



4차 시험 과제 제출

풀스택 자바 웹개발자 강의
방미소



목차

1. 레이아웃 정의
2. 구현 결과
3. 파트별 구현 결과 1~10

1. 레이아웃 정의 (전체)

■ 헤더

- 타이틀 위치

■ 메인

- 회원가입 폼

■ 푸터

- 기타 링크, 로고 등

NAVER

아이디

비밀번호

비밀번호 재확인

이름

생년월일

년(4자)

월

일

성별

성별

본인 확인 이메일(선택)

휴대전화

대한민국 +82

전화번호 입력

인증번호 받기

인증번호 입력하세요

가입하기

이용약관 | 개인정보처리방침 | 책임의 한계와 법적고지 | 회원정보 고객센터

© NAVER Corp.

1. 레이아웃 정의 (기능)

NAVER

아이디

dsfsfasdfsdfsdfdd

@naver.com

멋진 아이디네요!

비밀번호

..

사용불가

8~16자 영문 대 소문자, 숫자, 특수문자를 사용하세요.

비밀번호 재확인

필수 정보입니다.

이름

1

생년월일

2021

5

24

만 14세 미만의 어린이는 보호자 동의가 필요합니다.

성별

성별

필수 정보입니다.

본인 확인 이메일 (선택)

선택입력

휴대전화인증

아이핀 인증

☒ 아래 약관에 모두 동의합니다.

☒ 개인정보 이용

☒ 고유식별정보 처리

☒ 통신사 이용약관

☒ 인종차 이용약관

☒ 네이버 개인정보 수집

주요 기능

값에 대한 유효성 검사

- > 숫자, 영문자 등 문자 종류에 제한
- > 5~20자 내외 등 문자 개수에 제한
- > 값의 일치여부 확인

세부 디테일

- > 만 14세 미만 사용자의 경우 보호자 동의창이 따로 디스플레이
- > 약관 체크박스

2. 구현 결과

NAVER

아이디

@naver.com

비밀번호

비밀번호 재확인

이름

생년월일

년(4자) 월 일

성별

성별

본인 확인 이메일(선택)

선택입력

휴대전화

대한민국 +82

전화번호 입력

인증번호 받기

인증번호 입력하세요

가입하기

이용약관 | 개인정보처리방침 | 책임의 한계와 법적고지 | 회원정보 고객센터
©NAVER Corp.

NAVER

아이디

@naver.com

비밀번호

비밀번호 재확인

이름

생년월일

2010 4 12

성별

성별

본인 확인 이메일(선택)

선택입력

휴대전화인증

아이핀 인증

☐ 이백 약관에 모두 동의합니다.
☐ 개인정보 이용 ☐ 국문신발정보 처리
☐ 통신사 이용약관 ☐ 인증사 이용약관
☐ 네이버 개인정보 수집

보호자 이름

보호자 생년월일

년(4자) 월 일

보호자 성별/국적

성별

내국인

통신사

SKT

휴대전화

전화번호 입력

인증번호 받기

인증번호 입력하세요

가입하기

이용약관 | 개인정보처리방침 | 책임의 한계와 법적고지 | 회원정보 고객센터
©NAVER Corp.

구현 내용

- 각 영역별 유효성 검사
- 일정 연령대 이하 추가 파트 디스플레이
- 약관 체크박스 외
- HTML 문서 1개
- CSS 문서 2개 (SCSS / resetCSS)
- JS 문서 3개
 - Error Exception
 - Regex Helper
 - form Value Checker

3. 파트별 구현내용 - ① 아이디

아이디

@naver.com

```
<main>
  <div class="inner">
    <div class="container">
      <h2 class="blind-txt">회원가입 폼</h2>
      <form class="signup-form">
        <!-- id, password -->
        <div class="form-group">
          <div class="form-item">
            <label for="user_id"><h3>아이디</h3></label>
            <div class="form-input">
              <input type="text" name="user_id" id="user_id" maxlength="20">
              <span class="mail-addr input-r-side">@naver.com</span>
            </div>
          </div>
        </div>
      </form>
    </div>
  </div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
input ...

일부 자주 사용하는 CSS 기능은 HTML 문서에 클래스로 넣어줌
(ex: blind-txt -> 리더기에 읽히기 위해 제목을 숨겨둔 요소)

3. 파트별 구현내용 - ① 아이디

아이디

@naver.com

```
/**
 * @filename    : Exception
 * @author      : 방미소
 * @description : 에러 반환 기본 클래스
 */

class Exception extends Error {
  constructor(msg = '내용을 확인해주세요.', selector = null) {
    super(msg);
    this._selector = selector;
  }

  get selector() {
    return this._selector;
  }

  set selector(param) {
    this._selector = param;
  }
}
```

Exception.js

error를 원하는 형태로 반환할 기본 클래스 생성

3. 파트별 구현내용 - ① 아이디

아이디

@naver.com

