게임 만들기
두더지 게임

HTML CSS JavaScript

장 준

클라우드(AWS)활용 자바/스프링 개발 부트캠프 (8회차)

INDEX

목차

게임 설명

게임 설명 및 규칙

1. 게임 시작 버튼을 눌러 게임 시작
2. 게임이 시작되면 구멍에 랜덤으로 두더지 등장
3. 두더지(이미지)를 클릭 시 점수+3
4. 두더지와 구멍 테두리 사이나 갈색 판 클릭 시 점수-1
5. 게임 진행 중 초기화 버튼을 눌러 초기화 가능
6. 게임이 종료되면 상태 창에 최종 점수 표시

게임 구상

01

필드 생성

- 게임 필드
- 두더지, 구멍 9개씩
- 점수 창
- 시간 창
- 상태 창
- 시작 버튼
- 초기화 버튼

02

랜덤 위치와 점수

- 두더지 위치랜덤으로 변경
- 두더지 클릭 시 점수+3
- 두더지 이외의 것을 클릭 시 점수-1
- 최소 점수 0점

03

게임 시작 및 종료, 초기화

- 게임 시작 버튼을 누르면 게임 시작
- 초기화 버튼을 누르면 초기화 가능
- 제한 시간 30초가 지나면 게임 종료

✓ 주요 기능 설명

```
function createHoles() {
    gameField.innerHTML = ""; // 게임 필드 초기화
    holes = []; // holes 배열 초기화
    for (let i = 0; i < holeCnt; i++) { // 구멍 9개 생성
        // 구멍 요소 <div class="mole-hole" data-index="i">
        const holeEl = document.createElement("div");
        holeEl.classList.add("mole-hole"); // 클래스 추가
        holeEl.dataset.index = i; // 구멍 인덱스 설정
        // 두더지 요소
        const moleDivEl = document.createElement("div");
        moleDivEl.classList.add("mole"); // 클래스 추가
        const moleImgEl = document.createElement("img");
        moleImgEl.src = "img/mole.png";
        moleImgEl.style.width = '85px';
        moleImgEl.style.height = '100px';
        moleDivEl.appendChild(moleImgEl); // 두더지div 요소에 두더지img 요소 넣기
        holeEl.appendChild(moleDivEl); // 구멍 요소에 두더지 결합 요소 넣기
        // 결합된 구멍 요소를 클릭시 clickMole 실행
        holeEl.addEventListener('click', clickMole);
        gameField.appendChild(holeEl); // 게임 필드에 구멍+두더지 요소를 넣기
        holes.push(holeEl); // 구멍 배열에 구멍 요소 추가
    }
}
```

Tip!

```
<div id="gameField"> grid
  <div class="mole-hole" data-index="0">
    <div class="mole"> flex
      <img src="img/mole.png" style="widt
```

```
<div id="gameField"> grid
  <div class="mole-hole mole-up" da
    <div class="mole"> flex == $0
      <img src="img/mole.png" style=
```

구멍 생성

```
display: grid;
/* 세로줄(열) 3개 (너비 균등) */
grid-template-columns: repeat(3, 1fr);
/* 가로줄(행) 3개 (높이 균등) */
grid-template-rows: repeat(3, 1fr);
```

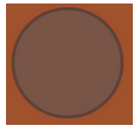
- 게임 필드, 구멍 배열 초기화
- 구멍 요소 생성 후 클래스 추가 및 인덱스 설정
- 두더지 요소 생성 후 클래스 추가
- 이미지 요소 생성 후 이미지 지정 및 스타일 설정
- 게임 필드에 구멍 요소와 두더지 요소 결합
- 결합된 구멍 요소 클릭 시 moleClick 함수 실행
- 구멍 배열에 구멍 요소 추가

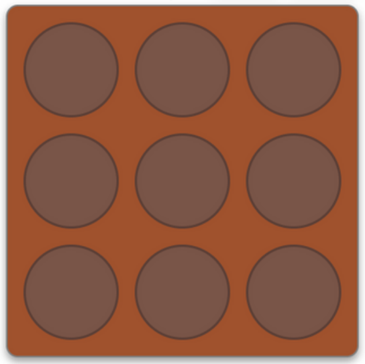
```
function showMole(holeEl) {
    holeEl.classList.add("mole-up");
}
```

```
function hideMole(holeEl) {
    holeEl.classList.remove("mole-up");
}
```

두더지 등장 / 퇴장

- ‘mole-up’ 클래스 추가 / 제거
- css 적용





✓ 주요 기능 설명

```
function popMoleCycle() {
    if (!isGameActive) return;
    // 이미 올라와 있는 두더지가 있다면 이번 주기 건너뜀
    if (document.querySelector('.mole-up')) return;
    // 두더지가 나올 다음 구멍을 랜덤으로 지정
    const nextHole = pickRandomHole();
    showMole(nextHole); // 랜덤 위치에 두더지 출현
    statusDisplay.textContent = "두더지 등장!";
    statusDisplay.style.color = "orange";

    moleTimeout = setTimeout(() => {
        hideMole(nextHole);
        moleTimeout = null; // 타임아웃 해제 표시
    }, moleHide); // 0.8초
}
```

```
.mole { /* 두더지 */
    bottom: -100%; /* 밑으로 내려가 있는 상태 */
    transition: bottom 0.2s ease-out; /* 두더지 움직임 */

/* mole-hole과 mole-up 동시에 가진 요소 안에 자식요소인 mole 선택 */
.mole-hole.mole-up .mole { /* 구멍에서 올라온 두더지 요소 */
    bottom: 0%; /* 완전히 올라온 상태 */
```

