인터페이스 구현

[2001020212_19v5]

1. 인터페이스 설계서 확인하기
2. 인터페이스 기능 구현하기
3. 인터페이스 구현 검증하기

송윤정

과정명: [디지털컨버전스] 뷰(Vue) 활용 프론트엔드 개발자 양성과정 22-1

교과목명: 애플리케이션 프로그래밍 및 구현

주제

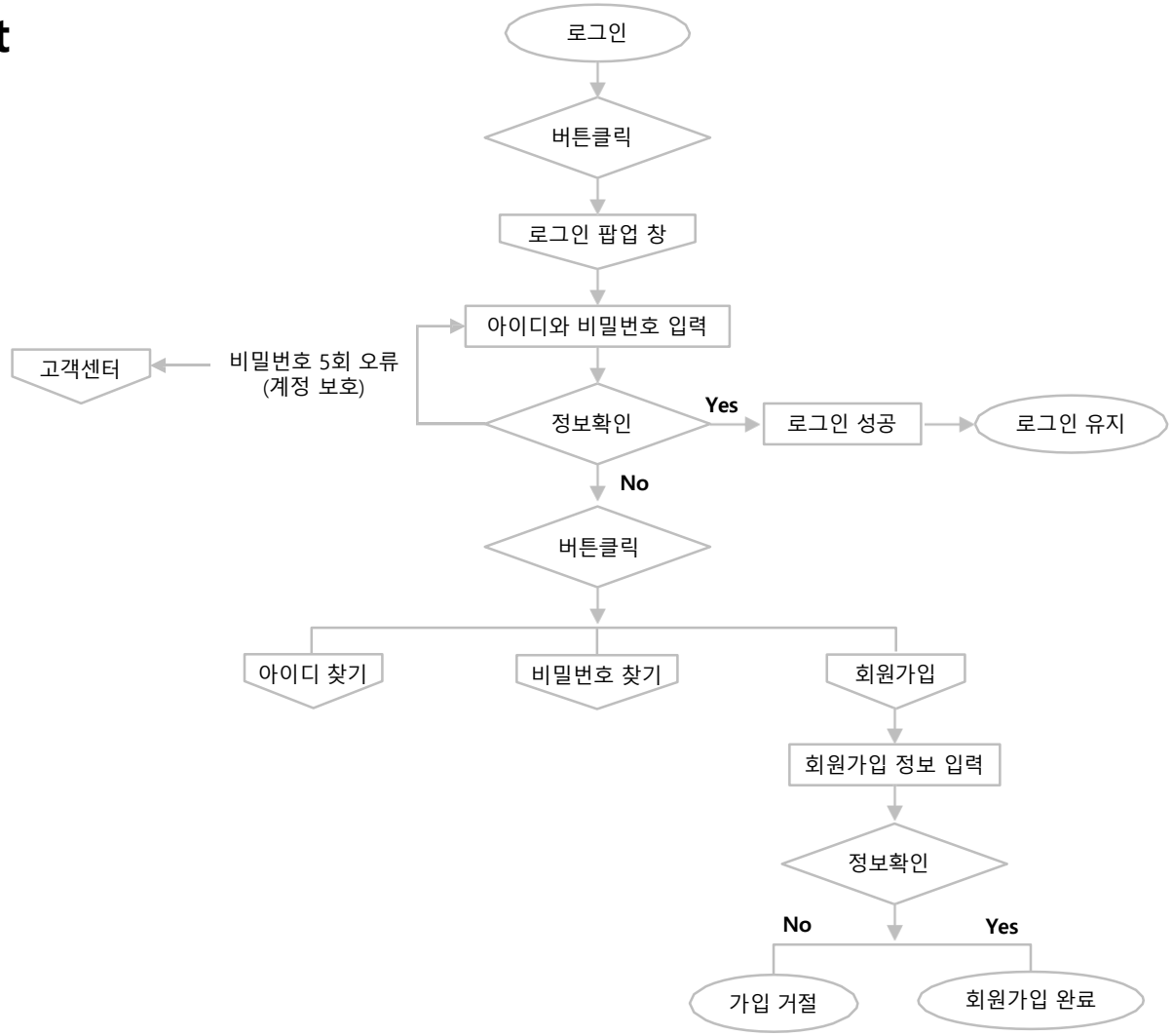
1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오.
2. 구현된 내부 모듈 상에서 단순 로그인의 성공, 실패가 아닌 사용성의 편의를 주기 위한 방안을 제시하고 문제 해결에 필요한 내용을 검증하여 서술하시오.
3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오.