```
/**
 * @filename : RegexHelper.js
 * @author : 방미소
 * @description : 정규표현식 체크
 */

class RegexHelper {

  /**
   * >> 값이 존재하는지 확인
   * @param {string} selector 검사 대상의 css 선택자
   * @param {string} msg 값이 존재하지 않을 경우 표시할 메시지
   */
  value(selector, msg) {
    const content = document.querySelector(selector).value;

    if (content === undefined || content === null || (typeof content === 'string' && content.trim().length === 0)) {
      throw new ErrorException(msg, selector);
    }

    return true;
  }

  /**
   * >> 값이 지정된 글자수를 초과하거나 미달했는지 확인
   * @param {string} selector 검사 대상의 css 선택자
   * @param {int} minlen 최소 글자수
   * @param {int} maxlen 최대 글자수
   * @param {string} msg 글자수가 맞지 않을 경우 표시할 메시지
   */
  valueLen(selector, minlen, maxlen, msg) {
    this.value(selector, msg);

    const content = document.querySelector(selector).value;

    if (content.trim().length < minlen || content.trim().length > maxlen) {
      throw new ErrorException(msg, selector);
    }

    return true;
  }
}
```

```
/**
 * >> 입력값이 정규표현식에 맞는지 확인
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg 오류 시 표시할 메시지
 * @param {object} regexExpr 검사할 정규표현식
 */

field(selector, msg, regexExpr) {
  this.value(selector, msg);

  const content = document.querySelector(selector).value;
  const src = content.trim();

  // 검사 오류 시
  if (!regexExpr.test(src)) {
    throw new ErrorException(msg, selector);
  }

  return true;
}

/**
 * ID : 영문 소문자, 숫자, 특수기호(_)(-)
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg 표시할 메시지
 */
checkId(selector, msg) {
  return this.field(selector, msg, /^[a-z0-9-]*$/);
}
```

RegexHelper.js

함수, 정규표현식에 따라 에러 체크

3. 파트별 구현내용 - ① 아이디

아이디

@naver.com

```
/**
 * @filename    : formValueChecker
 * @author      : 방미소
 * @description  : 폼 유효성 체크 이벤트
 */

/**
 * >> 아이디 유효성 체크
 * 검사 내용 : 값의 유무 / 값의 유효성
 *
 * 값이 적합할 경우 '멋진 아이디네요!' 메시지 출력
 */
document.querySelector('#user_id').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();

  try {
    regexHelper.value('#user_id', '필수 정보입니다.');
```

```
    regexHelper.valueLen('#user_id', 5, 20, '5~20자의 영문 소문자, 숫자와 특수기호(_),(-)만 사용 가능합니다.');
```

```
    regexHelper.checkId('#user_id', '5~20자의 영문 소문자, 숫자와 특수기호(_),(-)만 사용 가능합니다.');
```

```
  } catch (e) {
    alert(e.message);
    return;
  }

  alert('멋진 아이디네요!');
```

```
});
```

formValueChecker.js

blur 이벤트 활용

값의 유무,
값의 길이 및 문자 등 유효성 검사 실행

모든 유효성 검사를 통과하면 축하 메시지 출력

3. 파트별 구현내용 - ① 아이디

색상 변수 지정

```
/* *****  
variable  
***** */  
  
// color  
$main-background-color: #F5F6F7;  
$input-border-color: #DADADA;  
$input-border-color2: #847E7E;  
$input-content-color: #ddd;  
$input-content-color2: #919191;  
$button-border-color: #03B753;  
$button-color: #03C75A;  
  
// font  
$main-font-color: #000;  
$main-font-color2: #fff;  
$font-color1: #b6b3b3;  
$font-color2: #847E7E;
```

기본 페이지 레이아웃

```
/* *****  
Page Layout  
***** */  
  
* {  
  box-sizing: border-box;  
}  
  
body {  
  width: 100%;  
  background-color: $main-background-color;  
  font-family: 'Dotum', 'Helvetica', 'sans-serif';  
  font-size: 14px;  
}  
  
body * {  
  font-family: 'Dotum', 'Helvetica', 'sans-serif'  
}  
  
h3,  
strong {  
  font-weight: bold;  
}  
  
a,  
button,  
input,  
select {  
  cursor: pointer;  
}  
  
a {  
  text-decoration: none;  
  
  &:hover {  
    text-decoration: underline;  
    color: $button-border-color;  
  }  
  
  & *:hover {  
    text-decoration: underline;  
    color: $button-border-color;  
  }  
}
```

아이디

@naver.com

숨김 글자(h태그 등), flexbox 전용 코드

```
.blind-txt {  
  position: absolute;  
  width: 1px;  
  height: 1px;  
  margin: -1px;  
  overflow: hidden;  
  clip: rect(0, 0, 0, 0);  
  clip-path: inset(50%);  
}  
  
.flex-row {  
  display: flex;  
  justify-content: space-between;  
}  
  
.flex-col {  
  display: flex;  
  flex-direction: column;  
}  
  
.hide {  
  display: none;  
}
```

3. 파트별 구현내용 - ① 아이디

```
input,
select{
  padding: 10px 14px;
  border: 1px solid #{$input-border-color};
}

input:focus,
select:focus {
  border: 1px solid #{$button-border-color};
  outline: none;
}

input::placeholder {
  color: #{$font-color1};
}

select {
  background-image: url(../img/arrow.png);
  background-repeat: no-repeat;
  background-position: right;
}

button {
  border: 1px solid #{$button-border-color};
  background-color: #{$button-color};
  color: #{$main-font-color2};
}
```

input 박스, select, 버튼 등
주요 요소 CSS 설정

input 박스와 select의 경우
focus 시 테두리가 녹색으로 변함

아이디

input 박스 안에
메일 어드레스를 넣은 부분

```
.form-item {
  position: relative;

