**숫자 야구 게임 제작 보고서**

**1. 프로젝트 개요**

* 게임 제목 : 숫자야구게임
* 제작자 : 컴퓨터공학부 202395007 김수종
* 제작 기간 : 2025년 10월 31일
* 사용 언어 : JavaScript, HTML, CSS
* 팀 구성 : 개인
* 플랫폼 : Chrome, Edge 등

**2. 게임 기획 및 컨셉**

* 기획 의도 : 단순한 숫자 추리 게임을 통해 논리적 사고력과 집중력을 기를 수 있도록 기획. 짧은 시간 동안 누구나 쉽게 즐길 수 있는 두뇌 게임 제공
* 장르 : 캐주얼 / 퍼즐 / 추리 게임
* 핵심 아이디어 : 컴퓨터가 무작위로 생성한 1~10자리(기본 4자리) 숫자를 플레이어가 맞히는 방식.  
  시도할 때마다 “스트라이크(숫자와 자리 모두 일치)”와 “볼(숫자는 맞지만 위치 불일치)”의 피드백을 제공하여 점진적으로 정답에 접근.

사용자가 게임 플레이 방식을 변경할수 있는 설정 추가

* 타겟 유저/플레이 스타일 : 모든 연령층 대상, 짧은 시간에 간단한 게임을 즐기고 싶은 일반 사용자, 두뇌 게임, 퍼즐 게임을 좋아하는 사용자, 반복적인 도전과 기록 갱신을 선호하는 사용자

**3. 게임 시스템 구조**

* 게임 진행 방식 :

1. **숫자 생성**

* 컴퓨터가 0~9 사이의 숫자 중 중복되지 않는 4자리(1~10까지 설정 가능) 숫자를 무작위로 생성한다.

2. **플레이어 입력**

* 플레이어는 추측한 숫자를 입력한다.

3. **판정**

* 입력한 숫자와 정답 숫자를 비교하여 결과를 알려준다.
  + **스트라이크 (S)** : 숫자와 위치가 모두 일치
  + **볼 (B)** : 숫자는 맞지만 위치가 다름

4. **반복 진행**

* 플레이어는 결과를 참고하여 다시 숫자를 입력한다.
* 정답을 맞히거나 제한된 시도 횟수를 모두 사용 혹은 제한 시간이 지나면 게임이 종료된다.

5. **게임 종료 조건**

* 모든 숫자와 자리를 정확히 맞혀 **정답**을 달성하면 승리.
* 제한 횟수를 초과하면 패배.
* 제한 시간이 종료되면 패배
* 다이어그램 :

스크린샷, 텍스트, 도표, 디자인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**4. 기술적 구현 내용**

* 주요 스크립트 설명 :

1. 게임 설정 관리

javascript

let gameSettings = {

    maxAttempts: 10,

    numberLength: 4,

    allowDuplicates: false,

    timeLimit: 60,

    unlimitedAttempts: false,

    unlimitedTime: true

};

- **기능**: 게임 난이도 설정 (기회 횟수, 숫자 길이, 중복 허용, 시간 제한)

- **특징**: 실시간 설정 변경 가능

**2. 정답 생성 로직**

javascript

if (gameSettings.allowDuplicates) {

    for (let i = 0; i < gameSettings.numberLength; i++) {

        answer.push(Math.floor(Math.random() \* 10));

    }

} else {

    while (numbers.size < gameSettings.numberLength) {

        const randomNum = Math.floor(Math.random() \* 9) + 1;

        numbers.add(randomNum);

    }

}

- **\*\*기능\*\***: 중복 허용 여부에 따른 랜덤 숫자 생성

- **\*\*핵심\*\***: Set 자료구조로 중복 제거

**3. 스트라이크/볼 계산**

javascript

function calculateResult(userInput) {

    const userNumbers = userInput.split('').map(Number);

    let strikes = 0, balls = 0;

    // 스트라이크: 위치와 숫자 모두 일치

    for (let i = 0; i < gameSettings.numberLength; i++) {

        if (userNumbers[i] === answer[i]) strikes++;

    }

    // 볼: 숫자는 맞지만 위치 다름

    for (let i = 0; i < gameSettings.numberLength; i++) {

        if (userNumbers[i] !== answer[i] && answer.includes(userNumbers[i])) {

            balls++;

        }

    }

    return { strikes, balls };

}

```

- **기능**: 사용자 입력과 정답 비교

- **알고리즘**: 이중 반복문으로 정확한 판정

**4. 타이머 시스템**

```javascript

function startTimer() {

    if (gameSettings.unlimitedTime) return;

    timer = setInterval(() => {

        timeRemaining--;

        updateTimerDisplay();

        if (timeRemaining <= 0) {

            clearInterval(timer);

            handleTimeUp();

        }

    }, 1000);