평가기준 기본사항

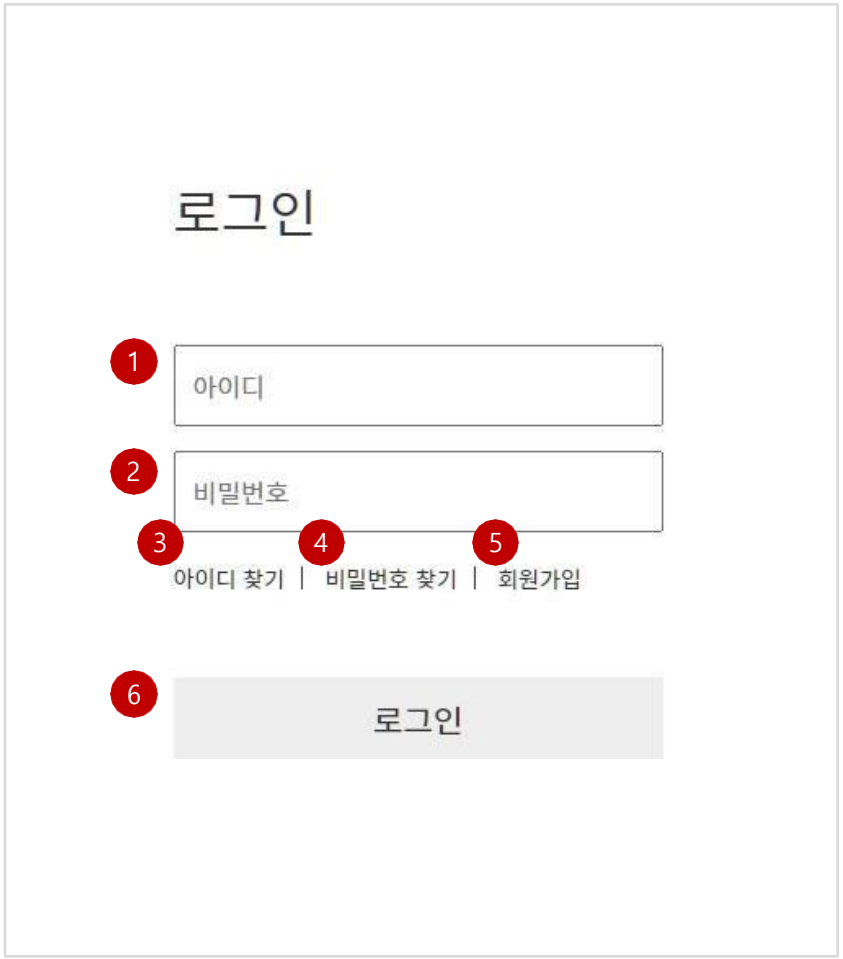
1. 로그인 인터페이스에 필요한 레이아웃 화면 설계서 및 실행에 필요한 흐름도(flowchart), 회원가입 양식을 제작 할 수 있다.
2. 로그인시 아이디, 패스워드 항목을 체크하는 코드를 작성하고 사용성을 기준으로 문제해결에 필요한 내용을 서술 할 수 있다.
3. 회원가입은 정식 회원 가입에 필요한 항목을 자신의 웹사이트 성격에 맞도록 도출하고 구성 내용에 대한 맥락을 서술 할 수 있다.
4. 회원가입 시 주어진 배열 데이터를 활용하여 아이디 사용 여부를 판단할 수 있는 코드를 작성하고 문제해결 내용을 서술 할 수 있다.
   - idData = ['greencomputer', 'redcomputer', 'bluecomputer', 'yellowcomputer', 'blackcomputer']
5. 제시된 조건 외 인터페이스 이용 시 사용성을 높일 수 있는 항목 추가 구성하여 제공할 수 있다.