  .input-r-side {
    position: absolute;
  }

  .mail-addr {
    top: 42px;
    right: 15px;
    color: #{$font-color2};
  }
}
```

3. 파트별 구현내용

실제 구동 모양

127.0.0.1:5501 내용:
필수 정보입니다.

아이디

@naver.com

확인

값이 없을 때

127.0.0.1:5501 내용:
5~20자의 영문 소문자, 숫자와 특수기호(_,-)만 사용 가능합니다.

아이디

adstf @naver.com

확인

글자 수 위반

127.0.0.1:5501 내용:
멋진 아이디네요!

아이디

adstf3243 @naver.com

확인

양식에 맞을 때

127.0.0.1:5501 내용:
5~20자의 영문 소문자, 숫자와 특수기호(_,-)만 사용 가능합니다.

아이디

테스트를위한텍스트23123 @naver.com

확인

문자 종류 위반

3. 파트별 구현내용 - ② 비밀번호

비밀번호

```
<div class="form-item">  
  <label for="user_pw"><h3>비밀번호</h3></label>  
  <div class="form-input">  
    <input type="password" name="user_pw" id="user_pw">  
    <span class="pw-check-msg input-r-side"></span>  
    <span class="pw-icon input-r-side"></span>  
  </div>  
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
input ...

이후 유효성 검사 결과에 따라 메시지가 출력될 곳을 span 태그로 작성
(span.pw-check-msg)

3. 파트별 구현내용 - ② 비밀번호

비밀번호

```
/**
 * PW : 영문 대/소문자, 숫자, 특수기호
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg      표시할 메시지
 */
checkPw(selector, msg) {
  return this.field(selector, msg, /^[a-zA-Z0-9#?!@$%^&*~]*$/);
}
```

RegexHelper.js

함수, 정규표현식에 따라 에러 체크

영어 대소문자, 숫자로만 구성할 수 있도록 작성

3. 파트별 구현내용 - ② 비밀번호

비밀번호



```
/**
 * >> 비밀번호 유효성 체크
 * 검사 내용 : 값의 유무 / 값의 유효성
 *
 * 값이 아무것도 입력되지 않았을 때는 회색 자물쇠
 * 값이 입력되었으나 부적합할 때는 빨간 자물쇠 + 오류 메시지
 * 값이 입력되었고 적합할 때는 녹색 자물쇠로 변화
 */
document.querySelector('#user_pw').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();
  const icon = document.querySelector('.pw-icon');
  const msg = document.querySelector('.pw-check-msg');

  try {
    regexHelper.value('#user_pw', '필수 정보입니다. ');
    regexHelper.valueLen('#user_pw', 8, 16, '8~16자 영문 대 소문자, 숫자, 특수문자를 사용하세요. ');
    regexHelper.checkPw('#user_pw', '8~16자 영문 대 소문자, 숫자, 특수문자를 사용하세요. ');
  } catch (e) {
    alert(e.message);

    const current = document.querySelector('#user_pw').value.trim();

    // 아이콘 & 메시지
    if (current !== '') {
      icon.classList.remove('checked');
      icon.classList.add('error');
      msg.innerHTML = '사용불가';
      msg.style.color = 'red';
    }

    return;
  }

  // 아이콘 & 메시지
  icon.classList.remove('error');
  icon.classList.add('checked');
  msg.innerHTML = '안전';
  msg.style.color = '■ #03C75A';
});
```

formValueChecker.js

비밀번호 입력값에 따라 input 박스 우측의 icon, msg에 변화가 발생하도록 작성

msg의 경우 빈 span 태그에 .innerHTML로 특정 조건에서만 출력되도록 함

3. 파트별 구현내용 - ② 비밀번호

비밀번호



```
.pw-check-msg {
  top: 44px;
  right: 43px;
  font-size: 12px;
}

.pw-icon {
  top: 37px;
  right: 15px;
  display: inline-block;
  width: 24px;
  height: 24px;
  background: url(../img/naver-icon.png);
  background-position: 0 0;
  background-size: 125px;

  &.checked {
    background-position: -54px 0;
    background-size: 125px;
  }

  &.error {
    background-position: -27px -27px;
    background-size: 125px;
  }
}
```

우측 아이콘이 들어갈 수 있도록 만들어준 span 태그에 position: absolute 지정

아이콘 이미지는 네이버 소스 이미지 사용

JS를 통해 비밀번호가 오류일 때,
알맞게 입력되었을 때 아이콘이 다르게 표기되도록
클래스를 따로 작성

(해당 클래스를 JS로 토글하며 조절)

3. 파트별 구현내용 - ② 비밀번호

실제 구동 모양

127.0.0.1:5501 내용:
필수 정보입니다.

아이디

@naver.com

비밀번호

확인

값이 없을 때

127.0.0.1:5501 내용:
8~16자 영문 대 소문자, 숫자, 특수문자를 사용하세요.

아이디

@naver.com

비밀번호

비밀번호 재확인

확인

입력 문자 수가 적거나
지정되지 않은 문자 사용

비밀번호

비밀번호

..

사용불가

비밀번호

.....