두더지 출현 주기

- 게임이 비활성화 상태인지 확인 (활성화 상태면 함수 종료)
- 이미 올라와 있는 두더지가 있다면 패스
- 두더지가 나올 다음 구멍을 랜덤으로 지정
- 랜덤 위치에 두더지 등장
- 상태창의 텍스트를 '두더지 등장!' 으로 변경 및 글자를 주황색으로 변경
- 0.8초 후에 두더지 퇴장

두더지 등장!

두더지 애니메이션 CSS

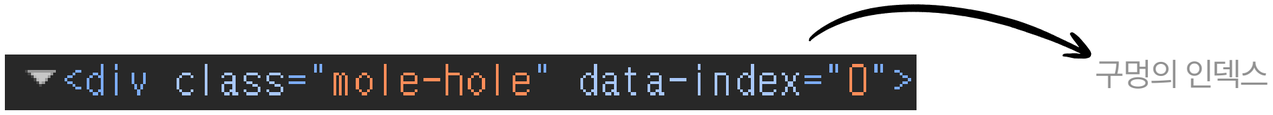
- 두더지는 밑으로 내려가 있는 상태 (-100%)
- bottom 속성이 변할 때, 0.2초 동안 부드럽게 감속하며 변화하도록 만드는 설정
- ‘mole-up’ 클래스가 추가 되면서 내려가 있던 두더지가 제자리로 오면서 위로 올라오는 듯한 느낌을 줌

주요 기능 설명

구멍 랜덤 선택

```
function pickRandomHole() {
    let randomIndex;
    do { // 1회 랜덤 생성 후 이전 인덱스와 랜덤 인덱스가 같다면 랜덤 인덱스 재생성
        randomIndex = Math.floor(Math.random() * holeCnt);
    } while (randomIndex === lastHoleIndex);
    lastHoleIndex = randomIndex; // 다음 비교를 위해 랜덤 인덱스 저장
    return holes[randomIndex]; // holes 배열의 랜덤 인덱스 값 반환
}
```

- 랜덤 인덱스 변수 선언
- do-while 반복문과 Math 함수를 사용해 랜덤 인덱스 생성
- 랜덤 인덱스를 구멍 배열 인덱스로 반환



✓ 주요 기능 설명

```
function clickMole(event) {
    if (!isGameActive) return;
    // 클릭한 <div class="mole-hole"> 컨테이너 자체
    const hole = event.currentTarget;
    // 두더지가 올라와 있는 상태인지 확인
    if (!hole.classList.contains('mole-up')) {
        miss();
        return;
    }
    // 클릭된 요소가 실제 IMG 태그라면
    if (event.target.tagName === 'IMG') {
        // 두더지가 올라와 있고, 이미지 자체 클릭
        hit();
        hideMole(hole); // 두더지 숨김
    } else {
        // 두더지는 올라와 있지만, 이미지 주변 여백 클릭
        miss();
    }
}
gameField.addEventListener('click', (event) => {
    // 클릭 대상이 gameField 자체이고 (구멍이 아닌 흙 바닥), 게임이 진행 중일 때만 처리
    if (event.target.id === 'gameField' && isGameActive) {
        miss();
    }
});
```

```
function hit() {
    score += 3;
    scoreDisplay.textContent = score;
    statusDisplay.textContent = 'Hit! (+3점)';
    statusDisplay.style.color = 'blue';
}
```

```
function miss() {
    if (score > 0) score--;
    scoreDisplay.textContent = score;
    statusDisplay.textContent = 'Miss... (-1점)';
    statusDisplay.style.color = 'red';
}
```

두더지 클릭, Hit & Miss 처리

- 게임이 활성화 상태인지 확인 (비활성화 상태면 함수 종료)
- 두더지가 올라와 있는 상태인지 확인 (1차 거름망) 아니라면 miss() 실행
- 두더지가 올라와 있고 이미지를 클릭했다면 hit() 실행
- 두더지가 올라와 있지만, 이미지 이외의 게임 필드를 클릭했다면 miss() (최소 점수 0점)