1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오. (1)

Flowchart

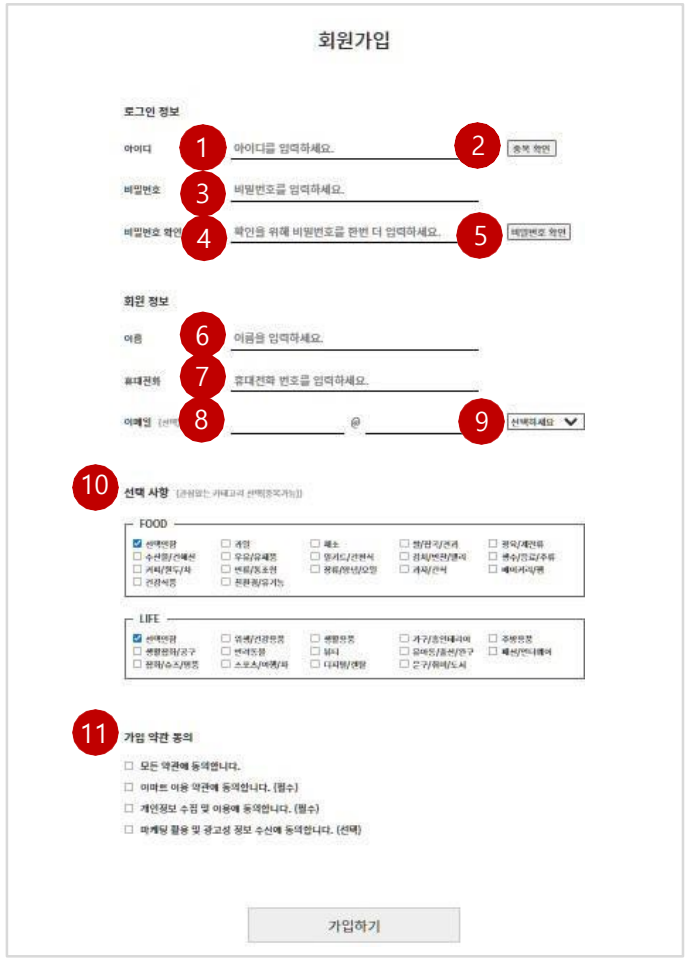


1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오. (2)



로그인 화면 설계서		
DESCRIPTION		
1	기능	아이디 정보 입력 필드
2	기능	비밀번호 정보 입력 필드
3	기능	아이디 찾기 버튼
4	기능	비밀번호 찾기 버튼
5	기능	회원가입 버튼
6	기능	로그인 버튼 - 아이디 입력 필드, 비밀번호 입력 필드 전체 입력 시 활성화 되는 버튼 case1 : 아이디, 비밀번호 일치하는 회원정보 있음 → 로그인 완료 case2 : 아이디 정보 있으나 비밀번호 틀림 → '비밀번호가 틀렸습니다.' → 재입력 (횟수 제한 안내) → 5회 이상 틀릴 시 계정 보호 case3 : 일치하는 회원정보 없음 → 로그인 실패 → '정보가 없습니다.' 안내 알림

1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오. (3)