안전

경고창 확인 클릭 후
적색 아이콘 / 메시지 출력

올바르게 입력 시
녹색 아이콘 / 메시지 출력

3. 파트별 구현내용 - ③ 비밀번호 재확인

비밀번호 재확인



```
<div class="form-item">
  <label for="user_pw_re"><h3>비밀번호 재확인</h3></label>
  <div class="form-input">
    <input type="password" name="user_pw_re" id="user_pw_re">
    <span class="pw-icon2 input-r-side"></span>
  </div>
</div>
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
input ...

아이콘 자리를 빈 span 태그로 작성

3. 파트별 구현내용 - ③ 비밀번호 재확인

```
/**
 * >> 두 값의 동일여부 확인
 * @param {string} origin 비교 대상의 css 선택자
 * @param {string} compare 검사 대상의 css 선택자
 * @param {string} msg 값이 일치하지 않을 경우 표시될 메시지
 */
compareTo(origin, compare, msg) {
  this.value(origin, msg);
  this.value(compare, msg);

  let org = document.querySelector(origin).value.trim();
  let cpr = document.querySelector(compare).value.trim();

  if (org !== cpr) {
    throw new Error(msg, origin);
  }

  return true;
}
```

RegexHelper.js

이전에 입력한 패스워드와 일치하는지 내용 확인

3. 파트별 구현내용 - ③ 비밀번호 재확인

비밀번호



```
/**
 * >> 비밀번호 일치여부 체크
 * 검사 내용 : 값의 유무 / 값의 일치 여부
 *
 * 값이 아무것도 입력되지 않았을 때는 회색 자물쇠
 * 값이 입력되었고 일치할 때는 녹색 자물쇠로 변화
 */
document.querySelector('#user_pw_re').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();
  const icon = document.querySelector('.pw-icon2');

  try {
    regexHelper.value('#user_pw_re', '필수 정보입니다.');
```

regexHelper.compareTo('#user_pw', '#user_pw_re', '비밀번호가 일치하지 않습니다.')

```
  } catch (e) {
    alert(e.message);
    icon.classList.remove('correct');
    return;
  }

  // 아이콘
  icon.classList.add('correct');
});
```

formValueChecker.js

유효성 검사 종료 후,
비밀번호가 일치하면 아이콘이 변경되도록 작성

수정하여 일치하지 않게 되면 아이콘도 회색으로 돌아옴

3. 파트별 구현내용 - ③ 비밀번호 재확인

```
.pw-icon2 {  
  top: 37px;  
  right: 15px;  
  display: inline-block;  
  width: 24px;  
  height: 24px;  
  background: url(../img/naver-icon.png);  
  background-position: -27px 0;  
  background-size: 125px;  
  
  &.correct {  
    background-position: -81px 0;  
    background-size: 125px;  
  }  
}
```

사용한 백그라운드 이미지
(네이버 소재)



빈 span 태그에 백그라운드 이미지로 아이콘을 넣어두고,
correct 클래스 유무에 따라 서식이 변경되도록 작성

3. 파트별 구현내용 - ③ 비밀번호 재확인

실제 구동 내용

비밀번호

.....

비밀번호 재확인

127.0.0.1:5501 내용:
필수 정보입니다.

확인

값이 없을 때

비밀번호

.....

비밀번호 재확인

..

127.0.0.1:5501 내용:
비밀번호가 일치하지 않습니다.

확인

비밀번호 불일치

비밀번호

.....

비밀번호 재확인

.....

안전

비밀번호 일치

3. 파트별 구현내용 - ④ 이름

이름

```
<div class="form-group">
  <div class="form-item">
    <label for="user_name"><h3>이름</h3></label>
    <div class="form-input">
      <input type="text" name="user_name" id="user_name">
    </div>
  </div>
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
input ...

각각의 label은 input, select와 연결되어 label 클릭 시 해당 input, select에 포커스

3. 파트별 구현내용 - ④ 이름

이름

```
/**
 * 이름 : 한글, 영문 대/소문자 (특수기호/공백x)
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg      표시할 메시지
 */
checkName(selector, msg) {
  return this.field(selector, msg, /^[가-힣a-zA-Z]*$/);
}
```

RegexHelper.js

함수, 정규표현식에 따라 에러 체크

3. 파트별 구현내용 - ④ 이름

이름

```
/**
 * >> 이름 유효성 체크
 * 검사 내용 : 값의 유무 / 값의 유효성
 *
 * 글자수에는 제한이 없지만 한글의 경우 자음/모음 단독 사용 불가(ex: ㄱㄴㅇ)
 */
document.querySelector('#user_name').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();

  try {
    regexHelper.value('#user_name', '필수 정보입니다. ');
    regexHelper.checkName('#user_name', '한글과 영문 대 소문자를 사용하세요. (특수기호, 공백 사용 불가)');
  } catch (e) {
    alert(e.message);
    return;
  }
});
```

formValueChecker.js

유효성 검사에 따라 오류처리 구문 작성

3. 파트별 구현내용 - ④ 이름

이름

```
input,
select,
button {
  width: 100%;
  height: 50px;
  font-size: 14px;;
}

input,
select{
  padding: 10px 14px;
  border: 1px solid #{$input-border-color};
}

input:focus,
select:focus {
  border: 1px solid #{$button-border-color};
  outline: none;
}

input::placeholder {
  color: #{$font-color1};
}
```

다른 input 박스와 유사하게 작성됨

3. 파트별 구현내용 - ④ 이름

실제 구동 내용

This screenshot shows the login form with the password field empty. A tooltip is visible above the password field, displaying the IP address '127.0.0.1:5501' and the message '필수 정보입니다.' (Required information). The name field is empty and has a green border. The '확인' (Confirm) button is visible.

