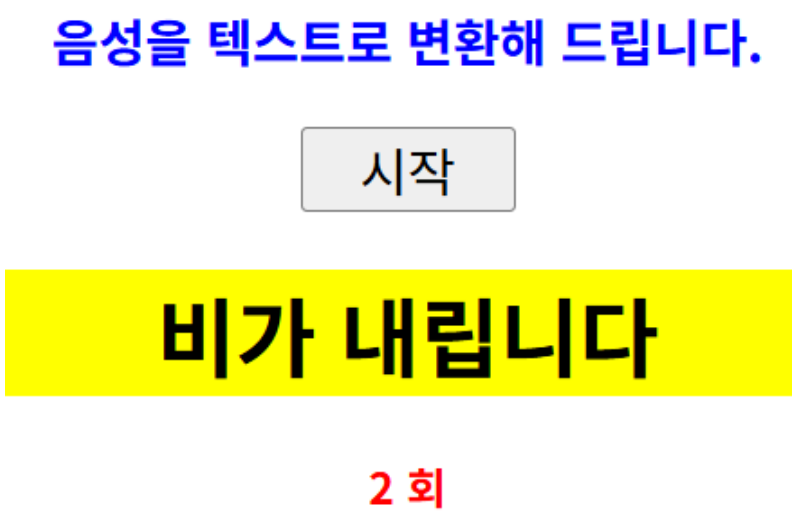
고객 요구 사항 (2)

- 말하는 도중에는 점 세 개가 나오도록 수정

(원본은 말하는 도중에 노란 라인에 이전 텍스트가 그대로 남아있음)



고객 요구 사항 (3)
- 몇 번 수행했는지 카운트 출력



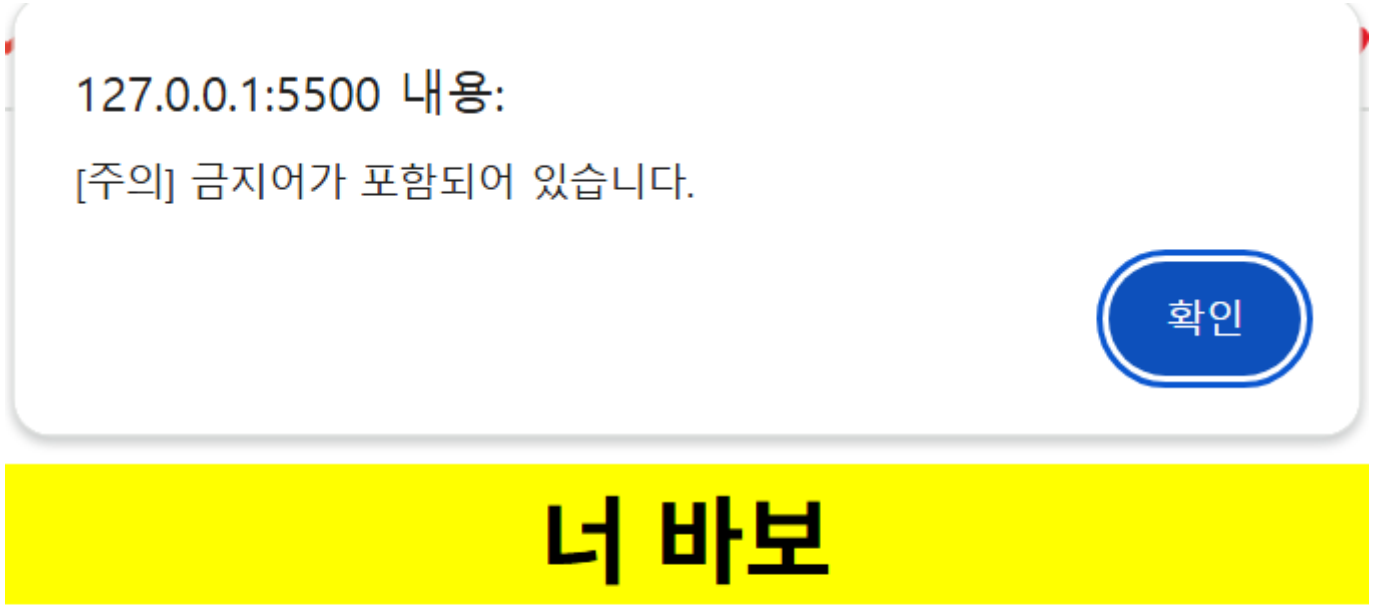
고객 요구 사항 (4)

- 공백제외 글자 수 출력

(공백 제거할 때 반복문 사용하지 말고, 문자열객체 및 배열객체의 메소드를 호출하세요.)

