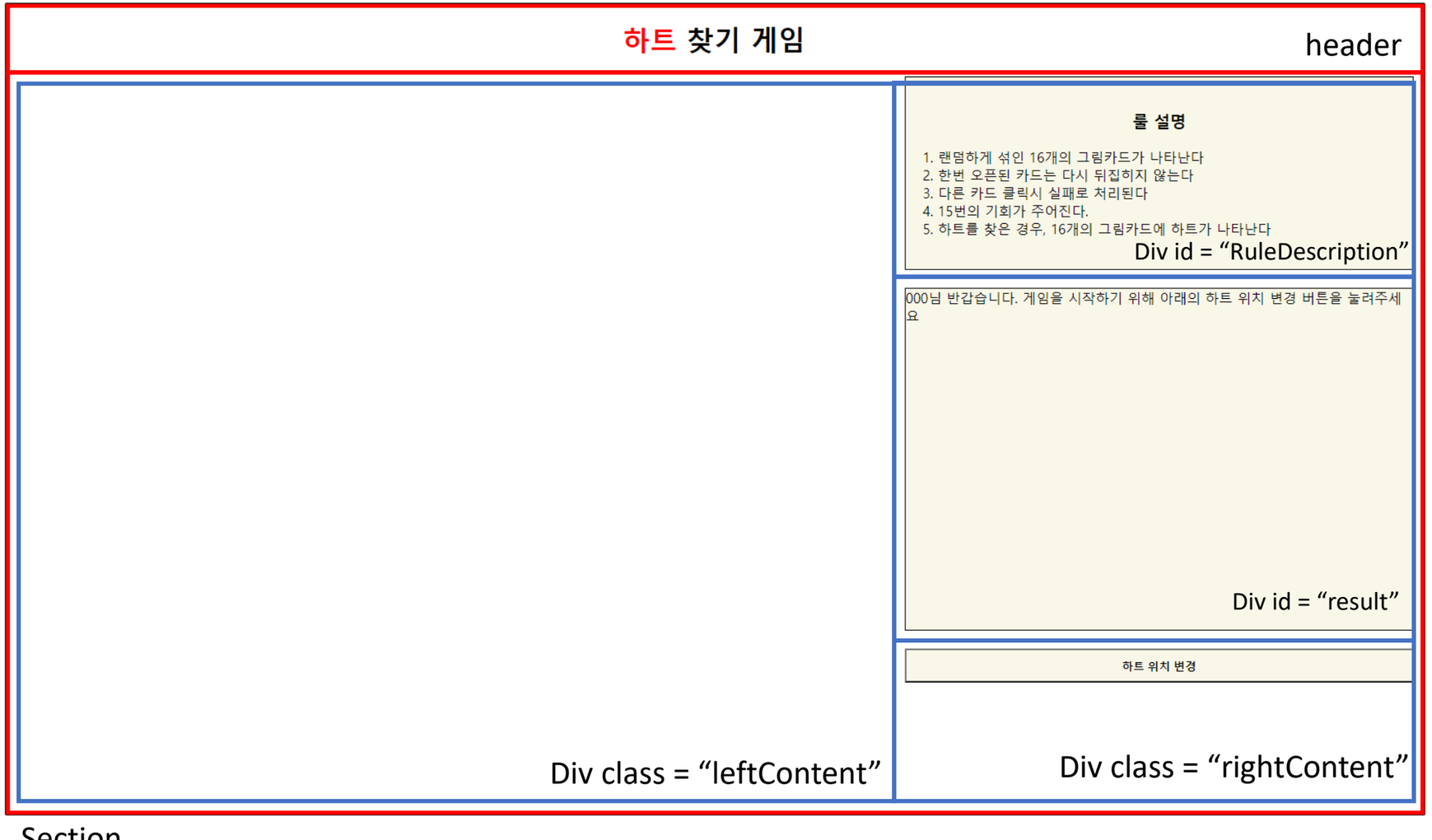
3차 프로그래밍 과제

하트 찾기 게임

이름 : 최경수

메인페이지 UI 구성 : index.html



진짜 게임판 처럼 이름과 룰 설명 결과창 게임 실행버튼이 있으면 좋을것 같아서 오른쪽과 같이 구현해보았습니다.

우선, 시맨틱 구조로 header와 section을 나누었고 더 자세히 section div로 나누었습니다.