비밀번호 재확인

127.0.0.1:5501 내용:
필수 정보입니다.

.....

확인

이름

|

값이 없을 때

This screenshot shows the login form with an invalid password entered. A tooltip is visible above the password field, displaying the IP address '127.0.0.1:5501' and the message '한글과 영문 대 소문자를 사용하세요. (특수기호, 공백 사용 불가)' (Use Korean and English uppercase and lowercase letters. (No special characters, no spaces)). The name field contains the text '마르' and has a green border. The '확인' (Confirm) button is visible.

비밀번호 재확인

127.0.0.1:5501 내용:
한글과 영문 대 소문자를 사용하세요. (특수기호, 공백 사용 불가)

.....

확인

이름

마르

잘못된 값을 입력했을 때

This screenshot shows the login form with a valid password entered. The name field contains the text '방미쇠' and has a green border. The '확인' (Confirm) button is visible.

이름

방미쇠

정상 입력 시

3. 파트별 구현내용 - ⑤ 생년월일

생년월일

년(4자)	월 ▼	일
-------	--------	---

기본 DOM 구조 및 HTML 코드

```
<div class="form-item">
  <label for="birth-yy"><h3>생년월일</h3></label>
  <div class="form-input flex-row">
    <input type="text" name="birth-yy" id="birth-yy" class="birth birth-yy" placeholder="년(4자)" maxlength="4">
    <select id="birth-mm" class="birth birth-mm">...
  </select>
    <input type="text" name="birth-dd" id="birth-dd" class="birth birth-dd" placeholder="일" maxlength="2">
  </div>
</div>
```

form › div.form-group › div.form-item › label
input ...

maxlength 속성으로 최대 글자 수를 4글자 / 2글자 제한함

3. 파트별 구현내용 - ⑤ 생년월일

생년월일

년(4자)	월 ▼	일
-------	--------	---

```
/**
 * 생년월일 : 숫자
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg 표시할 메시지
 */
checkBirth(selector, msg) {
  return this.field(selector, msg, /^[0-9]*$/);
}
```

RegexHelper.js

함수, 정규표현식에 따라 에러 체크
날짜는 임의로 숫자에 대해 체크

3. 파트별 구현내용 - ⑤ 생년월일

```
/**
 * >> 생년월일 유효성 체크
 * 검사 내용 : 값의 유무 / 값의 유효성
 *
 * 어떤 칸을 클릭하든 년 > 월 > 일 순서로 유효성 체크하도록 설정
 * 만약 나이 입력값이 만 14세 이하일 경우 보호자 동의 화면 출력
 */
document.querySelectorAll('.birth').forEach(v => {
  v.addEventListener('blur', e => {
    const regexHelper = new RegexHelper();

    try {
      regexHelper.value('#birth-yy', '태어난 년도 4자리를 정확하게 입력하세요.')
      regexHelper.valueLen('#birth-yy', 4, 4, '태어난 년도 4자리를 정확하게 입력하세요.')
      regexHelper.value('#birth-mm', '태어난 월을 선택하세요.')
      regexHelper.value('#birth-dd', '태어난 일(날짜) 2자리를 정확하게 입력하세요.')
      regexHelper.valueLen('#birth-dd', 2, 2, '태어난 일(날짜) 2자리를 정확하게 입력하세요.')
    } catch (e) {
      alert(e.message);
      return;
    }

    // 만 14세 이하 여부 체크
    // 해당할 경우 해당 컨테이너들의 display 상태 변화
    const yy = document.querySelector('#birth-yy').value;
    const mm = document.querySelector('#birth-mm').value;
    const dd = document.querySelector('#birth-dd').value;
    const today = new Date();

    let userAge = today.getFullYear() - yy;
    const m = today.getMonth() - mm;
    if (m < 0 || (m === 0 && today.getDate() < dd)) {
      userAge--;
    }

    if (userAge < 14) {
      const tap = document.querySelectorAll('.tap-container');
      tap.forEach(v => {
        v.classList.toggle('hide')
      });
    }
  })
})
```

생년월일

년(4자)

월

▼

일

formValueChecker.js

입력값이 숫자인지 검사하고,
입력값을 계산했을 때 일정 나이보다 어릴 경우 별도의
폼(보호자 폼)이 표시되도록 작성

만 나이 계산식 참고 :

<https://hianna.tistory.com/410>

3. 파트별 구현내용 - ⑤ 생년월일

생년월일

년(4자)	월 ▼	일
-------	--------	---

```
input,
select{
  padding: 10px 14px;
  border: 1px solid #input-border-color;
}

input:focus,
select:focus {
  border: 1px solid #button-border-color;
  outline: none;
}

input::placeholder {
  color: #font-color1;
}

select {
  background-image: url(../img/arrow.png);
  background-repeat: no-repeat;
  background-position: right;
}

button {
  border: 1px solid #button-border-color;
  background-color: #button-color;
  color: #main-font-color2;
}
```

다른 input 영역과 동일하게 작성됨

3. 파트별 구현내용 - ⑤ 생년월일

실제 구동 내용

127.0.0.1:5501 내용:
태어난 년도 4자리를 정확하게 입력하세요.

생년월일

년(4자) 2 일

확인

년도 미입력 시

127.0.0.1:5501 내용:
태어난 일(날짜) 2자리를 정확하게 입력하세요.

