Tic-Tac-Toe

목차

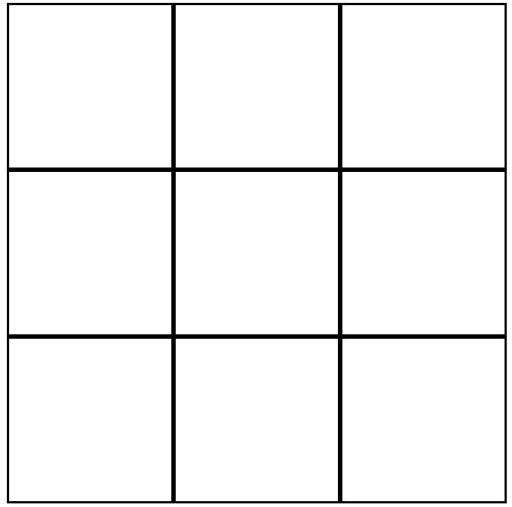
1. HTML

- 3 x 3 구역 분할 : <table> 태그
- 분할된 각 칸에 <button> 추가
- 각 버튼 "ttt-btn" 클래스 추가
- 각 버튼 1 ~ 9 클래스로 구분
- 영역 지정으로 결과 출력 위치 지정 및 재시작 버튼 생성

```
<body>
    <table>
        <tr><td><button class="ttt-btn 1"></button></td>
            <td><button class="ttt-btn 2"></button></td>
            <td><button class="ttt-btn 3"></button></td></tr>
        <tr><td><button class="ttt-btn 4"></button></td>
            <td><button class="ttt-btn 5"></button></td>
            <td><button class="ttt-btn 6"></button></td></tr>
        <tr><td><button class="ttt-btn 7"></button></td>
            <td><button class="ttt-btn 8"></button></td>
            <td><button class="ttt-btn 9"></button></td></tr>
    </table>

    <div id="result-box">
        <div id="result"></div><button id="replay">replay?</button>
    </div>


    <script src="./tic-tac-toe.js"></script>
```



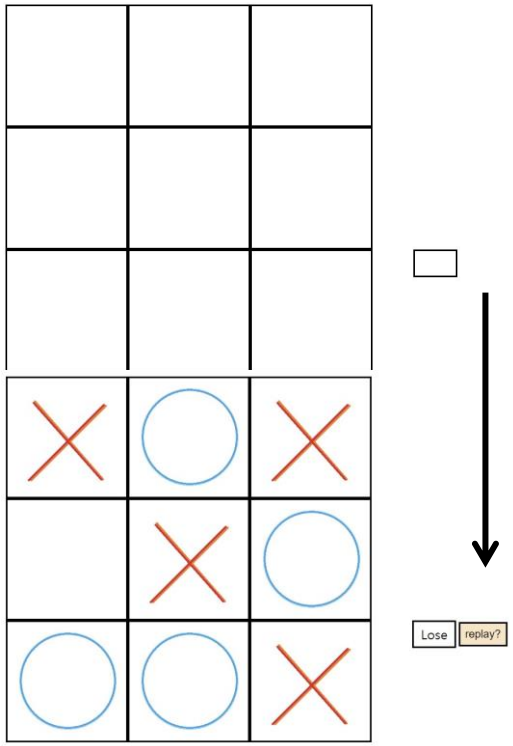
2. CSS

```css
.ttt-o {
    background-image: url(./img/o.jpg);
    background-size: 130px;
    background-repeat: no-repeat;
    background-position: left 8px top 10px;
}

.ttt-x {
    background-image: url(./img/x.jpg);
    background-size: 150px;
    background-repeat: no-repeat;
    background-position: left 1px top 10px;
}
```

```css
#result-box {
    position: absolute;
    top: 300px;
    left: 500px;
    width: 50px;
    height: 30px;
    border: solid;
}
div#result {
    text-align: center;
    line-height: 30px;
}

#replay {
    position: absolute;
    top: 0px;
    left: 55px;
    width: 60px;
    border: solid;
    background-color: bisque;
}
.hide {
    display: none;
}
```