고객 요구 사항 (5)

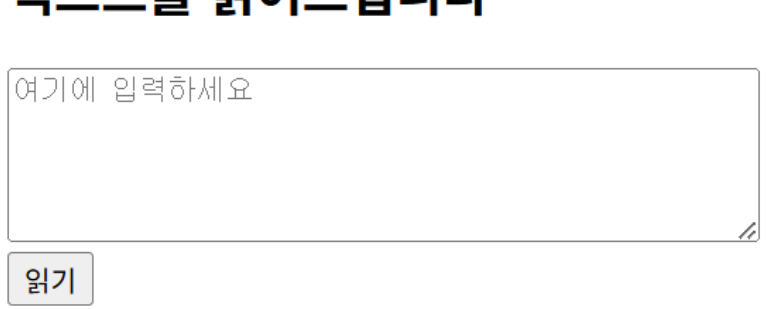
- 금지어를 말한 경우 알림창 띄우기 (금지어는 바보, 메롱 2개라고 가정)



#2 반대로, 글 읽어주는 컴퓨터

이번에는 텍스트를 음성으로 읽어주는 프로그램이 개발되었는데,
역시나 고객님께서 사용하시다가 추가 요구사항이 들어왔습니다.
그런데 팀장님께서 2가지 버전으로 코딩해보라고 합니다 T-T
(더 무서운 고객 == 팀장님)
고객님과 팀장님께 모두 감동을 안겨드리세요.

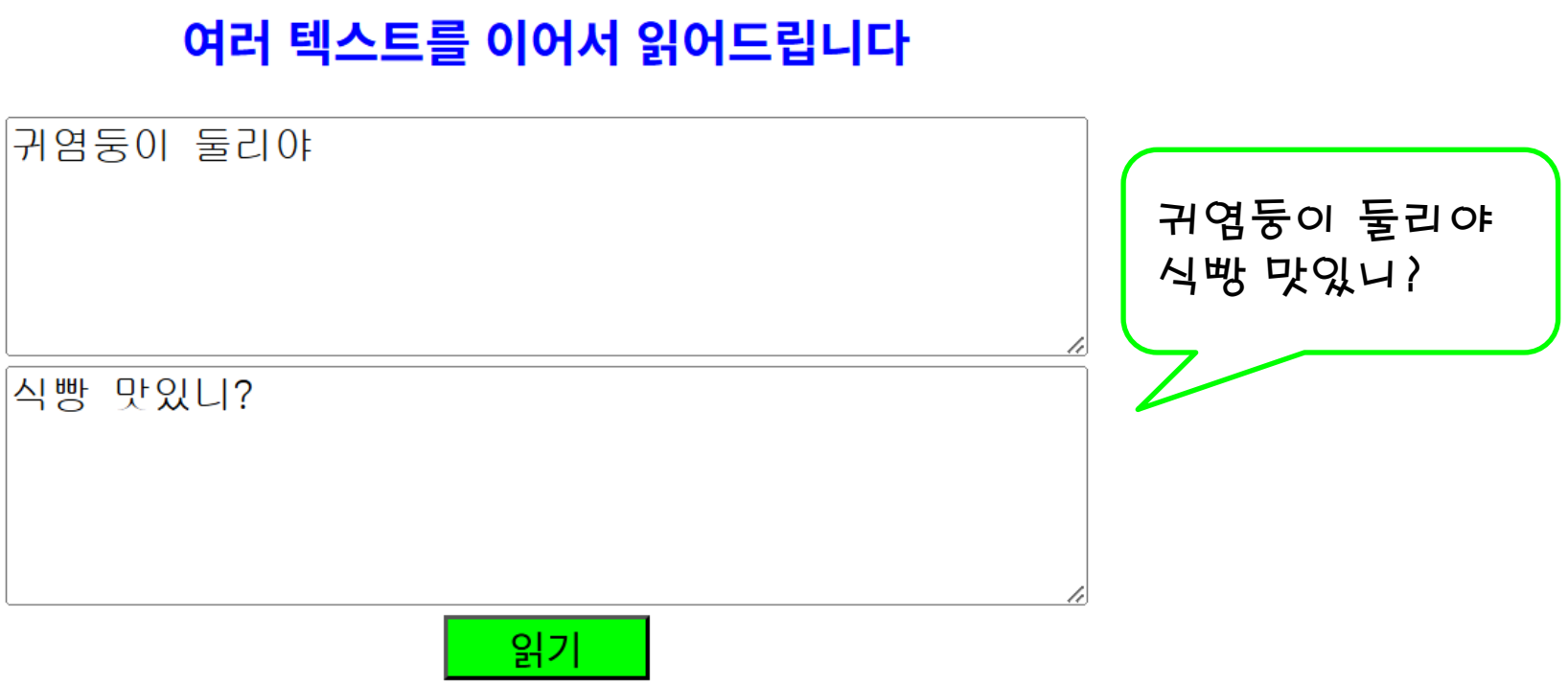
TTS원본.html




```
<!DOCTYPE html>
<html> <head>
  <meta charset="UTF-8">
  <title>TTS원본</title>
</head><body>
   <h2>텍스트를 읽어드립니다</h2>
   <textarea id="box" rows="5" cols="50" placeholder="여기에 입력하세요"></textarea><br>
   <button onclick="readText()">읽기</button>
   <script>
      function readText() {
         const a = document.getElementById("box");
         let text = a.value.trim();
         if (text === "") {
            alert("읽을 텍스트를 입력하세요.");   return;
         }
         const utterance = new SpeechSynthesisUtterance(text);
         window.speechSynthesis.speak(utterance);
      }
   </script>
</body>
</html>
```