생년월일

1999 6 일

확인

날짜 미입력 시

만 14세 미만 나이로
입력했을 경우
하단 숨겨진 폼이 열림

생년월일

2010 6 02

성별

성별

본인 확인 이메일(선택)

선택입력

휴대전화인증 아이핀 인증

☐ 아래 약관에 모두 동의합니다.
☐ 개인정보 이용 ☐ 고유식별정보 처리
☐ 통신사 이용약관 ☐ 인증사 이용약관
☐ 네이버 개인정보 수집

보호자 이름

보호자 생년월일

년(4자) 월 일

보호자 성별/국적

성별 내국인

3. 파트별 구현내용 - ⑤ 생년월일

생년월일

2010 6 02

성별

성별

본인 확인 이메일(선택)

선택입력

휴대전화인증 아이핀 인증

☐ 아래 약관에 모두 동의합니다.

☐ 개인정보 이용 ☐ 고유식별정보 처리

☐ 통신사 이용약관 ☐ 인증사 이용약관

☐ 네이버 개인정보 수집

보호자 이름

보호자 생년월일

년(4자) 월 일

보호자 성별/국적

성별 내국인

Hide 클래스 토글링하여 작동

```
<!-- under 14 years old -->
<div class="tap1 tap-container hide">
  <div class="tap-group">
    <div>
      <ul class="tap-title flex-row">
        <li class="tap-button active"><a href="">휴대전화인증</a></li>
        <li class="tap-button"><a href="">아이핀 인증</a></li>
      </ul>
      <div class="contract">
        <div class="contract-title">
          <input type="checkbox" id="contract-all"><label for="contract-all">아래 약관
        </div>
        <div class="contract-container flex-row">
          <ul class="contract-content">
            <li><input type="checkbox" id="contract-per" class="contract"><label for="
            <li><input type="checkbox" id="contract-tel" class="contract"><label for="
            <li><input type="checkbox" id="contract-naver" class="contract"><label for=
          </ul>
          <ul class="contract-content">
            <li><input type="checkbox" id="contract-id" class="contract"><label for="
            <li><input type="checkbox" id="contract-com" class="contract"><label for="
          </ul>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
.hide {
  display: none;
}
```

약관 체크박스 (모두 체크)

```
// 약관 전체 체크
document.querySelector('#contract-all').addEventListener('change', e => {
  document.querySelectorAll('.contract').forEach(v => {
    v.checked = e.currentTarget.checked;
  })
});
```

3. 파트별 구현내용 - ⑥ 성별

성별

성별 ▼

```
<div class="form-item">
  <label for="gender"><h3>성별</h3></label>
  <div class="form-input">
    <select name="gender" id="gender">
      <option value="">성별</option>
      <option value="1">남자</option>
      <option value="2">여자</option>
      <option value="3">선택 안함</option>
    </select>
  </div>
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
select...

3. 파트별 구현내용 - ⑥ 성별

성별

성별



```
/**
 * >> 값이 존재하는지 확인
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg      값이 존재하지 않을 경우 표시될 메시지
 */
value(selector, msg) {
  const content = document.querySelector(selector).value;

  if (content === undefined || content === null || (typeof content === 'string' && content.length === 0)) {
    throw new ErrorException(msg, selector);
  }

  return true;
}
```

RegexHelper.js

Select 설정되므로 값 유무만 판별

3. 파트별 구현내용 - ⑥ 성별

성별

성별 ▼

```
/**
 * >> 성별 입력값 체크
 * 검사 내용 : 값의 선택 여부
 *
 * 특이사항 없음
 */
document.querySelector('#gender').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();

  try {
    regexHelper.value('#gender', '필수 정보입니다. ');
  } catch (e) {
    alert(e.message);
    return;
  }
});
```

formValueChecker.js

값이 제대로 선택되었는지 체크

3. 파트별 구현내용 - ⑥ 성별

select 기본 설정

```
input,
select,
button {
  width: 100%;
  height: 50px;
  font-size: 14px;;
}

input,
select{
  padding: 10px 14px;
  border: 1px solid #input-border-color;
}

input:focus,
select:focus {
  border: 1px solid #button-border-color;
  outline: none;
}

input::placeholder {
  color: #font-color1;
}
```

성별

성별



우측 화살표는 배경 이미지로 삽입함

```
select {
  background-image: url(../img/arrow.png);
  background-repeat: no-repeat;
  background-position: right;
}
```

3. 파트별 구현내용 - ⑥ 성별

실제 구동 내용

성별

선택 안함

정상 화면

성별

성별

방미소

생년월일

년(4자)

성별

127.0.0.1:5501 내용:
필수 정보입니다.

확인

성별

성별

다시 성별(value = 0) 선택 시
오류 메시지 팝업

3. 파트별 구현내용 - ⑦ 이메일

본인 확인 이메일(선택)

선택입력

```
<div class="form-item">
  <label for="email"><h3>본인 확인 이메일</h3></label><span class="choice">(선택)</span>
  <div class="form-input">
    <input type="text" name="email" id="email" placeholder="선택입력">
  </div>
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
input ...