}

```

- **기능**: 실시간 카운트다운 타이머

- **특징**: 시간에 따른 UI 색상 변화 (정상→경고→위험)

* **문제 해결 경험**

**1. 애니메이션 과다 문제**

**\*\*문제\*\***: 홈런 시 과도한 애니메이션으로 사용자 경험 저하

css

수정 전 - 과도한 애니메이션

animation: homerunCelebration 1s ease-out, confetti 2s ease-out;

수정 후 - 자연스러운 애니메이션

@keyframes homerunSuccess {

    0% { transform: scale(0.95) translateY(10px); opacity: 0; }

    60% { transform: scale(1.02) translateY(-2px); opacity: 1; }

    100% { transform: scale(1) translateY(0); opacity: 1; }

}

```

**해결**: 키프레임 단순화, 지속시간 조정으로 자연스러운 효과 구현

**2. 반응형 레이아웃 깨짐**

**문제**: 다양한 해상도에서 UI 요소들이 겹치거나 깨짐

css

문제 해결 - clamp() 함수 사용

font-size: clamp(1rem, 2vw, 1.3rem);

padding: clamp(12px, 2.5vw, 18px);

max-width: min(90vw, 800px);

```

**해결**: `clamp()`, `min()`, `vw` 단위로 유동적 크기 조정

**3. 설정 UI 배치 문제**

**문제**: 사이드바가 게임 화면을 가려서 플레이 방해