그래서 1에선 4번에 달린 버튼을 클릭했을때, 4 * 4 형태의 카드가 나타나도록 했고 2번에는 왼쪽의 그림과 같이 룰 설명이 3번에는 다른 카드 클릭시 > 실패, 클릭횟수, 성공시 > 성공 등 결과를 출력하는 결과창을 만들었습니다. 이번에는 자바 스크립트가 메인이기 때문에 CSS와 html에 대한 설명은 간단하게 하고 넘어가겠습니다.

자바 스크립트 코드는 아래의 사이트를 참고하여 공부한뒤 작성하였습니다.

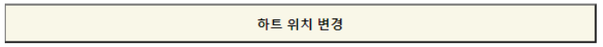
https://www.inflearn.com/course/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8-%EA%B2%8C%EC%9E%84-%EA%B0%9C%EB%B0%9C/lecture/17230?tab=curriculum

이벤트 구현 및 게임동작: indexJs.js

- 16개의 그림(카드) 영역 (4x4) 에서 한 개의 하트를 찾는 게임 작성
  - 배열을 이용하여 하트의 위치 결정(5)
    - 하트면 1, 하트가 아니면 0
  - <하트 위치 변경> 버튼을 누르면 배열의 내용을 섞어 하트 위치를 변경 후
    - 16개의 그림 영역에 카드 뒷면 이미지를 동적으로 생성(2)
  - 동적으로 생성한 카드 뒷면 이미지를 클릭
    - 배열에서의 해당 위치에 저장된 값에 따른 이미지 출력 (3)
      - 하트 또는 다른 이미지
    - 한번 클릭되어 오픈된 카드는 다시 뒤집지 않기(4)
    - 하트가 아닌 다른 이미지 클릭 시 "실패" 메시지 출력(6)
    - 15번을 클릭할 동안 하트를 못찾은 경우에도 "실패" 메시지 출력(7)
    - 하트를 클릭한 경우 성공 메시지와 함께 모든 16개의 카드 그림을 하트로 표시(8)
  - <하트 위치 변경> 버튼 클릭 시 게임 재시작(1)

보고서는 3차 프로그래밍 문제에 나와있는 문제에따라 형광펜 처진 순으로 진행될 예정입니다.

1. <하트 위치 변경> 버튼 클릭 시 게임 재시작




```
<input id="suffle" type="button" value="하트 위치 변경" onclick="cardShuffle()">
```

Html과 CSS로 만들어진 input type = "button"에 대하여 onClick리스너를 달아주었습니다.

```
function cardShuffle() {
    countBtn++;
    var printResult = document.getElementById("result");
    var message;
    //방법 1
    if(countBtn == 1){
        message = "게임을 시작하겠습니다. 위에 기재된 규칙에 따라
시작하십시오" + '<br>';
    }
    else if (countBtn > 1) {
        document.querySelector('#fix').innerHTML = '';
        Clickcount = 0;
        ClickedCard = [];
        Clickremain = 15;
        message = "하트 위치를 변경하였습니다." + '<br>' + '<br>';
    }
    random();
    cardSetting(4, 4);
    printResult.innerHTML = message;
}
```

첫번째, countBtn이라는 전역변수를 설정하여 클릭할때마다 증가하도록 했습니다.

두번째, 앞서 오른쪽 div에 결과값을 출력할 수 있도록 getElementById를 사용했고 결과값을 저장하기 위해 message변수를 설정했습니다.

세번째, 만약 첫번째 클릭일 경우 첫번째 게임을 의미하므로 countBtn = 1이 됩니다. 그래서 countBtn == 1인 경우 message를 왼쪽과 같이 설정했고
두번째 클릭일 경우, 게임을 다하고 클릭하거나 게임 중간에 클릭하거나 첫게임을 안하고 두번누르거나 등 이미 들어와있는 경우이기 때문에 카드 전체 틀이 초기화되면 안된다고 생각했고 Queryselector를 이용하여 내부 카드와 다른 전역 변수들을 초기화 했습니다. 다른 전역 변수는 다른 기능 설명과 함께 설명하겠습니다.

그래서 만약 위의 경우를 거쳤을때, 첫 클릭일 경우 random(), cardSetting을 불렀고 두번째 클릭을 경우 모두 초기화하여 random(), cardSetting를 부르도록 하였습니다.

카드 위치를 무작위로 설정하는 함수

카드 관련 이벤트 및 카드 삽입을 하는 함수

결과창 출력을 위해 innerHtml을 이용하여 messag를 출력하였습니다.