따로 서식을 넣을 문자를 span으로 분리

3. 파트별 구현내용 - ⑦ 이메일

본인 확인 이메일(선택)

선택입력

RegexHelper.js

```
/**
 * 이메일 : 한글, 영문 대/소문자 (특수기호/공백x)
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg 표시할 메시지
 */

checkEmail(selector, msg) {
  return this.field(selector, msg, /^[a-z0-9-]+(?:\.[a-z0-9-]+)*@((?:[a-z0-9-]+\.)+[a-z0-9-]{2,6})\.[a-z]{2,4}$/i);
}
```

함수, 정규표현식에 따라 에러 체크

3. 파트별 구현내용 - ⑦ 이메일

본인 확인 이메일(선택)

선택입력

```
/**
 * >> 이메일 유효성 체크
 * 검사 내용 : 값의 유효성
 *
 * 필수입력값이 아니므로 값의 유무는 검사하지 않음
 */
document.querySelector('#email').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();
  const current = e.currentTarget.value;

  if (current == "") {
    return true;
  }

  try {
    regexHelper.checkEmail('#email', '이메일 주소를 다시 확인해주세요.');
```

formValueChecker.js

필수 입력값이 아니므로 빈칸일 때
경고창이 뜨지 않도록 작성

3. 파트별 구현내용 - ⑦ 이메일

```
input,
select{
  padding: 10px 14px;
  border: 1px solid #input-border-color;
}

input:focus,
select:focus {
  border: 1px solid #button-border-color;
  outline: none;
}

input::placeholder {
  color: #font-color1;
}

select {
  background-image: url(../img/arrow.png);
  background-repeat: no-repeat;
  background-position: right;
}

button {
  border: 1px solid #button-border-color;
  background-color: #button-color;
  color: #main-font-color2;
}
```

본인 확인 이메일(선택)

선택입력

다른 input 영역과 동일하게 작성됨

3. 파트별 구현내용 - ⑦ 이메일

실제 구동 내용

본인 확인 이메일(선택)

빈칸인 채로
blur 해도 오류창 팝업 없음

성별

본인 확인 이메일(선택)

L O R L R L O R L

127.0.0.1:5501 내용:
이메일 주소를 다시 확인해주세요.

확인

성별

본인 확인 이메일(선택)

smdf@

127.0.0.1:5501 내용:
이메일 주소를 다시 확인해주세요.

확인

유효성 검사 식에 위반할 경우 오류창

본인 확인 이메일(선택)

정상 입력 시

3. 파트별 구현내용 - ⑧ 휴대전화

휴대전화

대한민국 +82

```
<div class="tap2 tap-container">
  <div class="form-item">
    <label for="phone"><h3>휴대전화</h3></label>
    <div class="form-input">
      <select name="phone" id="phone">
        <option value="">대한민국 +82</option>
        <option value="">일본 +81</option>
        <option value="">중국 +86</option>
      </select>
    </div>
  </div>
</div>
```

기본 DOM 구조 및 HTML 코드

form › div.form-group › div.form-item › label
select ...

휴대전화 부분은 하나의 form-item안에 작성됨

3. 파트별 구현내용 - ⑧ 휴대전화

휴대전화

대한민국 +82

```
/**
 * >> 값이 존재하는지 확인
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg      값이 존재하지 않을 경우 표시될 메시지
 */
value(selector, msg) {
  const content = document.querySelector(selector).value;

  if (content === undefined || content === null || (typeof content === 'string' && content.tr
    throw new ErrorException(msg, selector);
  }

  return true;
}
```

RegexHelper.js

select구조상 반드시 하나의 값을 선택하게 되므로 별도의 검사식을 작성하지 않았으나

값의 존재 유무는 해당 코드로 확인 가능함

3. 파트별 구현내용 - ⑧ 휴대전화

```
input,
select,
button {
  width: 100%;
  height: 50px;
  font-size: 14px;;
}

input,
select{
  padding: 10px 14px;
  border: 1px solid #000000;
}

input:focus,
select:focus {
  border: 1px solid #000000;
  outline: none;
}

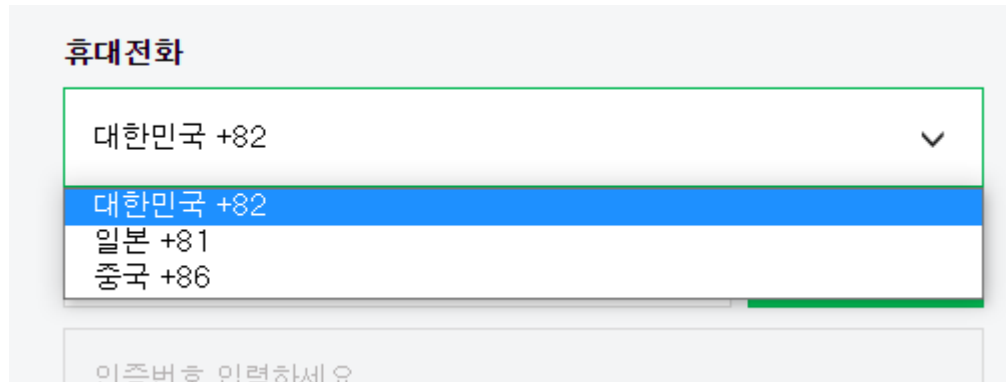
input::placeholder {
  color: #000000;
}

select {
  background-image: url(../img/arrow.png);
  background-repeat: no-repeat;
  background-position: right;
}
```