```html

수정 전 - 사이드바

<div class="settings-sidebar">...</div>

<!-- 수정 후 - 상단 패널 -->

<div class="settings-panel">

    <div class="settings-row">...</div>

</div>

```

**해결**: 사이드바를 상단 고정 패널로 변경하여 공간 효율성 향상

**4. 중복 숫자 검증 로직 오류**

**문제**: 중복 허용/불허 설정에 따른 입력 검증 실패

javascript

// 해결책 - 설정에 따른 분기 처리

if (!gameSettings.allowDuplicates) {

    if (inputValue.includes('0')) {

        return { valid: false, message: '1-9 숫자만 사용해주세요!' };

    }

    const uniqueDigits = new Set(inputValue);

    if (uniqueDigits.size !== gameSettings.numberLength) {

        return { valid: false, message: '중복되지 않는 숫자를 입력해주세요!' };

    }

}

**해결**: 설정 상태에 따른 조건부 검증 로직 구현

**5. 타이머 위치 문제**

**문제**: 타이머가 다른 UI 요소와 겹치거나 이상한 위치에 표시

css

해결책 - 고정 위치 설정 \*/

.timer-display {

    position: fixed;

    top: 50%;

    right: 20px;

    transform: translateY(-50%);

    z-index: 999;

}

**해결**: `position: fixed`와 `transform`으로 화면 우측 중앙 고정

**5. 디자인**

* UI 구성 :

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**6. 테스트 및 피드백**

* 테스트 방법 : 친구/학생
* 발견된 문제점 및 개선 : 문제점 발견 X, 하지만 UI에 있어 불편함이 어느정도 생길 수 있음을 확인

**7. 회고 및 느낀 점**

* 어떤 점이 가장 어려웠고, 어떤 점에서 성취감을 느꼈는가

원하는 추가기능을 넣어달라고 했는데 오류가 계속 발생하여 고치느라 힘들었고 결국 완성을 시켜 뿌듯했습니다.

* 다음에 만들고 싶은 게임 아이디어

고군분투 같은 인내심 및 집중력을 요구하는 게임을 만들어보고 싶습니다.

* 학습적으로 어떤 성장을 했는가

AI 프롬프트 작성 능력을 향상시켰습니다.

**8. 스크린샷 및 게임 URL**

* 주요 장면 스크린샷
* 기능별 화면

숫자 입력 후

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

기회를 전부 사용하였을 때

텍스트, 스크린샷, 폰트, 운영 체제이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

정답을 맞췄을 때 (맞추질 못해서 참고 자료가 없음)

* URL 주소 : https://rlatnwhd.github.io/Number-Baseball-Game/

**9. 코드**

* js (파일명 : number-baseball.js )
* // 게임 설정 변수
* let gameSettings = {
* maxAttempts: 10,
* numberLength: 4,
* allowDuplicates: false,
* timeLimit: 60,
* unlimitedAttempts: false,
* unlimitedTime: true
* };
* // 게임 상태 변수
* let answer = []; // 정답 숫자 배열
* let attempts = 0; // 시도 횟수
* let gameOver = false; // 게임 종료 여부
* let timeRemaining = 0; // 남은 시간
* let timer = null; // 타이머 객체
* // DOM 요소
* const form = document.getElementById('form');
* const input = document.getElementById('input');
* const logs = document.getElementById('logs');
* const attemptsNumber = document.getElementById('attempts-number');
* // 설정 관련 DOM 요소
* const applySettings = document.getElementById('apply-settings');
* // 시도 횟수 업데이트 함수
* function updateAttemptsDisplay() {
* if (gameSettings.unlimitedAttempts) {
* attemptsNumber.textContent = '무한';
* attemptsNumber.classList.remove('warning', 'danger');
* return;
* }
* const remaining = gameSettings.maxAttempts - attempts;
* attemptsNumber.textContent = remaining;
* // 남은 기회에 따라 색상과 애니메이션 변경
* attemptsNumber.classList.remove('warning', 'danger');
* if (remaining <= 3 && remaining > 1) {
* attemptsNumber.classList.add('warning');
* } else if (remaining <= 1) {
* attemptsNumber.classList.add('danger');
* }
* }
* // 타이머 관련 함수
* function createTimerDisplay() {
* if (!gameSettings.unlimitedTime) {
* const timerDiv = document.createElement('div');
* timerDiv.id = 'timer-display';
* timerDiv.className = 'timer-display';
* document.body.appendChild(timerDiv);
* updateTimerDisplay();
* timerDiv.style.display = 'block';
* }
* }
* function updateTimerDisplay() {
* const timerDiv = document.getElementById('timer-display');
* if (!timerDiv || gameSettings.unlimitedTime) return;
* const minutes = Math.floor(timeRemaining / 60);
* const seconds = timeRemaining % 60;
* timerDiv.textContent = `⏰ ${minutes}:${seconds.toString().padStart(2, '0')}`;
* // 시간에 따른 경고 표시
* timerDiv.classList.remove('warning', 'danger');
* if (timeRemaining <= 30 && timeRemaining > 10) {
* timerDiv.classList.add('warning');
* } else if (timeRemaining <= 10) {
* timerDiv.classList.add('danger');
* }
* }
* function startTimer() {
* if (gameSettings.unlimitedTime) return;
* timeRemaining = gameSettings.timeLimit;
* createTimerDisplay();
* timer = setInterval(() => {
* timeRemaining--;
* updateTimerDisplay();
* if (timeRemaining <= 0) {
* clearInterval(timer);
* handleTimeUp();
* }
* }, 1000);
* }
* function stopTimer() {
* if (timer) {
* clearInterval(timer);
* timer = null;
* }
* const timerDiv = document.getElementById('timer-display');
* if (timerDiv) {
* timerDiv.remove();
* }
* }
* function handleTimeUp() {
* const logDiv = document.createElement('div');
* logDiv.className = 'log-entry game-over';
* logDiv.innerHTML = `<strong>시간 초과! ⏰<br>정답: ${answer.join('')}</strong>`;
* logs.appendChild(logDiv);
