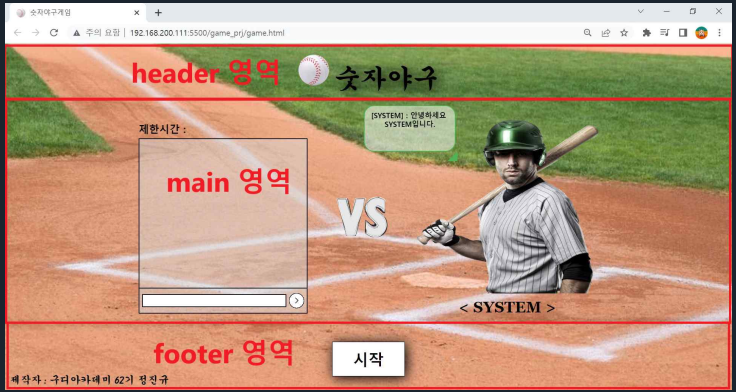
숫자야구

구디아카데미 62기 정진규

화면 레이아웃

01



전체적인 레이아웃을 나눠보자면 header, main, footer 로 나누어서 디자인하였음 (CSS flex레이아웃 속성을 사용)
각각의 영역에서도 flex레이아웃을 사용하여 div태그들을 구조화 시켰다. (가로배치, 세로배치)

```css
/* 배경 + 전체적인 레이아웃 */
#container_space {
  width: 100vw;
  height: 100vh;
  display : flex;
  flex-direction: column;
  background-image: url("./image/baseballround.jpg");
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  position: relative;
}
```

```css
#header_title {
  text-align: center;
  vertical-align: bottom;
  font-family: 'Edu NSW ACT Foundation', cursive;
  display: flex;
  justify-content: center;
}
```

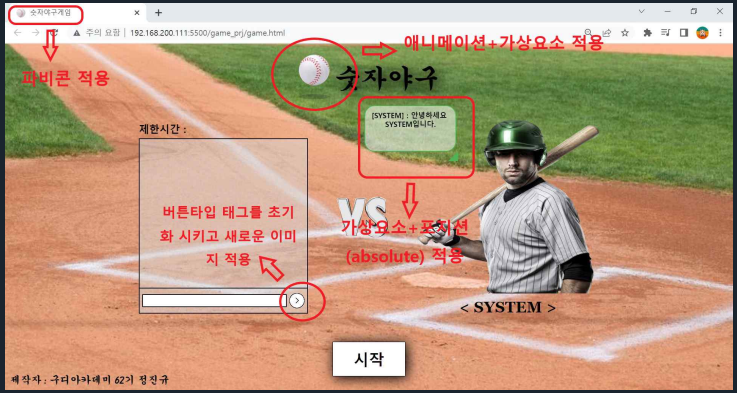
```css
#main_game{
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  padding-top: 50px;
}
```

```css
#footer {
  display: flex;
  align-items: center;
  justify-content: center;
  padding-top: 30px;
}
```

각각의 div에 id값을 지정해 CSS속성을 주었다.

상하좌우 flex속성은 웬만해서는 가운데정렬을 지향하였고, 속성값의 단위는 px과 뷰포트단위(vw, vh) 단위를 주로 사용하였다.

(뷰포트단위는 사용자 브라우저 해상도에 맞는 비율 단위이다.)



html 파비콘을 적용

css애니메이션 효과를 사용해 로고의 야구공이 동적인 느낌을 갖게 하였음

기존의 버튼을 전부 unset시키고 새로운 이미지의 버튼을 만들었음

말풍선과 화살표(가상요소)를 만들어서 포지션 absolute 적용

룰 설명

1. SYSTEM은 3자리 숫자중 각 자릿수가 중복되지 않은 숫자를 생각한다. (0은 제외)
2. USER는 임의의 중복되지않은 숫자 3자리를 입력한다.
3. 숫자도 같으면서 자릿수까지 같으면 strike, 숫자는 같지만 자릿수가 다를경우 ball
4. USER는 SYSTEM에게 3strike를 얻어내면 승리

※제한시간 60초안에 3strike가 나오지 않는다면 SYSTEM승리
※USER는 입력 조건사항을 지키지 않는다면 운영자가 경고메세지를 띄워줌
※USER : 검은색, SYSTEM : 파란색, 운영자 : 빨간색, 초록색

02

숫자야구 주요함수들

- createLi()
- systemNum()
- isDuplicate()
- startTimer()
- autoScrollAndFocus()
- removeLi()
- isDigitZero()
- chatAnswer()

03

주요 함수들(1)

li태그 생성 함수와 자동포커싱 함수

1. li태그를 생성하여 const li 상수에 담는다.
2. li 텍스트 컬러 여부를 체크한다 (기본값 :검정)
3. li객체를 ul태그의 자손으로 붙여버린다.