버튼을 클릭하기 전
countBtn == 0

룰 설명

1. 랜덤하게 섞인 16개의 그림카드가 나타난다
2. 한번 오픈된 카드는 다시 뒤집히지 않는다
3. 다른 카드 클릭시 실패로 처리된다
4. 15번의 기회가 주어진다.
5. 하트를 찾은 경우, 16개의 그림카드에 하트가 나타난다

000님 반갑습니다. 게임을 시작하기 위해 아래의 하트 위치 변경 버튼을 눌려주세요

하트 위치 변경

처음 버튼을 클릭했을때
countBtn == 1

룰 설명

1. 랜덤하게 섞인 16개의 그림카드가 나타난다
2. 한번 오픈된 카드는 다시 뒤집히지 않는다
3. 다른 카드 클릭시 실패로 처리된다
4. 15번의 기회가 주어진다.
5. 하트를 찾은 경우, 16개의 그림카드에 하트가 나타난다

게임을 시작하겠습니다. 위에 기재된 규칙에 따라 시작하십시오

하트 위치 변경

두번 이상 버튼을 클릭했을때
countBtn > 1

룰 설명

1. 랜덤하게 섞인 16개의 그림카드가 나타난다
2. 한번 오픈된 카드는 다시 뒤집히지 않는다
3. 다른 카드 클릭시 실패로 처리된다
4. 15번의 기회가 주어진다.
5. 하트를 찾은 경우, 16개의 그림카드에 하트가 나타난다

하트 위치를 변경하였습니다.

하트 위치 변경

2. <하트 위치 변경> 버튼을 누르면 배열의 내용을 섞어 하트 위치를 변경 후 16개의 그림 영역에 카드 뒷면 이미지를 동적으로 생성(2)

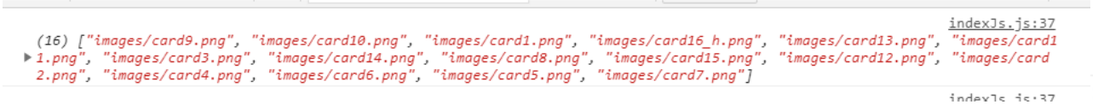

```
var cards = [
    "images/card1.png", "images/card2.png", "images/card3.png", "images/card4.png", "images/card5.png", "images/card6.png", "images/card7.png", "images/card8.png","images/card9.png", "images/card10.png", "images/card11.
png", "images/card12.png","images/card13.png", "images/card14.png", "images/card15.png", "images/card16_h.png"
];
```

파워포인트 기본도형과 아이콘으로 직접 카드를 만들었고 카드의 숫자에 따라 card1~15.png 찾아야 되는 하트는 card16_h.png로 cards배열에 삽입하였습니다.

```
function random() {
    Acards = [];
    Clonecards = cards.slice();
    for (var i = 0; 0 < Clonecards.length; i++) {
        Acards = Acards.concat(Clonecards.splice(Math.floor(Math.random(
) * Clonecards.length), 1));
    }
    console.log(Acards);
}
```

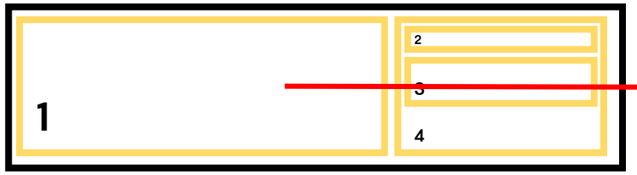
```
var cards = [
    "images/card1.png", "images/card2.png", "images/card3.png", "images/card4.png",
    "images/card5.png", "images/card6.png", "images/card7.png", "images/card8.png",
    "images/card9.png", "images/card10.png", "images/card11.png", "images/card12.png",
    "images/card13.png", "images/card14.png", "images/card15.png", "images/card16_h.png
"
];
//색깔 후보
var Clonecards = cards.slice();
//색깔
var Acards = [];
```

피셔-예이츠 방식으로 0부터 15까지 랜덤한 숫자를 뽑아 Acards라는 함수에 넣었고 cards의 경우 재시작하기 위해선 초기화가 필요한데 이때 초기화를 해놓으면 데이터가 다 날아가기때문에 Clonecards로 복사를 했습니다. 그래서 console.log로 콘솔창에 띄웠을때 다음과 같이 잘 섞이는 것을 볼 수 있었습니다.