다른 select 부분과 동일하게 작성됨

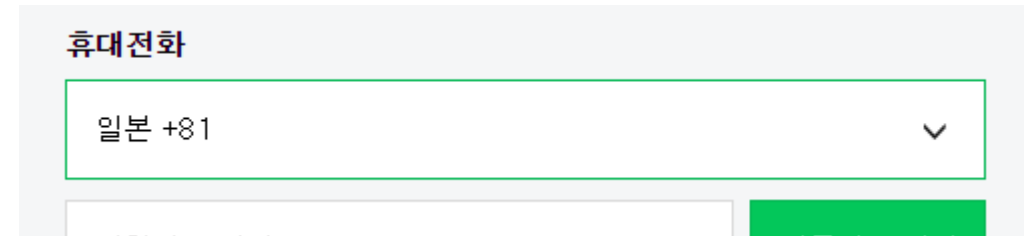
3. 파트별 구현내용 - ⑧ 휴대전화

실제 구동 내용



A screenshot of a web form titled "휴대전화" (Mobile Phone). It features a dropdown menu with a green border. The menu is open, showing a list of country codes: "대한민국 +82" (highlighted in blue), "일본 +81", and "중국 +86". Below the dropdown is a text input field with the placeholder text "이른번호 입력하세요" (Enter the number).

드롭다운 정상적으로 작동함



A screenshot of a web form titled "휴대전화" (Mobile Phone). It features a dropdown menu with a green border. The menu is closed, and the selected option is "일본 +81". Below the dropdown is a text input field with a green button to its right.

값 선택 시

3. 파트별 구현내용 - ⑨ 전화번호 입력창

인증번호 받기

```
<div class="flex-row">  
  <input type="text" name="phoneno" id="phoneno" placeholder="전화번호 입력" >  
  <button type="button" id="get-num-btn">인증번호 받기</button>  
</div>
```

기본 DOM 구조 및 HTML 코드

form > div.form-group > div.form-item > input
button ...

input과 button 병렬 배치

3. 파트별 구현내용 - ⑨ 전화번호 입력창

전화번호 입력

인증번호 받기

```
/**
 * 휴대전화 : 전화번호 형식
 * @param {string} selector 검사 대상의 css 선택자
 * @param {string} msg 표시할 메시지
 */
checkPhone(selector, msg) {
  const content = document.querySelector(selector).value;
  const arr = content.split('');
  let num = '';
  arr.forEach(v => {
    if (v !== '-') {
      num += v;
    }
  })
}
```

RegexHelper.js

함수, 정규표현식에 따라 에러 체크

3. 파트별 구현내용 - ⑨ 전화번호 입력창

전화번호 입력

인증번호 받기

```
/**
 * >> 휴대전화 유효성 체크
 * 검사 내용 : 값의 유무 / 값의 유효성
 *
 */
document.querySelector('#phoneno').addEventListener('blur', e => {
  const regexHelper = new RegexHelper();

  try {
    regexHelper.value('#phoneno', '필수 정보입니다.');
```

```
    regexHelper.checkPhone('#phoneno', '형식에 맞지 않는 번호입니다.');
```

```
  } catch (e) {
    alert(e.message);
    return;
  }

  const num = regexHelper.checkPhone('#phoneno', '형식에 맞지 않는 번호입니다.');
```

```
  e.currentTarget.value = num;
});
```

formValueChecker.js

정상적인 값이지만 숫자 중간에 '-'가 들어가 있는 경우
'-'를 제외하고 창에 출력되도록 작성

3. 파트별 구현내용 - ⑨ 전화번호 입력창

전화번호 입력

인증번호 받기

```
input:focus,  
select:focus {  
  border: 1px solid $button-border-color;  
  outline: none;  
}  
  
input::placeholder {  
  color: $font-color1;  
}  
  
select {  
  background-image: url(../img/arrow.png);  
  background-repeat: no-repeat;  
  background-position: right;  
}  
  
button {  
  border: 1px solid $button-border-color;  
  background-color: $button-color;  
  color: $main-font-color2;  
}
```

다른 input, button 부분과 동일하게 작성됨

3. 파트별 구현내용 - ⑨ 전화번호 입력창

실제 구동 내용

휴대전화

127.0.0.1:5501 내용:
필수 정보입니다.

확인

전화번호 입력

인증번호 받기

값이 없을 때

휴대전화

127.0.0.1:5501 내용:
형식에 맞지 않는 번호입니다.

확인

123

인증번호 받기

잘못된 값을
입력했을 때

휴대전화

대한민국 +82

01012345678

인증번호 받기

정상 작동 시
(‘-’ 삭제)

3. 파트별 구현내용 - ⑩ 인증번호 입력창

인증번호 입력하세요

```
<input type="text" name="authno" id="authno" placeholder="인증번호 입력하세요" disabled>
```

form › div.form-group › div.form-item › label
input ...

기본 DOM 구조 및 HTML 코드

disabled로 설정되었으며 별도의 JS 코드는
작성되지 않음

3. 파트별 구현내용 - ⑩ 인증번호 입력창

인증번호 입력하세요

```
input:focus,  
select:focus {  
  border: 1px solid $button-border-color;  
  outline: none;  
}  
  
input::placeholder {  
  color: $font-color1;  
}  
  
select {  
  background-image: url(../img/arrow.png);  
  background-repeat: no-repeat;  
  background-position: right;  
}  
  
button {  
  border: 1px solid $button-border-color;  
  background-color: $button-color;  
  color: $main-font-color2;  
}
```

다른 input과 동일하게 작성됨