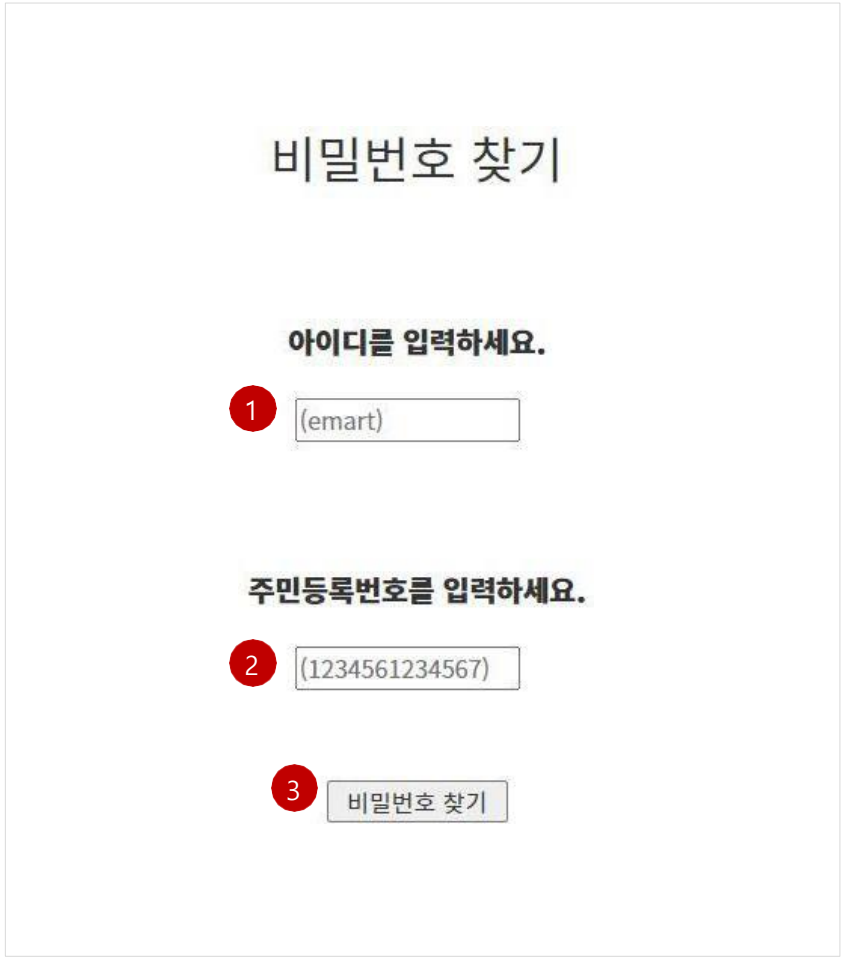


회원가입 화면 설계서					
DESCRIPTION					
1	기능	아이디 정보 입력 필드	6	기능	이름 정보 입력 필드
2	기능	아이디 중복 확인 버튼 case1 : 일치하는 기존 회원정보 있음(동일 아이디 존재) → '사용할 수 없습니다.' 안내 알림 case2 : 일치하는 기존 회원정보 없음 → '사용할 수 있습니다.' 안내 알림	7	기능	휴대전화 정보 입력 필드
			8	기능	이메일 정보 입력 필드
3	기능	비밀번호 정보 입력 필드	9	기능	이메일 주소 선택 옵션 case1: naver.com case2: hanmail.net case3: nate.com case4: yahoo.co.kr case5: korea.com case6: 직접입력
4	기능	비밀번호 재확인 정보 입력 필드			
5	기능	비밀번호 일치 확인 버튼 case1 : 비밀번호 일치 → '사용할 수 있습니다. 비밀번호가 일치합니다.' 안내 알림 case2 : 비밀번호 일치 → '사용할 수 없습니다. 비밀번호가 다릅니다.' 안내 알림	10	기능	선택 사항 옵션 (관심있는 카테고리 선택) - FOOD, LIFE (중복선택 가능)
			11	기능	약관 동의 옵션

1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오. (4)

아이디 찾기 화면 설계서		
DESCRIPTION		
1	기능	주민등록번호 정보 입력 필드
2	기능	아이디 찾기 버튼 case1 : 일치하는 주민등록번호 있음 → '아이디는 [?] 입니다.' 안내 알림 case2 : 일치하는 주민등록번호 없음 → '존재하지 않는 회원입니다.' 안내 알림

1. 로그인을 위한 인터페이스 화면 레이아웃을 설계하고 회원가입 내용과 함께 구현하시오. (5)



비밀번호 찾기 화면 설계서		
DESCRIPTION		
1	기능	아이디 정보 입력 필드
2	기능	주민등록번호 정보 입력 필드
3	기능	비밀번호 찾기 버튼 case1 : 일치하는 주민등록번호 있음 : 일치하는 아이디 있음 → '비밀번호는 [?] 입니다.' 안내 알림 case2 : 일치하는 주민등록번호 있음 : 일치하는 아이디 없음 → '아이디가 틀렸습니다.' 안내 알림 case3 : 일치하는 주민등록번호 없음 → '존재하지 않는 회원입니다.' 안내 알림