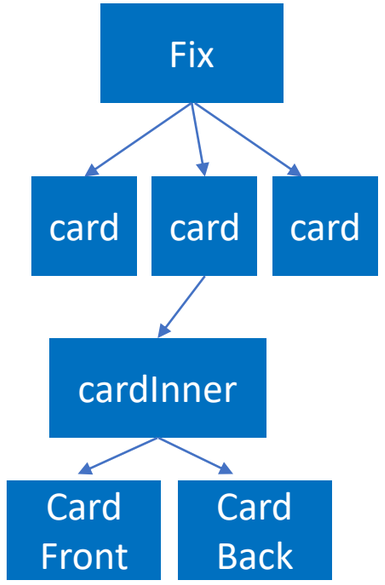
3. 배열에서의 해당 위치에 저장된 값에 따른 이미지 출력 – 하트 또는 다른 이미지

```
for (var i = 0; i < row * col; i++) {
      Clonecards = cards;
      var main = document.getElementById("leftContent");
      var fix = document.getElementById("fix");

      var card = document.createElement('div');
      card.className = 'card';

      var cardInner = document.createElement('div');
      cardInner.className = 'card-inner';

      var cardFront = document.createElement('div');
      cardFront.className = 'card-front';

      var cardBack = document.createElement('div');

      if (Acards[i] == "images/card16_h.png") {
          cardBack.className = 'card-back1';
          cardBack.style.backgroundImage = 'url(' + Acards[i] + ')';
          cardBack.style.backgroundRepeat = 'no-repeat';
          cardBack.style.backgroundSize = 'cover';
      } else {
          cardBack.className = 'card-back';
          cardBack.style.backgroundImage = 'url(' + Acards[i] + ')';
          cardBack.style.backgroundRepeat = 'no-repeat';
          cardBack.style.backgroundSize = 'cover';
      }

      cardInner.appendChild(cardFront);
      cardInner.appendChild(cardBack);
      card.appendChild(cardInner);
      fix.appendChild(card);
      main.appendChild(fix);
```



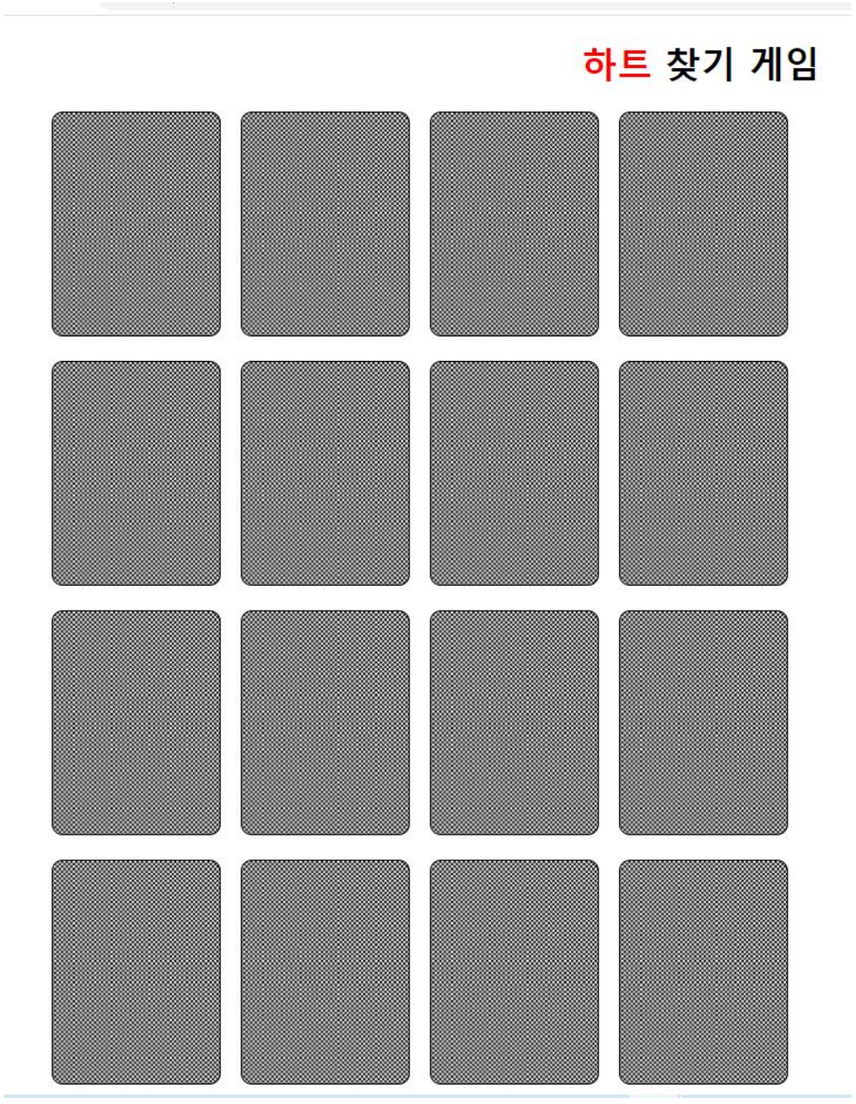

```
<div id="leftContent">
        <div id="fix">
        </div>
    </div>
```

메인 프레임에서 1번의 html코드가 오른쪽이고 이를 동적으로 생성하기 위해선 자바스크립트로 생성되도록 했습니다.