고객님의 요구사항

- 입력박스를 2개 만들고 디자인은 아래 화면처럼 해주세요.
  (가운데 정렬, 맨 위 파란 글자, 박스 내 글자크기, 버튼 변경(색, 글자크기, 여백)
- 한 박스에라도 글자 입력 시에는 읽어주세요. 두 박스 모두 비면 팝업창 띄워주세요.
- 위 박스부터 차례대로 텍스트 읽어주세요.



팀장의 요구사항

(1) 박스가 2개 뿐이므로 반복문 사용하지 말고 getElementById() 2번 사용해서 편하게 코딩한 소스 TTS수정후.html

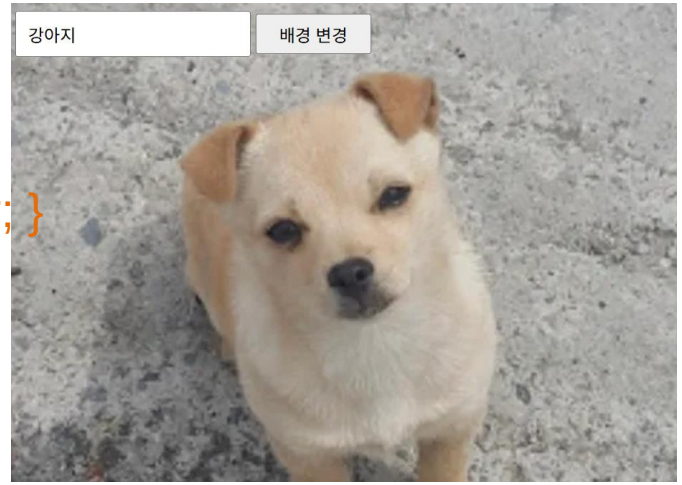
(2) 지금은 2개지만 앞으로 확장될 가능성을 대비해서 getElementsByClassName() 사용해서 배열로 받은 후 for-of 반복문으로 코딩한 소스 (TTS수정후for.html)
** 소스 안에 대괄호 [ ] 가 전혀 등장하지 않도록 해보세요 **

#3 음성으로 배경사진 바꾸기

단어를 입력하면 배경사진이 바뀌는 프로그램이 있습니다.
(API 사용하면 따로 사진위치 고민할 필요 없지만, 유료이므로 생략 ^^;;)

그런데 타이핑하지 않고 음성으로 말하면 배경사진이 바뀌게 해달라는 고객님의 요청이 접수되었습니다.