* gameOver = true;
* input.disabled = true;
* stopTimer();
* showRestartButton();
* }
* // 게임 시작 함수
* function startGame() {
* // 설정에 따라 숫자 생성
* const numbers = new Set();
* const maxDigit = gameSettings.allowDuplicates ? 10 : 9; // 0 포함 여부
* const startDigit = gameSettings.allowDuplicates ? 0 : 1; // 시작 숫자
* if (gameSettings.allowDuplicates) {
* // 중복 허용 시
* answer = [];
* for (let i = 0; i < gameSettings.numberLength; i++) {
* answer.push(Math.floor(Math.random() \* 10));
* }
* } else {
* // 중복 불허 시
* while (numbers.size < gameSettings.numberLength) {
* const randomNum = Math.floor(Math.random() \* 9) + 1; // 1-9만 사용
* numbers.add(randomNum);
* }
* answer = Array.from(numbers);
* }
* // 게임 상태 초기화
* attempts = 0;
* gameOver = false;
* logs.innerHTML = '';
* input.disabled = false;
* input.maxLength = gameSettings.numberLength;
* input.placeholder = `${gameSettings.numberLength}자리 숫자 입력`;
* input.focus();
* // 시도 횟수 표시 업데이트
* updateAttemptsDisplay();
* // 타이머 시작
* stopTimer(); // 기존 타이머 정리
* startTimer();
* console.log('정답:', answer); // 개발용 (실제로는 제거)
* }
* // 애니메이션 트리거 함수들
* function triggerInputShake() {
* input.classList.add('shake');
* setTimeout(() => input.classList.remove('shake'), 500);
* }
* function triggerInputSuccess() {
* input.classList.add('success');
* setTimeout(() => input.classList.remove('success'), 800);
* }
* // 입력 검증 함수
* function validateInput(inputValue) {
* // 설정된 길이와 일치하는지 확인
* if (inputValue.length !== gameSettings.numberLength) {
* triggerInputShake();
* return { valid: false, message: `${gameSettings.numberLength}자리 숫자를 입력해주세요!` };
* }
* // 숫자만 포함하는지 확인
* if (!/^\d+$/.test(inputValue)) {
* triggerInputShake();
* return { valid: false, message: '숫자만 입력해주세요!' };
* }
* // 중복 허용 여부에 따른 검증
* if (!gameSettings.allowDuplicates) {
* // 0이 포함되지 않았는지 확인 (1-9만 사용)
* if (inputValue.includes('0')) {
* triggerInputShake();
* return { valid: false, message: '1-9 숫자만 사용해주세요!' };
* }
* // 중복 숫자가 없는지 확인
* const uniqueDigits = new Set(inputValue);
* if (uniqueDigits.size !== gameSettings.numberLength) {
* triggerInputShake();
* return { valid: false, message: `중복되지 않는 ${gameSettings.numberLength}개의 숫자를 입력해주세요!` };
* }
* }
* return { valid: true };
* }
* // 스트라이크와 볼 계산 함수
* function calculateResult(userInput) {
* const userNumbers = userInput.split('').map(Number);
* let strikes = 0;
* let balls = 0;
* // 스트라이크 계산 (위치와 숫자가 모두 맞음)
* for (let i = 0; i < gameSettings.numberLength; i++) {
* if (userNumbers[i] === answer[i]) {
* strikes++;
* }
* }
* // 볼 계산 (숫자는 맞지만 위치가 다름)
* for (let i = 0; i < gameSettings.numberLength; i++) {
* if (userNumbers[i] !== answer[i] && answer.includes(userNumbers[i])) {
* balls++;
* }
* }
* return { strikes, balls };
* }
* // 결과 표시 함수
* function displayResult(userInput, strikes, balls) {
* if (strikes === gameSettings.numberLength) {
* // 홈런 - 별도 함수로 처리
* handleHomerun(userInput);
* return;
* }
* // 일반 결과 표시
* const logDiv = document.createElement('div');
* logDiv.className = 'log-entry';
* logDiv.innerHTML = `${userInput} : ${strikes} 스트라이크, ${balls} 볼`;
* logs.appendChild(logDiv);
* logs.scrollTop = logs.scrollHeight; // 스크롤을 최하단으로
* // 시도 횟수 업데이트
* updateAttemptsDisplay();
* }
* // 홈런 처리 함수
* function handleHomerun(userInput) {
* const logDiv = document.createElement('div');
* logDiv.className = 'log-entry homerun';
* logDiv.innerHTML = `<strong>홈런! 🎉<br>${userInput} 정답!</strong>`;
* logs.appendChild(logDiv);
* gameOver = true;
* input.disabled = true;
* stopTimer();
* triggerInputSuccess();
* // 홈런 사운드 효과 (브라우저 지원 시)
* if (typeof Audio !== 'undefined') {
* try {
* const audio = new Audio('data:audio/wav;base64,UklGRnoGAABXQVZFZm10IBAAAAABAAEAQB8AAEAfAAABAAgAZGF0YQoGAACBhYqFbF1fdJivrJBhNjVgodDbq2EcBj+a2/LDciUFLIHO8tiJNwgZaLvt559NEAxQp+PwtmMcBjiR1/LMeSwFJHfH8N2QQAoUXrTp66hVFApGn+DyvmweCSnU4v3ScA==')
* audio.play().catch(() => {}); // 오류 무시
* } catch(e) {}
* }
* showRestartButton();
* }
* // 게임 오버 처리 함수
* function handleGameOver() {
* const logDiv = document.createElement('div');
* logDiv.className = 'log-entry game-over';
* logDiv.innerHTML = `<strong>아웃! 😢<br>정답: ${answer.join('')}</strong>`;
* logs.appendChild(logDiv);
* gameOver = true;
* input.disabled = true;
* stopTimer();
* // 시도 횟수 업데이트
* updateAttemptsDisplay();
* showRestartButton();
* }
* // 다시하기 버튼 표시 함수
* function showRestartButton() {
* const restartBtn = document.createElement('button');
* restartBtn.textContent = '다시하기';
* restartBtn.className = 'restart-btn';