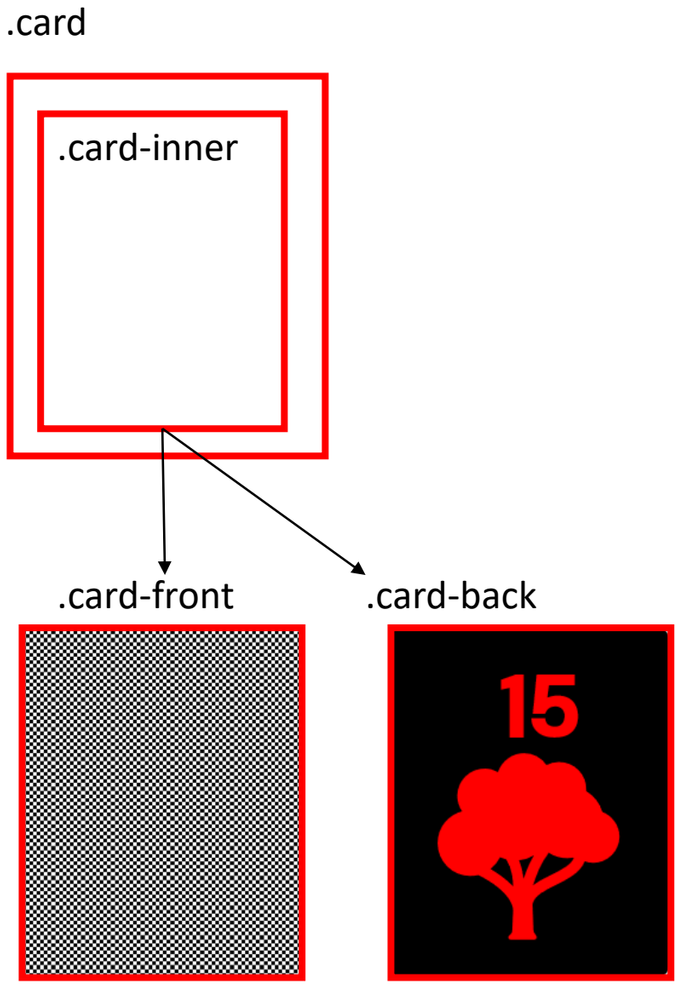


왼쪽과 같이 div를 appen하여 만들도록 했고 div자체로 마치 배열과 테이블처럼 보이게 4 * 4로 보이게 했고 만약 섞은 카드중에서
`if (Acards[i] == "images/card16_h.png")`
이면 className을 card-back1로 아니면 card-back으로 설정하도록 하여 구분했습니다. 그래서 무조건 마우스가 클릭됐을때, 카드가 나타나므로 Css에서 카드의 배경이미지를 설정해주었습니다.

```
▼<div id="leftContent"> == $0
  ▼<div id="fix">
    ▼<div class="card">
      ▼<div class="card-inner">
          <div class="card-front"></div>
          <div class="card-back" style="background-image: url("images/card14.png"); background-repeat: no-repea
          t; background-size: cover;"></div>
        </div>
      </div>
```




```css
.card {
    display: inline-block;
    margin-right: 20px;
    margin-bottom: 20px;
    width: 150px;
    height: 200px;

    perspective: 1000px;
}

.card-inner{
    position: relative;
    width: 100%;
    height: 100%;
    text-align: center;
    transition: transform 0.8s;
    transform-style: preserve-3d;
}

.card.flipped .card-inner{
    transform: rotateY(180deg);
}

.card-front, .card-back,.card-back1{

    position: absolute;
    width: 100%;
    height: 100%;
    border: 1px solid black;
    border-radius: 10px;
    backface-visibility: hidden;
}
.card-front{
    background-image: url(images/cardB.png);
    background-repeat: no-repeat;
    background-size: cover;
}
.card-back, .card-back1{
    transform: rotateY(180deg);
}
```