배경사진원본.html




```
<!DOCTYPE html><html> <head>
  <meta charset="UTF-8"> <title>배경사진</title>
  <style>
      body { height: 100vh;  background-size: cover;  background-position: center; }
      input { padding: 10px;  font-size: 16px; }
      button { padding: 5px 20px; font-size: 16px; cursor: pointer; }
   </style>
</head><body>
   <input type="text" id="inputWord" placeholder="단어를 입력하세요">
   <button id="changeButton">배경 변경</button>
   <script>
      const jsarr = {고양이: "이미지주소", 강아지: "이미지주소",  토끼:"이미지주소"  }; /* json */
      document.getElementById("changeButton").addEventListener("click", () => {
          const word = document.getElementById("inputWord").value.trim();
          const imageUrl = jsarr[word];
          if (imageUrl)
              document.body.style.backgroundImage = `url('${imageUrl}')`;
          else
            alert("해당 이미지를 찾을 수 없습니다. 다른 단어를 입력하세요!");
      });
   </script>
</body></html>
```

고객님의 요구 사항

(1) input: 타이핑 못하게 하고, 글자는 파란색 큰 글자 가운데 정렬
음성 인식 중에는 (마이크 사용 중)이라는 문구 나옴
(2) button: 음성인식 시작을 알리는 버튼
(3) 음성인식 결과는 input네모에 표현되며, 동시에 배경사진이 변경되도록

#4 slot machine

slot machine이 개발되었고 잘 운영되고 있었는데,
클라이언트로부터 추가 요구사항이 접수되었습니다.

slot.css

```
body {
  font-size: 30px;  display: flex;  flex-direction: column;
  align-items: center;  justify-content: center;  height: 100vh;  margin: 0;
}
.slot-machine {
  display: flex;  justify-content: center;  align-items: center;
  height: 200px;  font-size: 3.5em;  margin-bottom: 20px;
}
.slot {
  padding: 0 40px;
}
button {
  padding: 3px 10px;  font-size: 16px;
  background: #99aed5;  color: white;
  border: none;  cursor: pointer;
}
```

slot원본.html (1)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>슬롯머신</title>
  <link rel="stylesheet" href="slot.css">
</head>
<body>
  <div class="slot-machine">
    <span class="slot" id="slot1">🍀</span>
    <span class="slot" id="slot2">⭐</span>
    <span class="slot" id="slot3">⛄</span>
  </div>
  <button onclick="startSlotMachine()">슬롯머신 돌리기</button>
  <h3 id="result">ready</h3>

  <script>
    const symbols = ["🍀", "⭐", "⛄", "💘", "🐬"];
    const slotarr = document.querySelectorAll(".slot");
    const r = document.getElementById("result");
```

슬롯머신 돌리기

ready

slot원본.html (2)

```
function getRandomSymbol()  {
  return symbols[Math.floor(Math.random() * symbols.length)];
}

function startSlotMachine()
{
  const finalSymbols = [];
  for (let i = 0; i < slotarr.length; i++) finalSymbols.push(getRandomSymbol());

  let iterationCount = 0;

  function spinSlots () // 함수 안에 함수
  {
    iterationCount++;
    slotarr.forEach((s) => {s.innerHTML = getRandomSymbol();});
```

slot원본.html (3)

```
      if (iterationCount >= 10)
      {
        clearInterval(intervalId);
        slotarr.forEach((s, i) => { s.innerHTML = finalSymbols[i];});

        /* 결과 출력 */
        if (finalSymbols[0]==finalSymbols[1] && finalSymbols[1]==finalSymbols[2])
          r.textContent = "Congratulation!!";
        else if (finalSymbols[0]==finalSymbols[1] || finalSymbols[1]==finalSymbols[2] ||
                 finalSymbols[0]==finalSymbols[2])
          r.textContent = "2개 맞았음";
        else
          r.textContent = "꽝!";
      } // if >= 10