2. 구현된 내부 모듈 상에서 단순 로그인의 성공, 실패가 아닌 사용성의 편의를 주기 위한 방안을 제시하고 문제 해결에 필요한 내용을 검증하여 서술하시오. (1)

```
$(function() {
  let id = 'emart';
  let pw = '1234';

  let userId = '';
  let userPw = '';
  let count = 0;


  $('.btn').click(function() {
    for(let i=1; ; i++) {
      userId = $('#user_id').val();
      userPw = $('#user_pw').val();

      count++;
      let chance = 5 - count;

      if(id == userId && pw == userPw) {
        alert(userId + '님 환영합니다.');
        window.close();
        break;
      } else if(id == userId) {
        if(count === 5) {
          alert('계정보호 되었습니다. 고객센터에 문의하세요.');
          window.close();
        } else {
          alert('비밀번호가 틀렸습니다. ' + chance + '회 남음. (5회 오류 시 계정보호 처리됨.)');
        }
        break;
      } else {
        alert('정보가 없습니다.');
        $('#user_id').val('').focus();
        $('#user_pw').val('');
        break;
      }
    }
  })
})
```

* **아이디 / 비밀번호 오류 구분**
단순 로그인의 성공과 실패로 구현할 시,
아이디가 틀렸는지 비밀번호가 틀렸는지 알 수 없는 사용자는 사용성의 불편함을 느낀다.
아이디가 틀렸다면 정보가 없음(아이디 틀렸음)을 안내하고,
비밀번호가 틀렸다면 비밀번호가 틀렸음을 안내하여 사용성의 편의를 줄 수 있다.

→ 아이디와 비밀번호 정보가 둘 다 맞으면 'userId + 님 환영합니다.' 안내 알림.

→ 아이디 정보는 존재하지만 비밀번호 정보와 맞지 않을 시,
비밀번호가 틀렸다는 것을 알려주는 '비밀번호가 틀렸습니다.' 안내 알림.
그리고 비밀번호 입력 5회 횟수 제한을 두어, 5번 이상 입력 실패 시 계정이 보호됨.

→ 아이디가 틀렸을 시 '정보가 없습니다.' 안내 알림.

2. 구현된 내부 모듈 상에서 단순 로그인의 성공, 실패가 아닌 사용성의 편의를 주기 위한 방안을 제시하고 문제 해결에 필요한 내용을 검증하여 서술하시오. (2)

```
$(function() {
  const id = '1234561234567emart';
  let userNumber = '';
  let number = '';
  const findBtn = ('.find_btn');

  $(findBtn).click(function() {
    userNumber = $('.user_number').val();
    number = id.slice(0,13);

    if(userNumber == number) {
      alert('아이디는 [' + id.slice(13,19) +'] 입니다.')
    } else {
      alert('존재하지 않는 회원입니다.')
    }
  })
})
```

* **아이디 찾기**
로그인 시 아이디를 잊어버렸다면 아이디 찾기 기능이 필요하다.
사용성 편의를 주기위해 주민등록번호로 아이디 찾기 기능을 구현하였다.

아이디를 찾는 방법은 주민등록번호를 입력한 뒤,
아이디 찾기 버튼을 누르면 된다.

구현 방법은 아이디를 '주민등록번호 + 아이디'로 저장해두고
주민등록 번호가 맞으면 slice를 활용해 아이디 값만 추출하여 보여준다.
등록된 주민등록번호가 없다면 '존재하지 않는 회원입니다.'고 알려준다.

2. 구현된 내부 모듈 상에서 단순 로그인의 성공, 실패가 아닌 사용성의 편의를 주기 위한 방안을 제시하고 문제 해결에 필요한 내용을 검증하여 서술하시오. (3)

```
$(function() {
  const id = 'emart';
  const pw = '12345612345671234';
  let userId = '';
  let userNumber = '';
  let number = '';
  const findBtn = ('.find_btn');

  $(findBtn).click(function() {
    userId = $('.user_id').val();
    userNumber = $('.user_number').val();
    number = pw.slice(0,13);

    if(userNumber == number) {
      if(userId == id) {
        alert('비밀번호는 [' + pw.slice(13,19) +'] 입니다.')
      } else {
        alert('아이디가 틀렸습니다.')
      }
    } else {
      alert('존재하지 않는 회원입니다.')
    }

    $('.user_id').val('').focus();
    $('.user_number').val('');
  })
})
```

* **비밀번호 찾기**
로그인 시 비밀번호를 잊어버렸다면 비밀번호 찾기 기능이 필요하다.
사용성 편의를 주기위해 아이디와 주민등록번호로 비밀번호 찾기 기능을 구현하였다.