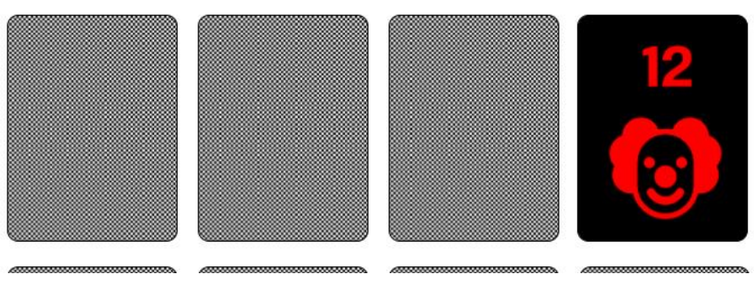


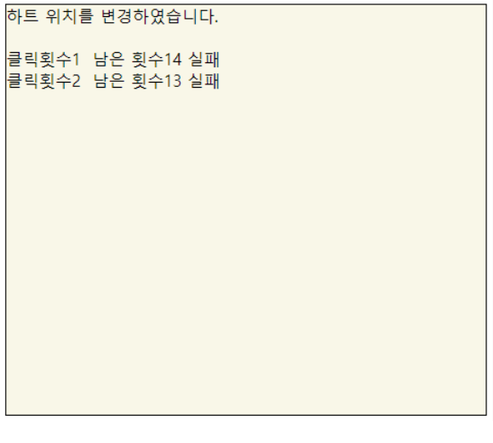
Position을 이용해 카드를 겹치고 transform을 활용해서 카드가 뒤집히도록 했습니다

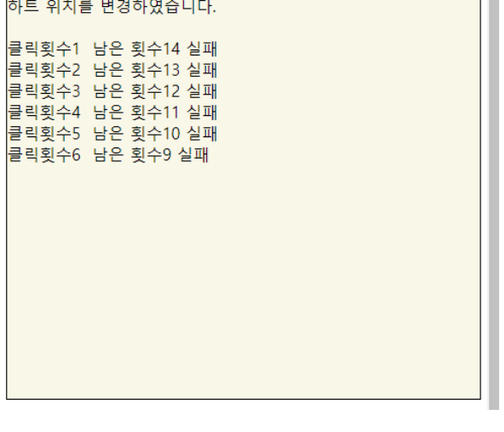
4. 한번 클릭되어 오픈된 카드는 다시 뒤집지 않기

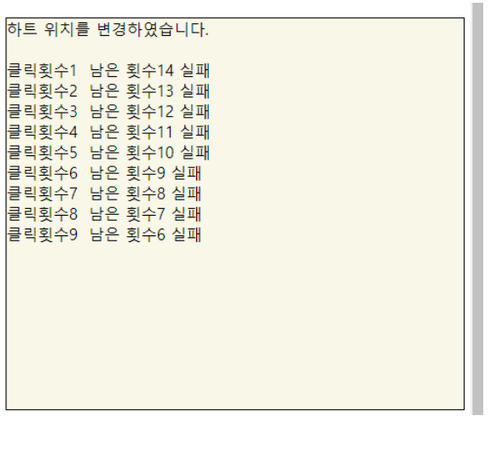
```
cardsClick = true;
        (function (c) {
            c.addEventListener('click', function () {
                if (!ClickedCard.includes(c) && cardsClick) {
                    c.classList.toggle('flipped');
                    Clickcount++;
                    Clickremain -= 1;
                    ClickedCard.push(c);
                    var printResult = document.getElementById("result");
                    message = "클릭횟수" + Clickcount + "  " + "남은 횟수" + Clickremain;
                    printResult.innerHTML += message;
```

True/false로 오픈된 카드에 대한 제한을 두었고 만약 이벤트 발생시 클릭된 카드에 대한 정보는 ClickedCard라는 배열에 저장되도록 했습니다. 그래서 만약 ClickedCard가 있을 경우 !ClickedCard.include(c)는 true가 되고 더 이상 클릭이 되지 않게 됩니다. 또한 횟수 체크를 위해 Clickcount라는 함수를 만들어 카운트 하도록 했고 별도로 남은 횟수를 알려주기 위해 Clickeremain을 15로 설정하여 1씩 빠지도록 했습니다.









