TEAM PROJECT

국가 제공 (공공API) 데이터를 활용한 응용 앱 구현

국립암데이터센터 API를 이용한 Cancer Dashboard

Team member: 배석찬, 박진서, 유흥일, 윤종식, 전영진

사이트 바로가기

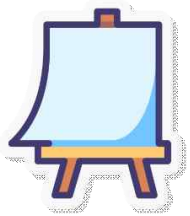
index

사이트 바로가기

PROJECT PLAN

사이트 바로가기

PROJECT PLAN

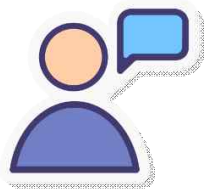


1. 주제 선정

2. 기획

디자인 컨셉 &

데이터 정보 조사



3. 역할 분담

화면 디자인 설계

Data 구조 설계

4. Front-end (Pug / Sass)

JSON parsing

Chart.js

5. 화면 기능구현

데이터 반영

디자인 변경 반영

6. 문제점 개선 및 합치기

7. 코드 점검 및 보안

피드백 반영

문서 작성

사이트 바로가기

PROJECT PLAN

DAY 1

- 주제 선정
- 기획 (디자인 컨셉 & 데이터 정보 조사)

DAY 2

- 역할 분담
- 화면 디자인 설계
- Data 구조 설계

DAY 3

- Front-end (Pug / Sass)
- JSON parsing
- Chart.js

DAY 4

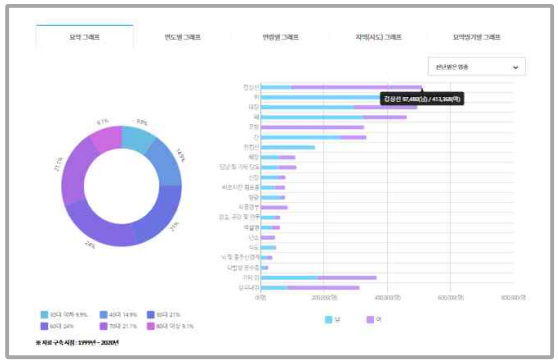
- 화면 기능구현
- 실시간 데이터 반영
- 디자인 변경 반영
- 문제점 개선

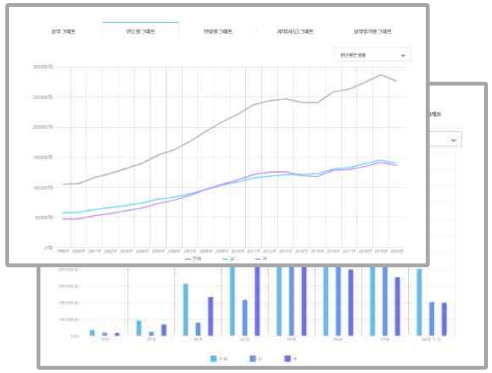
DAY 5

- 코드 점검 및 보안
- 피드백 반영
- 문서 작성

사이트 바로가기

PROJECT PLAN



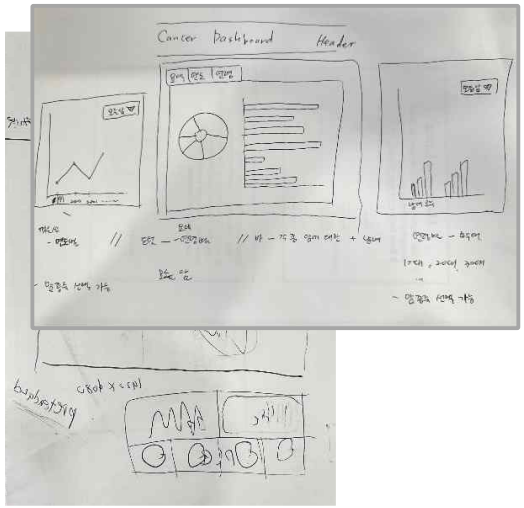


레퍼런스 조사

- 질병정보 앱에 포함되어야 할 항목 및 구현 방식을 알아보기 위하여 레퍼런스 조사.
- 레퍼런스 조사가 끝난 이후 구현할 항목들을 재정의하며 디자인 작업을 시작.

스케치

- 앱에서 보여줄 기능들을 작성하고, 영역을 어떻게 배치할 것인지 시안을 잡음.
- 그래프 종류와 내용들을 정리해서 표현 방식을 와이어프레임으로 정리.
- 대시보드 레퍼런스를 참고하여 디자인 작업을 함.