비밀번호를 찾는 방법은 아이디와 주민등록번호를 입력한 뒤,
비밀번호 찾기 버튼을 누르면 된다.

구현 방법은 비밀번호를 '주민등록번호 + 비밀번호'로 저장해두고,
중첩if를 사용하여 주민등록번호가 존재한다면
사용자가 입력한 아이디 값이 맞는지 확인한다.
주민등록 번호와 아이디가 맞으면 slice를 활용해 비밀번호 값만 추출하여 보여준다.
주민등록번호는 존재하지만 아이디가 틀리면 '아이디가 틀렸습니다.'라고 알려준다.
존재하지 않는 주민등록번호라면 '존재하지 않는 회원입니다.'라고 알려준다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (1)

```
// id pw change_event
let userIdCk = document.getElementById('user_id');
let userPwCk = document.getElementById('user_pw');
let userPwCkRe = document.getElementById('user_pw_re');

userIdCk.addEventListener('change', checkId);
userPwCk.addEventListener('change', checkPw);
userPwCkRe.addEventListener('change', checkPwRe);

function checkId() {
  if(userIdCk.value.length < 6 || userIdCk.value.length > 15) {
    document.querySelector('.id_message').textContent = '6~15자로 입력하세요.';
  } else {
    document.querySelector('.id_message').textContent = '사용가능한 아이디입니다.';
  }
}

function checkPw() {
  if(userPwCk.value.length < 4) {
    document.querySelector('.pw_message').textContent = '4자 이상 입력하세요.';
  } else {
    document.querySelector('.pw_message').textContent = '사용가능한 비밀번호입니다.';
  }
}

function checkPwRe() {
  if(userPwCk.value != 0 && userPwCkRe.value != 0) {
    if(userPwCk.value == userPwCkRe.value) {
      document.querySelector('.pw_re_message').textContent = '비밀번호가 일치합니다.';
    } else {
      document.querySelector('.pw_re_message').textContent = '비밀번호가 다릅니다.';
    }
  } else {
    document.querySelector('.pw_re_message').textContent = '';
  }
}

setInterval(checkId, 1);
setInterval(checkPw, 1);
setInterval(checkPwRe, 1);
```

* **아이디&비밀번호 글자 수 제한**
회원가입 제작 시 너무 짧거나 혹은 너무 긴 아이디&비밀번호는 사용할 수 없도록 제한이 필요하다.

글자 수를 제한하고 setInterval을 주어 실시간으로 값을 체크한다.
조건에 맞다면 '사용가능한 아이디입니다.', '사용가능한 비밀번호 입니다.' 안내 멘트를 출력한다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (2)

```
// #id_check_btn click_event (아이디 중복확인 버튼)
let userId = '';
let idData = '';
let userPw = '';
let userPwRe = '';

$('#id_chk_btn').click(function() {
  userId = $('#user_id').val();
  idData = new Array('greencomputer', 'redcomputer', 'bluecomputer', 'yellowcomputer', 'blackcomputer');

  for(let i in idData) {
    if(userId == idData[i]) {
      alert('사용할 수 없는 아이디 입니다.');
      $('#user_id').val('').focus();
      break;
    } else if(i == 4) {
      alert('사용가능한 아이디입니다.');
      $('#user_pw').focus();
    }
  }
})
```

* **아이디 중복 확인**
회원가입 제작 시 정식 가입을 할 때,
중복된 아이디가 이미 존재한다면
그 아이디는 사용할 수 없도록 제한해야 한다.

아이디 정보 입력 필드에
아이디를 입력 후 중복 확인 버튼 클릭 시,
배열 데이터인 idData에 동일한 아이디가
이미 존재한다면
'사용할 수 없는 아이디 입니다.' 안내 알림.