    } // function spinSlots

    const intervalId = setInterval(spinSlots, 100); // 0.1초

  } // function startSlotMachine
</script>
</body></html>
```

고객님의 요구사항 1.

현재 10회 랜덤반복 후 결과그림이 고정됩니다.
0.1초마다 반복하는 동안 화면에 1부터 10까지 출력되도록 해주세요.

고객님(보다 무서운 팀장님)의 요구사항 2.

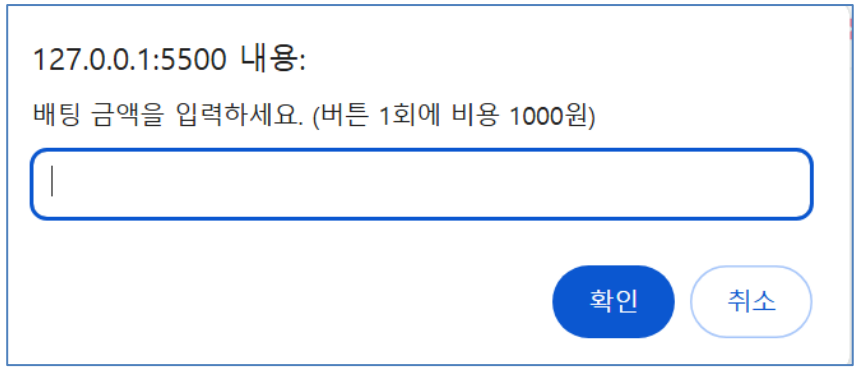
현재 startSlotMachine 함수 안에 spinSlots 함수가 있습니다.
spinSlots함수를 화살표 함수로 바꿔보세요.

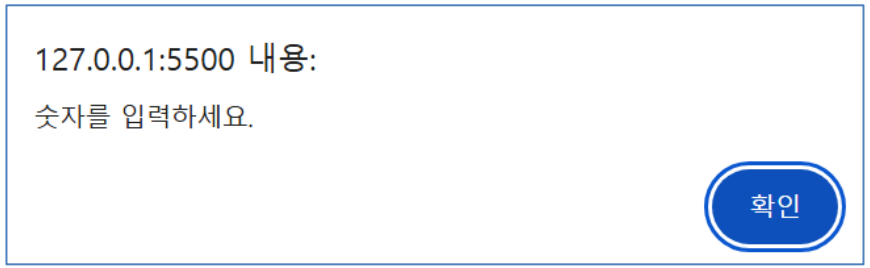
고객님의 요구사항 3.

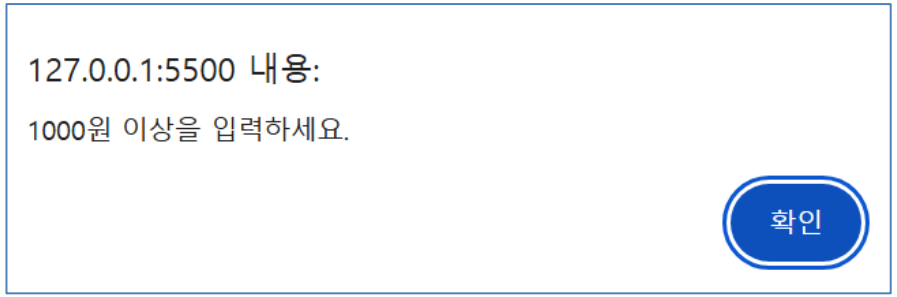
현재 특수문자그림 3개를 사용합니다. 그래서 div 안에 span태그 3개를 사용하였습니다.
그런데 진짜 그림으로 바꾸려고 합니다. span 3개 말고 img 3개로 바꿔보세요.

고객님의 요구사항 4.

- 버튼 1회 천원이라고 가정합니다. const cost=1000; 으로 선언 후 더이상 소스에 1000 이라는 숫자는 등장하지 않고 대신 cost변수 사용)