✓ 주요 기능 설명

```
startBtn.addEventListener('click',startGame);
function startGame() {
    if (isGameActive) return;
    isGameActive = true; // 게임 활성화
    score = 0;
    scoreDisplay.textContent = score;
    // 모든 타이머 초기화
    clearInterval(gameTimer);
    clearInterval(moleInterval);
    clearTimeout(moleTimeout);
    startBtn.textContent = "진행중..."
    startBtn.disabled = true;
    resetBtn.disabled = false;
    statusDisplay.textContent = `${gameTime}초 동안 두더지를 잡으세요!`;
    statusDisplay.style.color = "green";
    // 시간 타이머 시작
    startTimer();
    // 1.1초마다 popMoleCycle 함수 실행
    moleInterval = setInterval(popMoleCycle, moleCycle);
}
```

```
function startTimer() {
    // 게임이 시작될 때 남은시간을 게임 설정 시간으로 초기화
    remainTime = gameTime;
    // 남은 시간 출력
    timeDisplay.textContent = remainTime;
    // 1초마다 남은 시간 1 감소 후 남은 시간 업데이트
    gameTimer = setInterval(() => {
        remainTime--;
        timeDisplay.textContent = remainTime;
        if (remainTime <= 0) { // 시간이 0이되면 게임 종료
            endGame();
        }
    }, 1000); // 1초
}
```

30초 동안 두더지를 잡으세요!

게임 시작, 타이머

- 게임이 활성화 상태인지 확인 (비활성화 상태면 함수 종료)
- 게임 활성화 및 모든 타이머 초기화
- 점수 초기화 후 출력
- 초기화 버튼 활성화
- 시작 버튼의 텍스트를 '진행중...' 으로 변경 및 버튼 비활성화
- 상태창의 텍스트를 'gameTime(변수) 동안 두더지를 잡으세요!' 로 변경 및 글자를 초록색으로 변경
- 시간 타이머 함수 실행
- 1.1초 마다 popMoleCycle 함수 실행

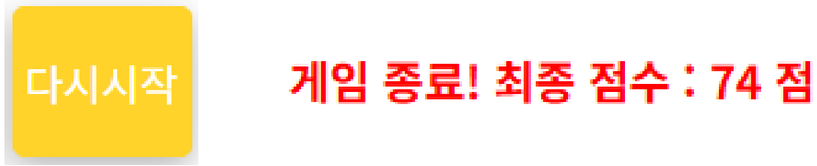
- 남은 시간을 게임 설정 시간으로 초기화 후 출력
- 1초마다 남은 시간 1 감소 후 남은 시간 업데이트
- 남은 시간이 0이하가 되면 게임 종료

✓ 주요 기능 설명

```
function endGame() {
    if (!isGameActive) return;
    isGameActive = false; // 게임 비활성화
    // 모든 타이머 초기화
    clearInterval(gameTimer);
    clearInterval(moleInterval);
    clearTimeout(moleTimeout);
    // 구멍에 모든 두더지 숨기기
    holes.forEach(hideMole);
    startBtn.textContent = "다시시작";
    startBtn.disabled = false;
    statusDisplay.textContent = `게임 종료!  최종 점수 : ${score} 점`;
    statusDisplay.style.color = "red";
}
```

게임 종료

- 게임이 비활성화 상태인지 확인 (활성화 상태면 함수 종료)
- 게임 비활성화
- 모든 타이머 초기화
- 모든 두더지 숨기기
- 시작 버튼의 텍스트를 '다시 시작'으로 변경 및 버튼 활성화
- 상태 창의 텍스트를 '게임 종료! 최종 점수 : score(변수) 점' 으로 변경 및 글자를 빨간색으로 변경

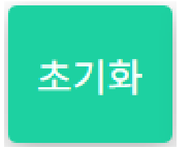


주요 기능 설명

```
resetBtn.addEventListener('click',resetGame);
function resetGame() {
    //모든 타이머 초기화
    clearInterval(gameTimer);
    clearInterval(moleInterval);
    clearTimeout(moleTimeout);
    // 구멍에 모든 두더지 숨기기
    holes.forEach(hideMole);
    isGameActive = false;
    score = 0;
    scoreDisplay.textContent = score;
    remainTime = gameTime;
    timeDisplay.textContent = remainTime;
    startBtn.textContent = "게임시작";
    startBtn.disabled = false;
    resetBtn.disabled = false;
    statusDisplay.textContent = "게임시작 버튼을 눌러 시작하세요!";
    statusDisplay.style.color = "black";
}
```

게임 초기화

- 모든 타이머 초기화
- 모든 두더지 숨기기
- 게임 비활성화
- 점수 초기화 후 출력
- 남은 시간을 게임 설정 시간으로 초기화 후 출력
- 시작 버튼의 텍스트를 '게임 시작'으로 변경
- 시작 버튼과 초기화 버튼 활성화
- 상태창의 텍스트를 '게임시작 버튼을 눌러 게임을 시작하세요!' 로 변경 및 글자를 검정색으로 변경



게임시작 버튼을 눌러 시작하세요!

평가 및 느낀점

setInterval과 setTimeout을 통해 두더지 등장-퇴장 주기와 게임 시간을 제어하는 비동기 처리의 중요성을 느꼈습니다. 그리고 CSS Transition을 활용한 애니메이션 구현과 Hit/Miss를 판정하는 상태 관리 로직을 만들며 프론트엔드 개발에 한층 더 흥미를 느끼게 되었습니다.

감사합니다.