우려했던 상황 : 좁은 div채팅창에 li가 계속붙어버리면 ul태그에서 오버플로우가 발생한다. css옵션으로 overflow = auto가 기본으로 적용되어 있어 스크롤이 자동으로 생성되지만 태그포커싱과 초점포커싱이 되질 않았다. 이부분의 구현이 개인적으로 스트레스도 가장 많이받았다.

해결 : 자동포커싱 함수를 정의하여 li태그가 생성될 때 마다 가장 마지막 li태그를 가져와 탭인덱스를 0으로 주고 포커싱을 하였고, ul.scrollTo(x좌표, y좌표) 함수를 사용해 스크롤을 내렸다. 그 후 리턴되어 입력단으로 자동 포커싱 시켰다.

```
// li태그 생성해서 채팅창에 갖다붙이는 함수
const createLi = function(text, red=false, blue=false, green=false){
  const li = document.createElement("li");
  if(red) li.style.color="red";   // 빨간글씨 출력용
  if(blue) li.style.color="blue"; // 파란글씨 출력용
  if(green) li.style.color="green"; // 초록글씨 출력용
  li.textContent = text;
  ul.appendChild(li);

  autoScrollAndFocus();
  input.focus();
}

// li태그들이 쌓여 오버플로우가 발생됐을 때 자동으로 시점이 포커싱되는 함수
const autoScrollAndFocus = function() {
  const lastLi = ul.querySelector("li:last-child");
  lastLi.tabIndex = 0;
  lastLi.focus();
  ul.scrollTo(0, ul.scrollHeight);
}
```

주요 함수들(2)

// 시스템 숫자 생성함수와 채팅초기화 함수

시스템 숫자 생성함수

1. 배을의 길이가 3이 아닐때까지 반복하여 1~9값을 받는다.
2. 중복을 없애기위해 조건식을 사용하였는데 흡칫할 수 있는 문장이다. 자바였으면 인덱스범위를 벗어나는 예외가 발생하는데 js는 이와는 다르게 undifined를 반환하여 조건검사가 가능하였다. 조건이 참이되면 배열에 값을 넣고 리턴한다.

- 생각하지도 못한 상황 발생 : 처음 시스템이 랜덤값을 배열로 받았는데 load를 하지 않으면 이 함수를 계속 호출하여도 값이 유지 되었다.
- 해결 : 첫줄에 배열의 길이를 0으로 해두면 초기화 된다는 사실을 구글링을 하여 해결하였다.

채팅 초기화 함수

- 구현이유 : load하지않고 게임 재시작을위해 채팅창을 깔끔하게 한다.
- 간단설명 : querySelectorAll()함수로 li태그를 모두 가져와 배열에 담은 후 반복문을 사용하여 전부 remove()하였다.

```
// 시스템 입력값받기
function systemNum (sysNum){
  sysNum.length=0;
  while(sysNum.length != 3){
    let n = Math.ceil(Math.random()*9);
    if(n!==sysNum[0] && n!==sysNum[1])
      sysNum.push(n);
  }
  return sysNum;
};

// ul태그에 달려있는 li태그를 전부 지우는 함수
const removeLi = function(){
  const lis = document.querySelectorAll("li")
  for(let i = 0; i < lis.length; i++)
    lis[i].remove();
}
```

주요 함수들(3)

사용자 입력값 체크 함수

중복체크 함수

- 구현이유 : 사용자도 정해진 룰에 따라 입력을 해야 하기 때문
- 간단원리 : 3자리 숫자이기 때문에 약간 무식한 방법이지만 split()함수를 사용해 한글자씩 잘라내어 그 중 2개씩 OR 연산자로 참/거짓을 리턴하였다.

0체크 함수

- 구현이유 : 위와 같다.
- 간단원리 : 위와같이 한글자씩 잘라내어 0이 맞는지 체크

궁금증

Q. 비교연산이 왜 ==가 아니고 ===인지?

A. 자바스크립트에서 == 연산은 단순 값만 비교한다 숫자 3과 문자열 "3"을 비교하면 true가 나온다.

하지만 === 연산은 값과 타입을 같이 비교해준다.

```
// 사용자가 입력한 숫자 중복체크하는 함수
function isDuplicate(n) {
  const digits = String(n).split('');
  return digits[0] === digits[1] || digits[0] === digits[2]
         || digits[1] === digits[2];
}

// 사용자가 입력한 숫자중에 0이 들어가 있는지 확인하는 함수
function isDigitZero(n){
  const digits = String(n).split('');
  return digits[0] === "0" || digits[1] === "0"
         || digits[2] === "0";
}
```

주요함수들(4)

숫자게임의 카운트다운 함수

카운트다운 함수

이 함수는 아래의 기능을 가지고 있다.