* restartBtn.onclick = () => {
* restartBtn.remove();
* startGame();
* };
* logs.appendChild(restartBtn);
* }
* // 폼 제출 이벤트 처리
* form.addEventListener('submit', (e) => {
* e.preventDefault();
* if (gameOver) return;
* const userInput = input.value.trim();
* // 입력 검증
* const validation = validateInput(userInput);
* if (!validation.valid) {
* alert(validation.message);
* input.value = '';
* return;
* }
* // 시도 횟수 증가
* attempts++;
* // 스트라이크와 볼 계산
* const { strikes, balls } = calculateResult(userInput);
* // 성공적인 입력에 대한 애니메이션
* if (strikes < 4) {
* triggerInputSuccess();
* }
* // 결과 표시
* displayResult(userInput, strikes, balls);
* // 게임 종료 조건 확인
* if (strikes === gameSettings.numberLength) {
* // 홈런 - 이미 displayResult에서 처리됨
* } else if (!gameSettings.unlimitedAttempts && attempts >= gameSettings.maxAttempts) {
* // 아웃
* handleGameOver();
* }
* // 입력창 초기화
* input.value = '';
* input.focus();
* });
* // 숫자만 입력 가능하도록 제한
* input.addEventListener('input', (e) => {
* if (gameSettings.allowDuplicates) {
* e.target.value = e.target.value.replace(/[^0-9]/g, '');
* } else {
* e.target.value = e.target.value.replace(/[^1-9]/g, '');
* }
* });
* // 설정 적용 이벤트
* applySettings.addEventListener('click', () => {
* const confirmed = confirm('설정을 적용하면 현재 게임이 다시 시작됩니다. 계속하시겠습니까?');
* if (!confirmed) return;
* // 설정 값 읽어오기
* const maxAttempts = parseInt(document.getElementById('max-attempts').value) || 10;
* const numberLength = parseInt(document.getElementById('number-length').value) || 4;
* const allowDuplicates = document.getElementById('allow-duplicates').checked;
* const timeLimit = parseInt(document.getElementById('time-limit').value) || 60;
* const unlimitedAttempts = document.getElementById('unlimited-attempts').checked;
* const unlimitedTime = document.getElementById('unlimited-time').checked;
* // 설정 유효성 검사
* if (numberLength < 2 || numberLength > 10) {
* alert('숫자 길이는 2~10 사이여야 합니다.');
* return;
* }
* if (!unlimitedAttempts && (maxAttempts < 1 || maxAttempts > 999)) {
* alert('기회 횟수는 1~999 사이여야 합니다.');
* return;
* }
* if (!unlimitedTime && (timeLimit < 10 || timeLimit > 999)) {
* alert('시간 제한은 10~999초 사이여야 합니다.');
* return;
* }
* if (!allowDuplicates && numberLength > 9) {
* alert('중복 숫자를 허용하지 않으면 최대 9자리까지만 가능합니다.');
* return;
* }
* // 설정 적용
* gameSettings = {
* maxAttempts,
* numberLength,
* allowDuplicates,
* timeLimit,
* unlimitedAttempts,
* unlimitedTime
* };
* // 게임 재시작
* startGame();
* alert('설정이 적용되었습니다!');
* });
* // 무한 기회 체크박스 이벤트
* document.getElementById('unlimited-attempts').addEventListener('change', (e) => {
* document.getElementById('max-attempts').disabled = e.target.checked;
* });
* // 무제한 시간 체크박스 이벤트
* document.getElementById('unlimited-time').addEventListener('change', (e) => {
* document.getElementById('time-limit').disabled = e.target.checked;
* });
* // 게임 시작
* startGame();
* html (파일명 : index.html )
* <!DOCTYPE html>
* <html lang="ko">
* <head>
* <meta charset="UTF-8">
* <meta name="viewport" content="width=device-width, initial-scale=1.0">
* <title>숫자야구</title>
* <link rel="stylesheet" href="css/number-baseball.css">
* </head>
* <body>
* <!-- 상단 설정 패널 -->
* <div id="settings-panel" class="settings-panel">
* <h3>게임 설정</h3>
* <div class="settings-row">
* <div class="setting-item">
* <label for="max-attempts">기회 횟수</label>
* <input type="number" id="max-attempts" min="1" max="999" value="10">
* <label class="checkbox-label">
* <input type="checkbox" id="unlimited-attempts"> 무한
* </label>
* </div>
* <div class="setting-item">
* <label for="number-length">숫자 길이</label>
* <input type="number" id="number-length" min="2" max="10" value="4">
* </div>
* <div class="setting-item">
* <label class="checkbox-label">
* <input type="checkbox" id="allow-duplicates"> 중복 허용
* </label>
* </div>
* <div class="setting-item">
* <label for="time-limit">시간 제한(초)</label>
* <input type="number" id="time-limit" min="10" max="999" value="60" disabled>
* <label class="checkbox-label">
* <input type="checkbox" id="unlimited-time" checked> 무제한
* </label>
* </div>
* <div class="setting-item">
* <button id="apply-settings" class="apply-btn">설정 적용</button>
* </div>
* </div>
* </div>
* <div class="main-content">
* <h1>숫자야구 게임</h1>
* <div class="instruction">
* 4자리 숫자를 맞춰보세요! (중복 없음, 1-9 숫자만 사용)
* </div>
* <div class="attempts-counter" id="attempts-counter">
* <span class="attempts-text">남은 기회:</span>
* <span class="attempts-number" id="attempts-number">10</span>
* </div>