5. 배열을 이용하여 하트의 위치 결정
하트면 1, 하트가 아니면 0

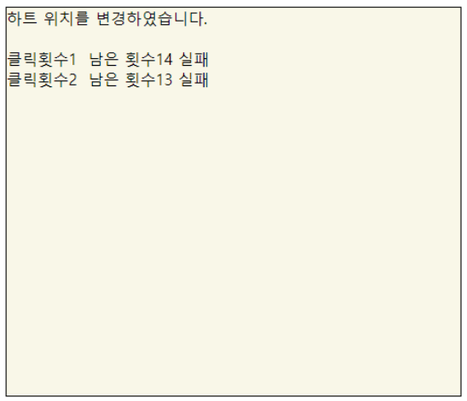
```
for (var j = 0; j < ClickedCard.length; j++) {
                    if (ClickedCard[j].querySelector('.card-back1')) {
                        Clickcard = 1;
                    } else {
                        Clickcard = 0;

                    }
                }
                if (Clickcard == 0) {
                    message = ' ' + "실패" + '<br>';
                    printResult.innerHTML += message;
                }
```

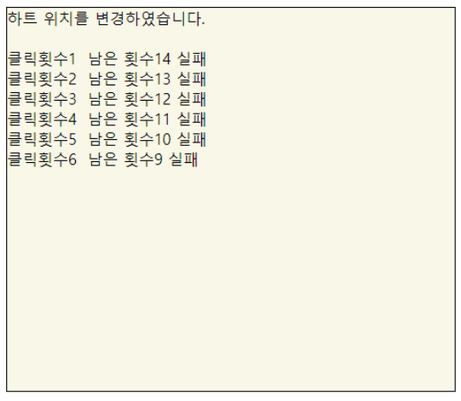
Div생성할때, 클래스 명으로 구분할 수 있도록 했었는데 이것이 나중에 구분하기 위한 밑밥으로 만 약 queryselector 했 을 때 .card-back1이면 하트이므로 변수 clickcard = 1 아니면 clickcard = 0;

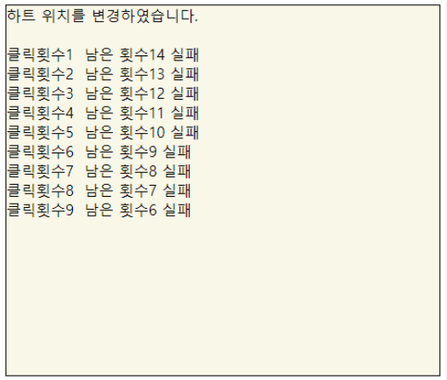
따라 console.log로 찍어보면 아닌 카드 클릭시 0 하트 카드 클릭 시 1 이 나온다.

6. 하트가 아닌 다른 이미지 클릭 시 "실패" 메시지 출력

따라 Clickcard가 0인 경우는 하트카드가 아니니까 0인 경우만 따지 면된다.따라 innerHtml를 통해 결과값을 출력하였다.







7. 15번을 클릭할 동안 하트를 못찾은 경우에도 "실패" 메시지 출력

```
                                if (Clickcount == 15) {
                    if (Clickcard == 1) {
                        cardsClick = false;
                        message = "성공";
                        printResult.innerHTML = message;
                        document.querySelectorAll('.card-back').forEach(function (cardBack, index) {
                            setTimeout(function () {
                                cardBack.style.backgroundImage = 'url(' + "images/card16_h.png" + ')';
                            }, 1000 + 100 * index);
                        });
                        document.querySelectorAll('.card').forEach(function (card, index) {
                            cardsClick = false;
                            setTimeout(function () {
                                card.classList.add('flipped');
                            }, 1000 + 100 * index);
                        });
                    } else {
                        cardsClick = false;
                        message = "실패하셨습니다 다시 도전해주세요";
                        printResult.innerHTML = message;
                    }
                }
```

```
else if (Clickcount < 16 && Clickcard == 1) {
                        cardsClick = false;
                        message = "성공";
                        printResult.innerHTML = message;
                        document.querySelectorAll('.card-back').forEach(function (cardBack, index) {
                            setTimeout(function () {
                                cardBack.style.backgroundImage = 'url(' + "images/card16_h.png" + ')';
                            }, 1000 + 100 * index);
                        });
                        document.querySelectorAll('.card').forEach(function (card, index) {
                            setTimeout(function () {
                                card.classList.add('flipped');
                            }, 1000 + 100 * index);
                        });
                    }
                }
```

성공의 경우

성공과 실패의 경우는 매우 간단하다. 먼저 성공의 경우
첫번째, Clickcount < 16인 동안 Clickcard == 1이면 된다.

```
else if (Clickcount < 16 && Clickcard == 1)
```

이 말은 주어진 15번의 클릭 중에서 하트 카드를 선택하면 된다는 것이다.