- 그리고 배팅금액을 입력할 수 있도록 수정하세요. (숫자인지 먼저 체크하고, 그 다음 최소금액 이상인지 체크해서, 1000원 이상 숫자를 입력할 때까지 반복)

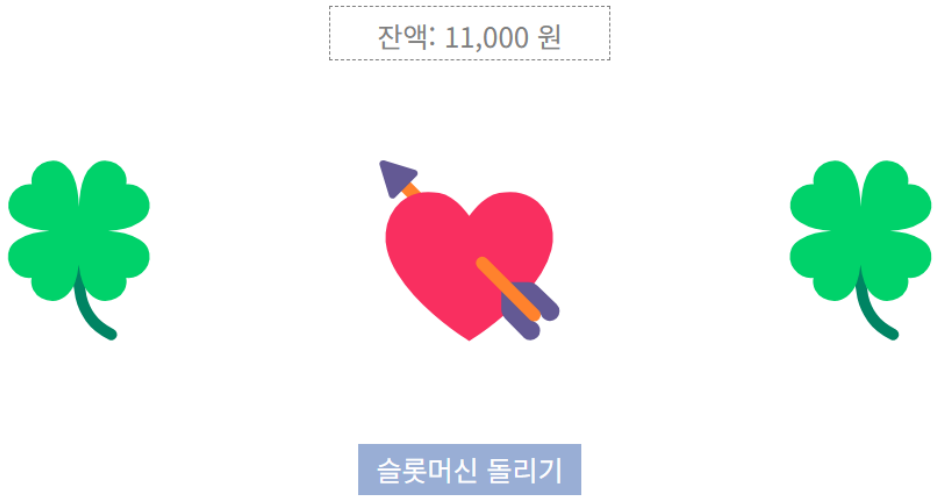






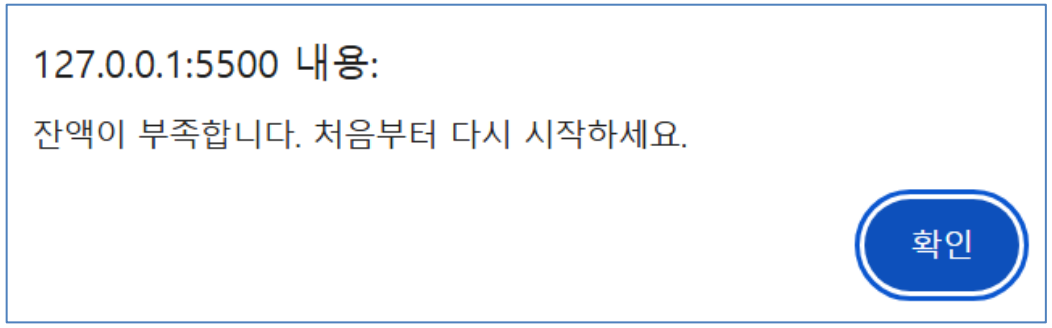
고객님의 요구사항 5.

화면 윗부분에 잔액이 보이고, 꽝이면 cost만큼 줄어들고(즉, 천원 마이너스),
2개 맞으면 cost만큼 늘어나고(즉, 천원 플러스),
잭팟이면 cost의 10배만큼 잔액 늘어나도록(즉, 만원 플러스) 변경하세요.
화면에 출력할 때 천단위 쉼표 넣으려면 money가 잔액 변수일 때
... ${money.toLocaleString()} ... 이렇게 하시면 됩니다.

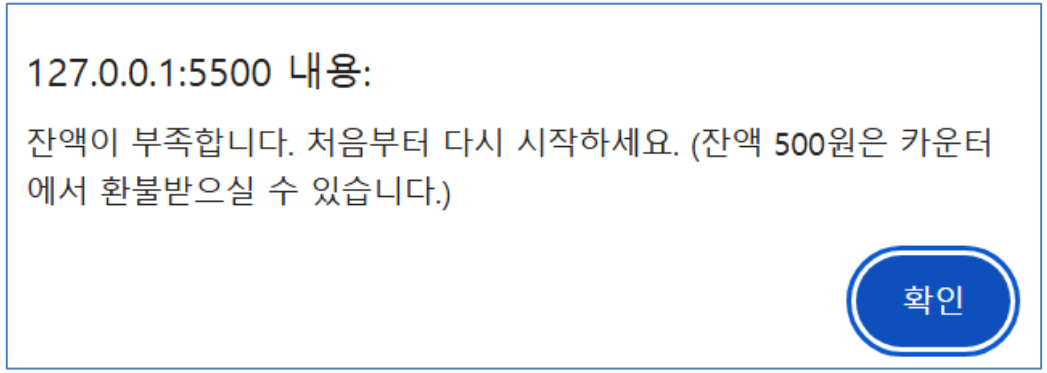


고객님의 요구사항 6.

잔액이 cost보다 적으면 다시 배팅하라는 팝업창이 뜨고, 프로그램 다시 시작
(다시 시작하는 방법은 여러 가지인데 location.reload(); 가 그 중 하나입니다.)



잔액 0원 경우



잔액이 애매하게 남은 경우

#5 올림픽

올림픽 육상 경기가 개발되었고 잘 운영되고 있었는데,
팀장님으로부터 소스를 수정하라는 오더가 들어왔습니다.

올림픽.css