* <form id="form">
* <input type="text" id="input" placeholder="4자리 숫자 입력" maxlength="4">
* <button>확인</button>
* </form>
* <div id="logs"></div>
* </div>
* <script src="js/number-baseball.js"></script>
* </body>
* </html>
* css (파일명 : number-baseball.css )
* body {
* font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
* margin: 0;
* padding: 0;
* background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
* min-height: 100vh;
* color: #333;
* animation: backgroundPulse 8s ease-in-out infinite;
* box-sizing: border-box;
* }
* .main-content {
* max-width: min(90vw, 800px);
* margin: 0 auto;
* padding: min(3vw, 20px);
* box-sizing: border-box;
* }
* body::before {
* content: '';
* position: fixed;
* top: 50%;
* left: 50%;
* transform: translate(-50%, -50%);
* background-image: url('image/baseball.png');
* background-size: contain;
* background-repeat: no-repeat;
* background-position: center;
* width: min(25vw, 250px);
* height: min(25vw, 250px);
* opacity: 0.1;
* z-index: -1;
* pointer-events: none;
* }
* @keyframes backgroundPulse {
* 0%, 100% { filter: brightness(1); }
* 50% { filter: brightness(1.1); }
* }
* h1 {
* text-align: center;
* color: white;
* text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
* margin-bottom: 10px;
* font-size: clamp(1.8rem, 4vw, 2.5rem);
* animation: titleBounce 4s ease-in-out infinite;
* }
* @keyframes titleBounce {
* 0%, 100% { transform: translateY(0); }
* 50% { transform: translateY(-5px); }
* }
* .attempts-counter {
* text-align: center;
* background: rgba(255,255,255,0.95);
* padding: clamp(12px, 2.5vw, 18px);
* border-radius: clamp(10px, 2vw, 18px);
* margin-bottom: clamp(15px, 3vw, 25px);
* box-shadow: 0 4px 15px rgba(0,0,0,0.1);
* animation: slideInDown 0.8s ease-out;
* }
* .attempts-text {
* font-size: clamp(1rem, 2vw, 1.3rem);
* font-weight: 600;
* color: #666;
* margin-right: clamp(8px, 1.5vw, 12px);
* }
* .attempts-number {
* font-size: clamp(1.5rem, 3vw, 2.2rem);
* font-weight: bold;
* color: #667eea;
* text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
* transition: all 0.3s ease;
* display: inline-block;
* }
* .attempts-number.warning {
* color: #ff6b35;
* animation: warningPulse 0.5s ease-in-out;
* }
* .attempts-number.danger {
* color: #e74c3c;
* animation: dangerShake 0.6s ease-in-out;
* }
* @keyframes warningPulse {
* 0%, 100% { transform: scale(1); }
* 50% { transform: scale(1.2); }
* }
* @keyframes dangerShake {
* 0%, 100% { transform: translateX(0); }
* 10%, 30%, 50%, 70%, 90% { transform: translateX(-3px); }
* 20%, 40%, 60%, 80% { transform: translateX(3px); }
* }
* @keyframes slideInDown {
* 0% {
* transform: translateY(-100%);
* opacity: 0;
* }
* 100% {
* transform: translateY(0);
* opacity: 1;
* }
* }
* .instruction {
* text-align: center;
* background: rgba(255,255,255,0.9);
* padding: clamp(12px, 2.5vw, 18px);
* border-radius: clamp(8px, 1.5vw, 12px);
* margin-bottom: clamp(20px, 4vw, 35px);
* box-shadow: 0 4px 15px rgba(0,0,0,0.1);
* font-weight: 500;
* font-size: clamp(0.95rem, 1.8vw, 1.1rem);
* }
* #form {
* background: white;
* padding: clamp(18px, 3vw, 30px);
* border-radius: clamp(12px, 2vw, 18px);
* box-shadow: 0 8px 25px rgba(0,0,0,0.15);
* display: flex;
* gap: clamp(12px, 2.5vw, 20px);
* margin-bottom: clamp(20px, 4vw, 35px);
* animation: slideInUp 0.8s ease-out 0.2s both;
* position: relative;
* overflow: hidden;
* }
* #form::before {
* content: '';
* position: absolute;
* top: -50%;
* left: -50%;
* width: 200%;
* height: 200%;
* background: linear-gradient(45deg, transparent, rgba(255,255,255,0.1), transparent);
* transform: rotate(45deg);
* animation: shimmer 3s infinite;
* }
* @keyframes shimmer {
* 0% { transform: translateX(-100%) translateY(-100%) rotate(45deg); }
* 100% { transform: translateX(100%) translateY(100%) rotate(45deg); }
* }
* @keyframes slideInUp {
* 0% {
* transform: translateY(100%);
* opacity: 0;
* }
* 100% {
* transform: translateY(0);
* opacity: 1;
* }
* }
* #input {
* flex: 1;
* padding: clamp(12px, 2.5vw, 18px);
* border: 2px solid #e0e0e0;
* border-radius: 8px;
* font-size: clamp(1rem, 2.2vw, 1.4rem);
* text-align: center;
* letter-spacing: clamp(4px, 1.2vw, 10px);
* font-weight: bold;
* transition: all 0.3s ease;
* position: relative;
* box-sizing: border-box;
* }
* #input:focus {
* outline: none;
* border-color: #667eea;
* box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
* transform: scale(1.02);
* animation: inputGlow 2s ease-in-out infinite;