구현 방법은
배열 데이터에 존재하는 아이디값을 저장한다.
배열 데이터를 활용하여
아이디 사용 가능 여부를 판단한다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (3)

```
// id pw change_event
let userIdCk = document.getElementById('user_id');
let userPwCk = document.getElementById('user_pw');
let userPwCkRe = document.getElementById('user_pw_re');

userIdCk.addEventListener('change', checkId);
userPwCk.addEventListener('change', checkPw);
userPwCkRe.addEventListener('change', checkPwRe);

function checkId() {
  if(userIdCk.value.length < 6 || userIdCk.value.length > 15) {
    document.querySelector('.id_message').textContent = '6~15자로 입력하세요.';
  } else {
    document.querySelector('.id_message').textContent = '사용가능한 아이디입니다.';
  }
}

function checkPw() {
  if(userPwCk.value.length < 4) {
    document.querySelector('.pw_message').textContent = '4자 이상 입력하세요.';
  } else {
    document.querySelector('.pw_message').textContent = '사용가능한 비밀번호입니다.';
  }
}

function checkPwRe() {
  if(userPwCk.value != 0 && userPwCkRe.value != 0) {
    if(userPwCk.value == userPwCkRe.value) {
      document.querySelector('.pw_re_message').textContent = '비밀번호가 일치합니다.';
    } else {
      document.querySelector('.pw_re_message').textContent = '비밀번호가 다릅니다.';
    }
  } else {
    document.querySelector('.pw_re_message').textContent = '';
  }
}

setInterval(checkId, 1);
setInterval(checkPw, 1);
setInterval(checkPwRe, 1);
```

* **비밀번호 재확인**
회원가입 제작 시 비밀번호 재확인을 하지 않는다면 잘못 입력된 비밀번호로 설정될 수도 있다.
비밀번호 재확인으로 오류를 줄여 사용성의 편의를 줄 수 있다.

'비밀번호'와 '비밀번호 확인'의 입력 필드에 값을 입력하면 setInterval이 실시간으로 비밀번호가 일치하는지 확인한다.

구현방법은
처음 입력한 비밀번호(userPWCk)와 재확인을 위해 다시 입력한 비밀번호(userPwCkRe)가 동일하다면 '비밀번호가 일치합니다.' 안내 멘트 출력.
다르다면 '비밀번호가 다릅니다.' 안내 멘트 출력.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (4)

```
// tel change_event
const telFirst = document.getElementById('tel_first');
const telMid = document.getElementById('tel_mid');
const telLast = document.getElementById('tel_last');

telFirst.addEventListener('keyup', moveTel1);
telMid.addEventListener('keyup', moveTel2);

function moveTel1() {
  if(telFirst.value.length == 3) {
    telMid.focus();
  }
}

function moveTel2() {
  if(telMid.value.length == 4) {
    telLast.focus();
  }
}
```

* **전화번호 입력 시 자동으로 입력 칸 이동**
  전화번호 입력 시 자동으로 입력 칸을 이동해줌으로써
  사용자에게 편의를 제공 할 수 있다.

  구현방법은
  첫번째 칸에 글자 수가 3이 된다면 .focus()로 두번째 칸으로 이동한다.
  두번째 칸에 글자 수가 4가 된다면 .focus()로 마지막 칸으로 이동한다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (5)

```
// mail change_event
$('#mail_select').change(function() {
  $('#mail_site').val($('#mail_select').val());
})
```

* **이메일 자동선택**
  이메일을 입력할 때 선택항목을 제시하여
  사용성의 편의를 줄 수 있다.

  구현방법은
  .change로 값이 변하는 것을 인지하고
  선택한 값을 이메일 입력 칸의 값으로 넣어준다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (6)

```
// like_food_info click_event (관심 카테고리 FOOD)
const likeFoodNo = document.getElementById('like_food_no');

likeFoodNo.addEventListener('click', function() {
  for(let i=1; i<=16; i++) {
    let likeFood = document.getElementById('like_food' + i);

    if(likeFoodNo.checked) {
      likeFood.checked = false;
    } else {
      likeFoodNo.checked = true;
    }
  }
})

for(let i=1; i<=16; i++) {
  let likeFood = document.getElementById('like_food' + i);

  likeFood.addEventListener('click', function() {
    if(likeFood.checked) {
      likeFoodNo.checked = false;
    } else if(document.getElementById('like_food1').checked == false &&
              document.getElementById('like_food2').checked == false &&
              document.getElementById('like_food3').checked == false &&
              document.getElementById('like_food4').checked == false &&
              document.getElementById('like_food5').checked == false &&
              document.getElementById('like_food6').checked == false &&
              document.getElementById('like_food7').checked == false &&
              document.getElementById('like_food8').checked == false &&
              document.getElementById('like_food9').checked == false &&
              document.getElementById('like_food10').checked == false &&
              document.getElementById('like_food11').checked == false &&
              document.getElementById('like_food12').checked == false &&
              document.getElementById('like_food13').checked == false &&
              document.getElementById('like_food14').checked == false &&
              document.getElementById('like_food15').checked == false &&
              document.getElementById('like_food16').checked == false) {
      likeFoodNo.checked = true;
    }
  })
}
```