```
.track {
  width: 96%;  height: 30px;  background: #ded;
  position: relative;  margin: 20px 0;
  border-left: solid 5px #bbb;  border-right: solid 5px red;
}
.athlete {
  width: 30px;  height: 30px;   background: #ec9c55;
  position: absolute;  top: 0;
  border-radius: 0 10px 10px 0;
}
.number {
  position: absolute;  color: white;
  font-size: 20px;  left: 50%;
  transform: translateX(-50%);
}
```

올림픽원본.html (1)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>올림픽</title>
  <link rel="stylesheet" href="올림픽.css">
</head>
<body>
  <div class="track" id="track1"></div>
  <div class="track" id="track2"></div>
  <div class="track" id="track3"></div>
  <div class="track" id="track4"></div>
  <div class="track" id="track5"></div>

  <script>
    const NUM = 5;
    let finishedCount = 0;
    let winner = null;
```

올림픽원본.html (2)

```
const tracks = document.querySelectorAll(".track");
const athletes = [];

for (let i = 0; i < NUM; i++) {
  const a = document.createElement("div");
  a.className = "athlete";
  a.style.left = "0";

  const n = document.createElement("div");
  n.className = "number";
  n.textContent = i + 1;
  a.appendChild(n);

  tracks[i].appendChild(a);
  athletes.push(a);
}

const gameInterval = setInterval(() => {
  for (let i = 0; i < NUM; i++)
    moveAthlete(athletes[i], i);
}, 100);
```

올림픽원본.html (3)

```
    function moveAthlete(a, i) {
      const distance = Math.random() * 10;
      const currentLeft = parseFloat(a.style.left);
      const newLeft = currentLeft + distance;

      a.style.left = newLeft + "px";

      if (newLeft >= (tracks[i].clientWidth - 28) && !winner) {
        winner = a;
        a.style.backgroundColor = "blue";
        finishedCount++;

        if (finishedCount === 1) {
          clearInterval(gameInterval);
        }
      }
    }

  </script>
</body>
</html>
```

팀장님의 요구사항 1.

원본 소스 body에 있는 트랙 div 5개를 삭제하고
자바스크립트에서 create, append 할 것!

팀장님의 요구사항 2.

이번에는 반대로,
현재 원본 자바스크립트에서 선수 div (넘버 div 포함)를 create, append 하고 있는데 이를 삭제하고,
body쪽에 직접 코딩하세요.

팀장님의 요구사항 3.

원본 자바스크립트 소스 setInterval() 괄호 안 화살표 함수를 삭제하고,
별도의 함수를 정의하세요.

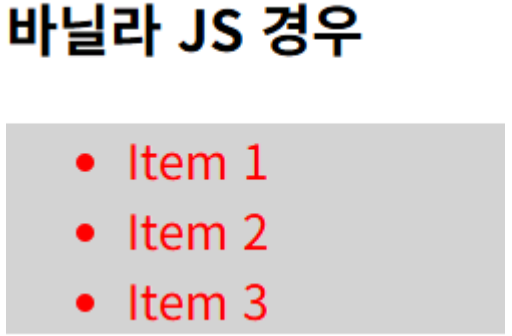
#6 Vanilla JS? jQuery? React? Vue.js?

새롭게 시작하는 Front-end 개발 프로젝트에서 위 4가지 중 어느 쪽을 선택할 것인지 검토

- 현재 FE 개발자들의 가장 큰 관심사 중의 하나
- 한 번 결정되면 되돌리기 어려움. 신중히 접근해야
- 나무위키 등을 검색해도 내용이 잘 정리되어 있습니다.

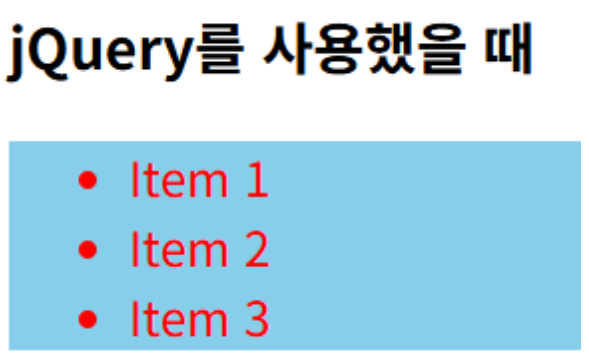
Vanilla JS, jQuery 단순 비교

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"> <title>Vanilla JS</title>
</head>
<body>
    <h3>바닐라 JS 경우</h3>
    <ul id="list">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <script>
        document.getElementById('list').style.background = 'lightgray';
        document.querySelectorAll('li').forEach(i => i.style.color = 'red');
    </script>
</body>
</html>
```



Vanilla JS, jQuery 단순 비교

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"> <title>jQuery</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h3>jQuery를 사용했을 때</h3>
    <ul id="list">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <script>
        $('#list').css('background', 'skyblue');
        $('li').css('color', 'red');
    </script>
</body>
</html>
```