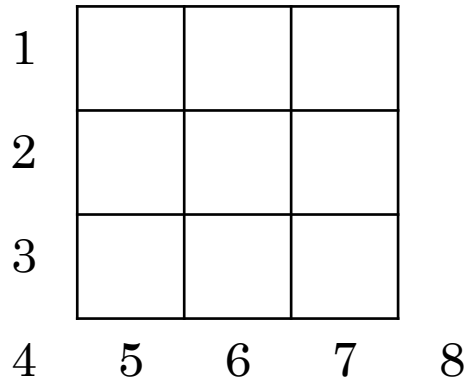


- ttt-o, ttt-x 클래스로 선택된 영역 표시
- Replay버튼에 hide 클래스 추가로 숨김 처리 후 게임 종료 시 해당 클래스제거

3. JavaScript

```
let spaces = [[0,0],[1,0],[2,0],[3,0],[4,0],[5,0],[6,0],[7,0],[8,0],[9,0]];
const lines = [[],[1,2,3],[4,5,6],[7,8,9],[7,5,3],[7,4,1],[8,5,2],[9,6,3],[9,5,1]];
const routeLines = [[],[1,5,8],[1,6],[1,4,7],[2,5],[2,4,6,8],[2,7],[3,4,5],[3,6],[3,7,8]];
const corner = [1,3,7,9];
const cross = [2,4,6,8];
```

```
function checkLineFilled(lineNumber) {
    var fillcount = 0;
    var userFill = 0;
    var comFill = 0;
    lines[lineNumber].forEach(function(element){
        switch (checkSpaceFilled(element)) {
            case 1:
                userFill++;
                fillcount++;
                break;
            case 2:
                comFill++;
                fillcount++;
                break;
            default:
                break;
        }
    });
    return [fillcount,userFill,comFill];
```



- Space 배열으로 각 위치의 상태 관리
  => 0 = 빈칸, 1 = 사용자칸, 2 = 컴퓨터칸

- 라인의 번호지정 후 선택되는 칸을 포함한 라인의 상태 확인

3. JavaScript

```
let btn = document.getElementsByClassName("ttt-btn");
var resultDiv = document.getElementById("result");
var resultBtn = document.getElementById("replay");
resultBtn.classList.add("hide");
```

```
btn[i].onclick = function() {
    if (userTurnFlag) {
        btn[i].classList.add("ttt-o");
        btn[i].classList.add("filled");
        btn[i]["disabled"] = true;
        latestTurnOfUser = parseInt(btn[i].classList[1]);
        spaces[latestTurnOfUser][1] = 1;

        checkLatestLine(latestTurnOfUser);
```

- 특정 요소들에 속성들을 부여하여 호출 및 관리
- 각 버튼에 클릭 시 ttt-o 및 filled클래스를 부여한 후 추가입력 방지를 위한 비활성화
- 사용자의 턴 진행 후 라인의 상태를 점검

3. JavaScript

```
btnCount++;
if (btnCount == 9) {
    gameOver("Draw");
}

if (playcount == 0) {
    turnOfComFirst();
    playcount++;
}
else if (playcount == 1) {
    turnOfComSecond();
    playcount++;
}
else if (playcount > 1 && btnCount < 8) {
    if (nextTargetSpace != 0 && spaces[nextTargetSpace][1] == 0) {
        if (absolTargetSpace != 0 && spaces[absolTargetSpace] == 0) {
            turnOfCom(absolTargetSpace);
        }
        else {
            turnOfCom(nextTargetSpace);
        }
    }
    else {
        turnOfCom(getRandomSpace());
    }
}
```

- 버튼이 채워질 때 마다 확인하여 9개가 모두 채워지면 Draw판정
- 처음 2번의 턴에 승패가 거의 결정됨으로 컴퓨터의 첫번째 두번째 턴의 행동 강제 설정
- 컴퓨터는 일반적으로 사용자의 마지막 위치를 확인하여 방어 여부 확인
- 컴퓨터가 이길 수 있는 상황이 될 경우 방어하지 않고 공격 설정 (정상작동X)
- 아무런 위치가 설정되지 않으면 랜덤위치 선택

3. JavaScript

```
function gameOver(result) {
    for (let j = 0; j < btn.length; j++) {
        btn[j].disabled = true;
    }
    resultDiv.innerHTML = result;
    resultBtn.classList.remove("hide");
}
```

```
resultBtn.onclick = function() {
    playcount = 0;
    btnCount = 0;
    userTurnFlag = true;
    nextTargetSpace = 0;
    absolTargetSpace = 0;

    latestTurnOfUser = 0;
    latestRTurnOfCom = 0;



    window.location.reload();
}
```

- 승패나 무승부가 결정될 경우 모든 버튼의 비활성화, 결과 출력 및 replay버튼 노출 설정

- replay버튼 선택시 변수 초기화 및 페이지 새로고침

감사합니다.