* }
* #input.shake {
* animation: inputShake 0.5s ease-in-out;
* }
* #input.success {
* animation: inputSuccess 0.8s ease-in-out;
* }
* @keyframes inputGlow {
* 0%, 100% { box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1); }
* 50% { box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.3); }
* }
* @keyframes inputShake {
* 0%, 100% { transform: translateX(0); }
* 10%, 30%, 50%, 70%, 90% { transform: translateX(-5px); }
* 20%, 40%, 60%, 80% { transform: translateX(5px); }
* }
* @keyframes inputSuccess {
* 0% { background-color: #ffffff; }
* 50% { background-color: #d4edda; }
* 100% { background-color: #ffffff; }
* }
* #input:disabled {
* background-color: #f5f5f5;
* color: #999;
* }
* button {
* padding: clamp(12px, 2.5vw, 18px) clamp(20px, 4vw, 30px);
* background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
* color: white;
* border: none;
* border-radius: 8px;
* font-size: clamp(0.95rem, 2vw, 1.2rem);
* font-weight: bold;
* cursor: pointer;
* transition: all 0.3s ease;
* position: relative;
* overflow: hidden;
* box-sizing: border-box;
* }
* button::before {
* content: '';
* position: absolute;
* top: 50%;
* left: 50%;
* width: 0;
* height: 0;
* background: rgba(255, 255, 255, 0.2);
* border-radius: 50%;
* transform: translate(-50%, -50%);
* transition: width 0.6s, height 0.6s;
* }
* button:hover {
* transform: translateY(-2px) scale(1.03);
* box-shadow: 0 6px 20px rgba(102, 126, 234, 0.3);
* }
* button:hover::before {
* width: 300px;
* height: 300px;
* }
* button:active {
* transform: translateY(-1px) scale(1.02);
* animation: none;
* }
* @keyframes buttonPulse {
* 0%, 100% { box-shadow: 0 8px 25px rgba(102, 126, 234, 0.4); }
* 50% { box-shadow: 0 12px 35px rgba(102, 126, 234, 0.6); }
* }
* #logs {
* background: white;
* border-radius: clamp(12px, 2vw, 18px);
* padding: clamp(15px, 3vw, 25px);
* box-shadow: 0 8px 25px rgba(0,0,0,0.15);
* max-height: clamp(300px, 50vh, 500px);
* overflow-y: auto;
* animation: slideInUp 0.8s ease-out 0.4s both;
* }
* .log-entry {
* padding: clamp(12px, 2.5vw, 18px) clamp(15px, 3vw, 22px);
* margin-bottom: clamp(10px, 2vw, 15px);
* border-radius: clamp(6px, 1.2vw, 10px);
* background: #ffffff;
* border-left: clamp(3px, 0.6vw, 5px) solid #6c757d;
* font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
* font-size: clamp(1rem, 2vw, 1.3rem);
* font-weight: 600;
* color: #2c3e50;
* line-height: 1.5;
* letter-spacing: clamp(0.3px, 0.8vw, 0.8px);
* transition: all 0.3s ease;
* animation: logSlideIn 0.5s ease-out;
* transform-origin: left center;
* box-shadow: 0 2px 8px rgba(0,0,0,0.08);
* }
* @keyframes logSlideIn {
* 0% {
* opacity: 0;
* transform: translateX(-100%) scale(0.8);
* }
* 50% {
* transform: translateX(10%) scale(1.1);
* }
* 100% {
* opacity: 1;
* transform: translateX(0) scale(1);
* }
* }
* .log-entry:hover {
* background: #f1f3f4;
* transform: translateX(3px);
* box-shadow: 0 4px 12px rgba(0,0,0,0.12);
* }
* .log-entry.homerun {
* background: linear-gradient(135deg, #28a745, #20c997);
* color: white;
* border-left: 4px solid #155724;
* animation: homerunSuccess 1s ease-out;
* text-align: center;
* font-weight: bold;
* }
* .log-entry.game-over {
* background: linear-gradient(135deg, #dc3545, #c82333);
* color: white;
* border-left: 4px solid #721c24;
* text-align: center;
* animation: gameOverShake 1s ease-in-out;
* }
* .restart-btn {
* width: 100%;
* margin-top: 20px;
* padding: 15px;
* background: linear-gradient(135deg, #28a745, #20c997);
* font-size: 1.2rem;
* border-radius: 10px;
* animation: restartBounce 3s ease-in-out infinite;
* }
* .restart-btn:hover {
* background: linear-gradient(135deg, #218838, #1e7e34);
* animation: none;
* }
* @keyframes homerunSuccess {
* 0% {
* transform: scale(0.95) translateY(10px);
* opacity: 0;
* }
* 60% {
* transform: scale(1.02) translateY(-2px);
* opacity: 1;
* }
* 100% {
* transform: scale(1) translateY(0);
* opacity: 1;
* }
* }
* @keyframes gameOverShake {
* 0%, 100% { transform: translateX(0); }
* 10%, 30%, 50%, 70%, 90% { transform: translateX(-5px); }
* 20%, 40%, 60%, 80% { transform: translateX(5px); }
* }
* @keyframes restartBounce {
* 0%, 100% { transform: translateY(0) scale(1); }
* 50% { transform: translateY(-2px) scale(1.01); }
* }
* /\* 스크롤바 스타일링 \*/
* #logs::-webkit-scrollbar {
* width: 8px;
* }
* #logs::-webkit-scrollbar-track {
* background: #f1f1f1;
* border-radius: 10px;
* }
* #logs::-webkit-scrollbar-thumb {
* background: #888;