한영사전 – 데이터 (공통으로 사용)

```
// words.js
const dictionary = {
  "사과": "apple",
  "바나나": "banana",
  "포도": "grape",
  "오렌지": "orange",
  "수박": "watermelon",
  "체리": "cherry",
  "키위": "kiwi",
  "복숭아": "peach",
  "레몬": "lemon",
  "망고": "mango",
};

export default dictionary; // 모듈 방식 사용
```

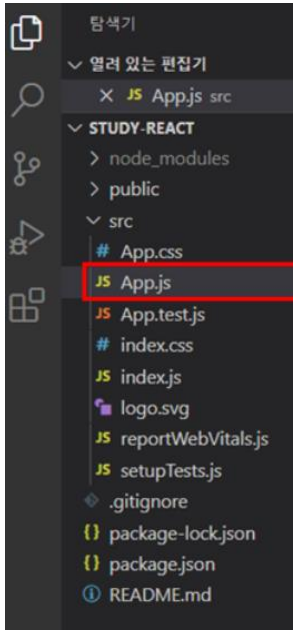
한영사전 – Vanilla.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"> <title>Vanilla JS Dictionary</title>
</head>
<body>
  <input type="text" id="input-word" placeholder="한국어 단어 입력">
  <button id="translate-button">번역</button>
  <p id="result"></p>

  <script type="module">
    import dictionary from './words.js';

    document.getElementById('translate-button').addEventListener('click', () => {
      const inputWord = document.getElementById('input-word').value;
      const result = dictionary[inputWord] || '단어를 찾을 수 없습니다.';
      document.getElementById('result').textContent = result;
    });
  </script>
</body>
</html>
```

한영사전 – jquery.html

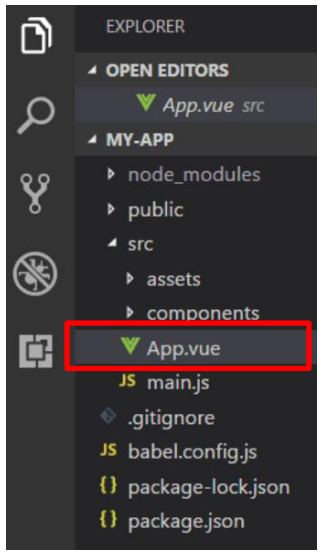
```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8"> <title>jQuery Dictionary</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <input type="text" id="input-word" placeholder="한국어 단어 입력" />
  <button id="translate-button">번역</button>
  <p id="result"></p>

  <script type="module">
    import dictionary from './words.js';

    $(document).ready(function () {
      $('#translate-button').click(function () {
        const inputWord = $('#input-word').val();
        const result = dictionary[inputWord] || '단어를 찾을 수 없습니다.';
        $('#result').text(result);
      });
    });
  </script>
</body>
</html>
```

한영사전 – App.js (리액트)




```
import React, { useState } from 'react';
import dictionary from './words.js';

function App() {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');

  const handleTranslate = () => {
    setResult(dictionary[input] || '단어를 찾을 수 없습니다.');
  };

  return (
    <div>
      <input
        type="text"
        value={input}
        onChange={(e) => setInput(e.target.value)}
        placeholder="한국어 단어 입력" />
      <button onClick={handleTranslate}>번역</button>
      <p>{result}</p>
    </div>
  );
}
export default App;
```

한영사전 – App.vue (뷰)




```
<template>
  <div>
    <input type="text"
      v-model="inputWord"
      placeholder="한국어 단어 입력" />
    <button @click="translate">번역</button>
    <p>{{ result }}</p>
  </div>
</template>
<script>
import dictionary from './words.js';
export default {
  data() {
    return {
      inputWord: '',
      result: '',
    };
  },
  methods: {
    translate() {
      this.result = dictionary[this.inputWord] || '단어를 찾을 수 없습니다.';
    },
  },
};
</script>
```