두번째, Clickcount == 15 && Clickcard == 1

```
if (Clickcount == 15) {
      if (Clickcard == 1)
```

이 말은 마지막 기회인 15번째 클릭에서 하트카드를 찾았다 라고 생각하면 된다.

실패의 경우

성공과 실패의 경우는 매우 간단하다.
반대로 실패의 경우 단 한가지만 있으면 된다.

두번째, Clickcount == 15 && Clickcard != 1

```
if (Clickcount == 15) {
      if (Clickcard == 1){
    }
} else{
```

이 말은 마지막 기회인 15번째 클릭에서 하트카드를 찾지 못하고 다른 카드를 찾아 결국 주어진 기회동안 하트카드를 찾지 못했음을 나타낸다.

성공의 경우

첫번째

두번째

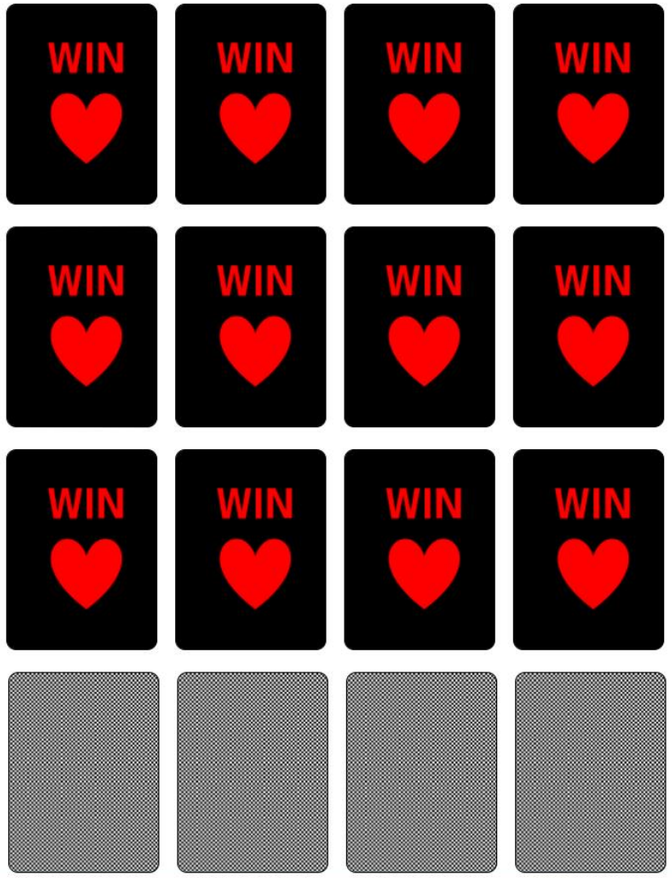
실패의 경우

8. 하트를 클릭한 경우 성공 메시지와 함께 모든 16개의 카드 그림을 하트로 표시

```
document.querySelectorAll('.card-back').forEach(function (cardBack, index) {
                                    setTimeout(function () {
                                        cardBack.style.backgroundImage = 'url(' + "images/card16_h.png" + ')';
                                    }, 1000 + 100 * index);
                                });
                                document.querySelectorAll('.card').forEach(function (card, index) {
                                    cardsClick = false;
                                    setTimeout(function () {
                                        card.classList.add('flipped');
                                    }, 1000 + 100 * index);
                                });
```

이것도 매우 간단했다. queryselectorAll을 이용하여 하트자리를 제외한 모든 .card-back 클래스에 대하여 "images/card16_h.png“ 를 집어 넣으면 되고 setTimeout를 이용하여 하나씩 나타나도록 했다. 밑의 코드는 하나씩 하나씩 카드가 앞면을 보이도록 한것이고 성공한 경우에 모든 카드가 앞면을 보이며 하트 카드로 채워지도록 했다.

하트 찾기 게임



룰 설명

1. 랜덤하게 섞인 16개의 그림카드가 나타난다
2. 한번 오픈된 카드는 다시 뒤집히지 않는다
3. 다른 카드 클릭시 실패로 처리된다
4. 15번의 기회가 주어진다.
5. 하트를 찾은 경우, 16개의 그림카드에 하트가 나타난다

성공

하트 위치 변경

하트 찾기 게임

룰 설명

1. 랜덤하게 섞인 16개의 그림카드가 나타난다
2. 한번 오픈된 카드는 다시 뒤집히지 않는다
3. 다른 카드 클릭시 실패로 처리된다
4. 15번의 기회가 주어진다.
5. 하트를 찾은 경우, 16개의 그림카드에 하트가 나타난다

성공

하트 위치 변경