* border-radius: 10px;
* }
* #logs::-webkit-scrollbar-thumb:hover {
* background: #555;
* }
* /\* 상단 설정 패널 \*/
* .settings-panel {
* background: white;
* padding: 20px;
* margin: 20px auto;
* max-width: min(95vw, 1200px);
* border-radius: 15px;
* box-shadow: 0 4px 20px rgba(0,0,0,0.1);
* }
* .settings-panel h3 {
* margin: 0 0 15px 0;
* text-align: center;
* color: #667eea;
* font-size: 1.3rem;
* font-weight: bold;
* }
* .settings-row {
* display: flex;
* flex-wrap: wrap;
* gap: 20px;
* align-items: end;
* justify-content: center;
* }
* .setting-item {
* display: flex;
* flex-direction: column;
* align-items: center;
* min-width: 120px;
* }
* .setting-item label {
* font-weight: 600;
* margin-bottom: 5px;
* color: #333;
* font-size: 0.9rem;
* text-align: center;
* }
* .setting-item input[type="number"] {
* width: 80px;
* padding: 8px;
* border: 2px solid #e0e0e0;
* border-radius: 5px;
* font-size: 0.95rem;
* text-align: center;
* transition: border-color 0.3s ease;
* }
* .setting-item input[type="number"]:focus {
* outline: none;
* border-color: #667eea;
* }
* .checkbox-label {
* display: flex !important;
* align-items: center;
* margin-top: 5px !important;
* margin-bottom: 0 !important;
* font-weight: normal !important;
* cursor: pointer;
* font-size: 0.85rem;
* }
* .checkbox-label input[type="checkbox"] {
* margin-right: 5px;
* width: 16px;
* height: 16px;
* cursor: pointer;
* }
* .apply-btn {
* padding: 12px 25px;
* background: linear-gradient(135deg, #28a745, #20c997);
* color: white;
* border: none;
* border-radius: 8px;
* font-size: 1rem;
* font-weight: bold;
* cursor: pointer;
* transition: all 0.3s ease;
* margin-top: 0;
* }
* .apply-btn:hover {
* background: linear-gradient(135deg, #218838, #1e7e34);
* transform: translateY(-2px);
* box-shadow: 0 5px 15px rgba(40,167,69,0.3);
* }
* /\* 타이머 표시 \*/
* .timer-display {
* position: fixed;
* top: 50%;
* right: 20px;
* transform: translateY(-50%);
* background: rgba(255,255,255,0.95);
* padding: 15px 20px;
* border-radius: 25px;
* box-shadow: 0 4px 15px rgba(0,0,0,0.15);
* font-weight: bold;
* font-size: 1.2rem;
* color: #333;
* z-index: 999;
* display: none;
* border: 3px solid #667eea;
* }
* .timer-display.warning {
* background: rgba(255,107,53,0.9);
* color: white;
* animation: timerWarning 1s ease-in-out infinite;
* }
* .timer-display.danger {
* background: rgba(231,76,60,0.9);
* color: white;
* animation: timerDanger 0.5s ease-in-out infinite;
* }
* @keyframes timerWarning {
* 0%, 100% { transform: scale(1); }
* 50% { transform: scale(1.05); }
* }
* @keyframes timerDanger {
* 0%, 100% { transform: scale(1); }
* 50% { transform: scale(1.1); }
* }
* /\* 반응형 디자인 \*/
* /\* 작은 화면 (900px 이하) - 설정 패널 세로 정렬 \*/
* @media (max-width: 900px) {
* .settings-row {
* flex-direction: column;
* gap: 15px;
* }
* .setting-item {
* width: 100%;
* max-width: 250px;
* }
* .setting-item input[type="number"] {
* width: 100%;
* max-width: 150px;
* }
* }
* /\* 태블릿 (800px 이하) \*/
* @media (max-width: 800px) {
* .settings-panel {
* margin: 15px;
* padding: 15px;
* }
* .main-content {
* max-width: 95vw;
* padding: 20px;
* }
* #form {
* flex-direction: column;
* gap: 15px;
* }
* #input {
* letter-spacing: 6px;
* }
* }
* /\* 모바일 (600px 이하) \*/
* @media (max-width: 600px) {
* .settings-panel {
* margin: 10px;
* padding: 12px;
* }
* .settings-panel h3 {
* font-size: 1.1rem;
* margin-bottom: 10px;
* }
* .setting-item label {
* font-size: 0.8rem;
* }
* .checkbox-label {
* font-size: 0.75rem;
* }
* .apply-btn {
* padding: 10px 20px;
* font-size: 0.9rem;
* }
* .main-content {
* padding: 15px;
* }
* #form {
* flex-direction: column;
* gap: 12px;
* padding: 20px;
* }
* #input {
* letter-spacing: 4px;
* font-size: 1.1rem;
* }
* .timer-display {
* right: 10px;
* top: 40%;
* font-size: 1rem;
* padding: 12px 16px;
* border-width: 2px;
* }
* }
* /\* 아주 작은 화면 (400px 이하) \*/
* @media (max-width: 400px) {
* .settings-panel {
* margin: 5px;
* padding: 10px;
* }
* .settings-row {
* gap: 10px;
* }
* .setting-item {
* min-width: 100px;
* }
* .setting-item input[type="number"] {
* width: 70px;
* padding: 6px;
* font-size: 0.9rem;
* }
* .main-content {
* padding: 10px;
* }
* .attempts-counter {
* padding: 12px;
* }
* .instruction {
* padding: 12px;
* font-size: 0.9rem;
* }
* #form {
* padding: 15px;
* gap: 10px;
* }
* #input {
* padding: 12px;
* font-size: 1rem;
* letter-spacing: 3px;
* }
* button {
* padding: 12px 18px;
* font-size: 0.95rem;
* }
* }