* **선택사항 체크박스**
마트 웹사이트 특성상 많은 카테고리가 존재한다.
고객 사용성의 편의를 제공하기 위해 정식 회원가입 시,
관심있는 카테고리 데이터를 저장해두고 웹사이트 이용 시 맞춤 추천 서비스를 제공한다.
선택을 원하지 않는 사용자를 위해서 필수 사항이 아닌,
선택 사항으로 만들고 '선택안함' 옵션을 두어 checked 설정하였다.

선택안함을 체크한다면 모든 선택 사항이 제거되고,
하나라도 선택 한다면, 선택안함 항목의 체크가 제거된다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (7)

```
// #all_chk #chk click_event (가입 약관 동의)
const allChk = document.getElementById('all_chk');
const chk1 = document.getElementById('chk1');
const chk2 = document.getElementById('chk2');
const chk3 = document.getElementById('chk3');

allChk.addEventListener('click', function() {
  if(allChk.checked) {
    chk1.checked = true;
    chk2.checked = true;
    chk3.checked = true;
  } else {
    chk1.checked = false;
    chk2.checked = false;
    chk3.checked = false;
  }
})
chk1.addEventListener('click', function() {
  if(chk1.checked == false) {
    allChk.checked = false;
  } else if(chk2.checked && chk3.checked) {
    allChk.checked = true;
  }
})
chk2.addEventListener('click', function() {
  if(chk2.checked == false) {
    allChk.checked = false;
  } else if(chk1.checked && chk3.checked) {
    allChk.checked = true;
  }
})
chk3.addEventListener('click', function() {
  if(chk3.checked == false) {
    allChk.checked = false;
  } else if(chk1.checked && chk2.checked) {
    allChk.checked = true;
  }
})
```

* **가입 약관 동의**
  정식 회원가입 시 가입 약관 동의는 필수사항이다.

  '모든 약관에 동의합니다.'를 선택한다면
  모든 약관이 선택되고
  하나라도 동의하지 않는다면
  '모든 약관에 동의합니다.'가 선택 해제된다.

3. 회원가입 제작 시 정식 가입이라는 문제를 두고 필요한 내용을 도출하고 문제해결의 맥락을 서술하시오. (8)

```
<div class="login_info info">
  <div class="sub_title">로그인 정보<span style="color: firebrick; font-size: 12px; margin-left: 10px;">(*필수입력)</span></div>

  <div class="id">
    <p>아이디<span style="color: firebrick;">*</span></p>
    <input type="text" id="user_id" placeholder="아이디를 입력하세요." required>
    <button id="id_chk" style="width: 70px;">중복 확인</button>
  </div><!--id-->

  <div class="pw">
    <p>비밀번호<span style="color: firebrick;">*</span></p>
    <input type="password" id="user_pw" placeholder="비밀번호를 입력하세요." required>
  </div><!--pw-->

  <div class="pw_re">
    <p>비밀번호 확인<span style="color: firebrick;">*</span></p>
    <input type="password" id="user_pw_re" placeholder="확인을 위해 비밀번호를 한번 더 입력하세요." required>
    <button id="pw_chk" style="width: 90px">비밀번호 확인</button>
  </div>
</div><!--login_info-->
```

* **필수입력**
정식 회원가입 시 꼭 필요한 필수정보들이 있다.
<input>의 required 속성을 활용하고
이를 사용자가 알 수 있도록 '*'를 표기하였다.