1. 1초의 주기로 카운트가 다운이 된다.
2. 카운트가 30초단위로 운영자가 메세지를 띄워준다.
3. 카운트가 10초가 됐을 때 숫자표시계가 빨간색이 되어서 임박함을 알려준다.
4. 카운트가 0이 되면 시스템의 승리이므로 게임이종료된다.

추가 : 카운트가 0이 되었을 때

- 채팅창에 운영자와 시스템이 메세지를 입력한다.
- 입력창을 초기화하고 버튼과 입력창을 비활성화 시킨다.
- 입력창에는 placeholder 옵션으로 게임종료 메세지를 띄워준다.

```
// 숫자게임 타이머함수
const startTimer = function(){
  timer = setInterval(()=>{
    if(cnt==10){
      second.style.color = "red";
      second.style.fontSize = "40px";
      second.style.bottom = "-5px";
      second.textContent = --cnt;
      createLi("[운영자] : " + cnt+"초 남았습니다.",false, false, true);
      sysMessage.textContent = "[SYSTEM] : 10초만 지나면 저의 승리";
    }else if(cnt==0){
      clearInterval(timer);
      createLi("[운영자] : 시간초과. SYSTEM 승리",false,false,true);
      createLi("[SYSTEM] : 제가 이겼습니다. 개꿀~ 닙은 " + sysNum.toString(),false,true);
      input.value = "";
      sysMessage.textContent = "[SYSTEM] : 제가 이겼습니다. 개꿀~";
      input.disabled = true;
      input.placeholder = "게임종료";
      chatButton.disabled = true;
    }else if (cnt != 60 && cnt%30 == 0){
      createLi("[운영자] : " + cnt+"초 남았습니다.",false, false, true);
      second.textContent = --cnt;
      sysMessage.textContent = "[SYSTEM] : 서두르시길...킥킥";
    }else{
      second.textContent = --cnt;
    }
  },1000);
}
```

주요함수들(5) - 1

사용자의 입력값 체크 함수 - 1

입력값 체크 함수 - 1

이 함수는 로직은 아래와 같다.

만약 입력값이 123이상 987이하 숫자일 때

if : 중복검사 후 중복이 맞으면 메세지 출력

else if : 0이 들어가 있으면 메세지 출력

else : 위의 검사들을 통과 했으면 숫자야구 알고리즘을 실행하여 strike와 ball의 카운트를 체크한다.

```
const chatAnswer = function() {
  const text = input.value;
  createLi("[USER] : "+ text);
  const num = parseInt(text);

  if((123<=num && num<=987)){
    if(isDuplicate(num)){
      createLi("[운영자] : USER님 중복된 숫자가 있습니다.",true);
      sysMessage.textContent = "[SYSTEM] : USER님이 입력한 숫자에는 중복이
      있어요 제대로 입력해주세요.";
    }else if(isDigitZero(num)){
      createLi("[운영자] : USER님 숫자 0이 존재합니다.",true);
      sysMessage.textContent = "[SYSTEM] : USER님이 입력한 숫자에는 0이 있어요
      제대로 입력해주세요.";
    }else{
      const myNum = String(num).split('');

      let ball=0, strike=0;
      for(let i=0; i<3; i++) {
        for(let j=0; j<3; j++) {
          if(sysNum[i]==myNum[j]) {
            if(i==j) strike++;
            else ball++;
          }
        }
      }
```

주요함수들(5) - 2

사용자의 입력값 체크 함수 - 2

입력값 체크 함수 - 2

이 함수는 로직은 아래와 같다.

만약 strike 카운트가 3이면

1. 타이머를 종료시키고 운영자와 시스템이 메세지를 출력한다.
2. 게임타이머가 끝났을 때와 동일하게 입력단들을 비활성화 시킨다.

그렇지 않다면

1. 시스템이 입력된 값의 strike와 ball 카운트값을 메세지로 띄워준다.

3자리 숫자 이외에는 모든값을 오류값으로 보고 재입력하라고 한다.

채팅이 완료되면 입력단은 공백으로 리셋시킨다.

```
    if(strike === 3){
      createLi("[운영자] : 정답. USER 승리 (기록 : " + (90-cnt) + "초)",false,
      false,true);
      clearInterval(timer);
      createLi("[SYSTEM] : 제가 졌습니다. ㅠㅠ",false,true);
      sysMessage.textContent = "[SYSTEM] : 제가 졌습니다. ㅠㅠ";
      input.value = "";
      input.disabled = true;
      input.placeholder = "게임종료";
      chatButton.disabled = true;
      return;
    }else{
      createLi("[SYSTEM] : " + strike + "스트라이크 " + ball + "볼",false,
      true);
      sysMessage.textContent = "[SYSTEM] : " + strike + "스트라이크 " + ball +
      "볼";
    }
  }
  }else{
    createLi("[운영자] : 숫자 3개만 입력하세요.",true);
    sysMessage.textContent = "USER님 정확한 값을 입력해주세요."
  }
  input.value = "";
}
```

숫자야구 이벤트처리

- DOMContentLoaded
- Click
- Keyup

04

이벤트처리(1) - DOMContentLoaded

DOMContentLoaded

문서 로드 이벤트

- 간단설명 : 웹페이지가 로드 되었을 때 발생하는이벤트
  (HTML, CSS, JavaScript의 구문이 모두 분석 후 발생이벤트)
- 특정 태그들을 querySelector로 가져와서 상수값으로 선언 하였다.
- 게임시작버튼을 누르지 않았기에, 입력단은 비활성화가 된다.

```
document.addEventListener("DOMContentLoaded",()=>{
  const startButton = document.querySelector("#start");
  const input = document.querySelector("#hh");
  const chatButton = document.querySelector("#gg");
  const ul = document.querySelector("#view");
  const sysMessage = document.querySelector("#sys_text");
  const sysNum = [];
  let timer;
  let cnt;

  sysMessage.textContent = "[SYSTEM] : 안녕하세요 SYSTEM입니다.";
  input.disabled = true;
  chatButton.disabled = true;
```

이벤트처리(2) - Click

Click

클릭 이벤트

- 게임시작 버튼을 클릭후 발생하는 이벤트처리 함수이다.
- 3초의 딜레이를 주어서 시스템이 숫자를 생각할 시간을 준다. (생각하는 척)
- 시스템이 생각하는 시간에 카운트값을 할당하고 스타일을 적용한다.
- 만약 이전의 타이머가 clear되지 않았다면 다시한번 클리어해준다.
- 3초의 시간이 흐르면 입력단을 활성화 시키고 메세지를 채팅창에 띄워준 후에 startTimer() 함수를 실행시켜 게임을 시작한다.

```
startButton.addEventListener("click",()=>{
  input.placeholder = "";
  removeLi();

  createLi("3초 뒤 게임시작...");
  createLi("[SYSTEM] : 숫자를 생각하고 있을게요.",false,true);
  sysMessage.textContent = "[SYSTEM] : 숫자를 생각하고 있을게요";

  cnt = 90;
  const second = document.querySelector("#second");
  second.style.color = "black";
  second.style.fontSize = "20px";
  second.style.bottom = "0";
  second.textContent = cnt;

  clearInterval(timer);

  let readyCnt = 3;
  const ready = setInterval(()=>{
    createLi(readyCnt-- + "...");
    if(readyCnt==0) {
      clearInterval(ready);
      createLi("게임을 시작합니다. (제한시간 :90초)");
      systemNum(sysNum);
      input.disabled = false;
      chatButton.disabled = false;
      createLi("[SYSTEM] : 숫자 3개 생각 완료 ㅎㅎ 답을 맞춰보시죠",false,true);
      sysMessage.textContent = "[SYSTEM] : 숫자 3개 생각 완료 ㅎㅎ 답을 맞춰보시죠";
      startTimer();
    }
  },1000);
});
```

이벤트처리(3) - Keyup

Keyup

키 업 이벤트

- 간단설명 : 입력단에 값을 입력하고 엔터키를 누르거나 보내기 버튼을 누르면 이벤트가 발생하고 charAnswer()함수가 실행된다.

- 주의사항 : 공백이 아니면서 엔터키만 인식을 받아 채팅창으로 입력된다.

```
// input태그에 값을 입력하고 보내기 버튼을 눌렀을경우 이벤트처리 함수
chatButton.addEventListener("click",()=>{
  if(input.value.trim() !== "") chatAnswer();
});

// input태그에 값을 입력하고 엔터키를 눌렀을경우 이벤트처리 함수
input.addEventListener("keyup",(event)=>{
  if(event.code === "Enter" && input.value.trim() !== "") chatAnswer();
});
```

본인평가 및 소감

05

나에게 주는 점수 : ★★★☆☆

- html : div태그를 너무 중구난방으로 사용하였음
- css : 알고사용한 속성도 있는반면 얻어걸렸거나 굳이 안써도되는 속성이 사용된 것 같음
- js : 그나마 알고리즘과 함수의 개념을 어느정도 익혀놨기에 괜찮았음

소감

코딩테스트 알고리즘만 주구장창 풀다가 새로운 개발의 벽에 마주한 느낌이였다. 대학생시절 c언어로 별찍기 했을 때와 자바 프로젝트 이후로 처음 밤을 새서 코딩하였다. 힘들고 스트레스 많이 받았지만 재밌었다. JSP를 배우고나서 1차 프로젝트를 하게 될텐데 많이 두렵다.

감사합니다

즐겜하세요