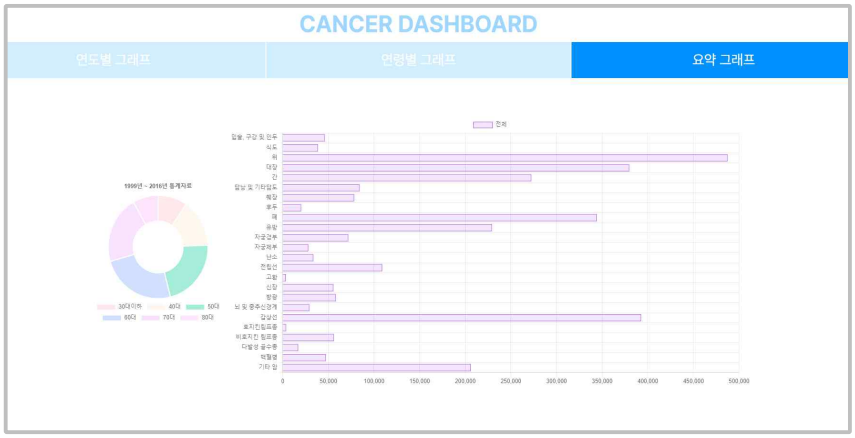


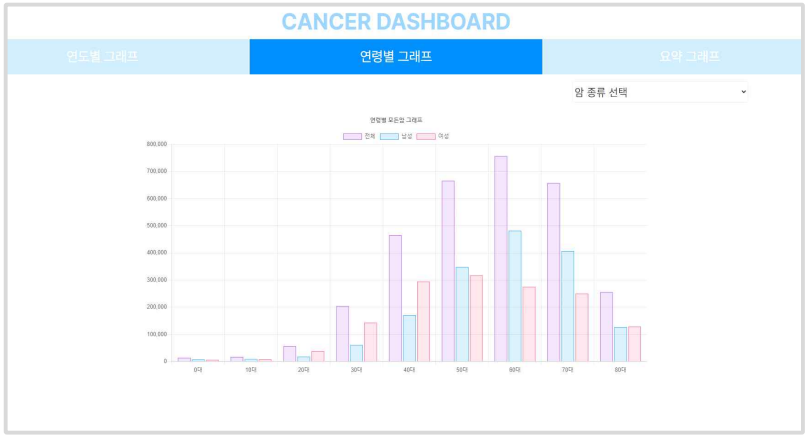
3차 팀플 주제 : 건강보험관리공단 API를 이용한 병원정보관리(암)			
팀장	배석찬		
팀원	박진서, 유홍일, 윤종식, 전영진		
대상암	폐암	• 다른 암 데이터 필요하면 후에 보충	
역할	박진서	와이어프레임, 프런트 (HTML, CSS) (PUG와 SASS 사용)	
	전영진	와이어프레임, 퍼블리싱 (프런트)	
	배석찬	json parsing	
	유홍일	json parsing	
	윤종식	ppt 및 서류작업	
작업 데이터 종류	1. 환자 수	연도별	라인그래프로 표현
		나이별	막대그래프로 표현
		성별	막대그래프로 표현
	2. 흡연 여부	과거	원그래프로 표현
		현재	원그래프로 표현
	3. 음주 여부	과거	원그래프로 표현
		현재	원그래프로 표현
	4. 암 병력 여부	원 그래프로 표현	

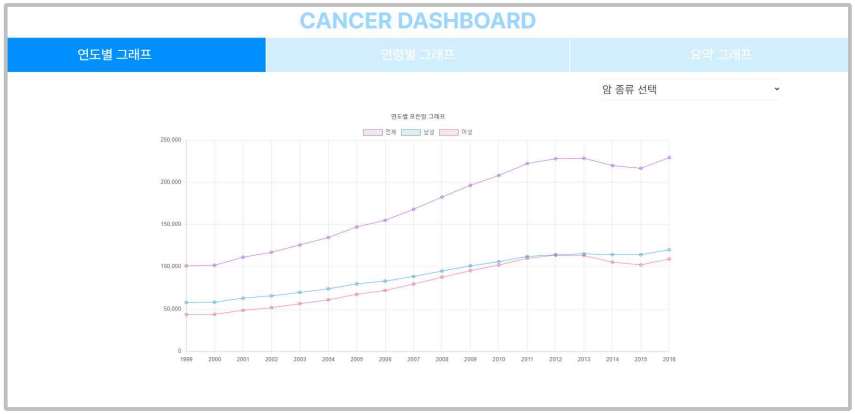
사이트 바로가기

PROJECT PLAN

대시보드 기능 구현 설계







화면 구성

- 대시보드의 형태로 요약, 연령별, 연령별 정보를 각각 원 그래프, 막대 그래프, 선 그래프로 표현.
- 사용자가 각각의 변화를 그래프와 색상으로 인식하도록 시각적으로 표현.
- 사용자가 직관적으로 시스템을 확인할 수 있게 설계.

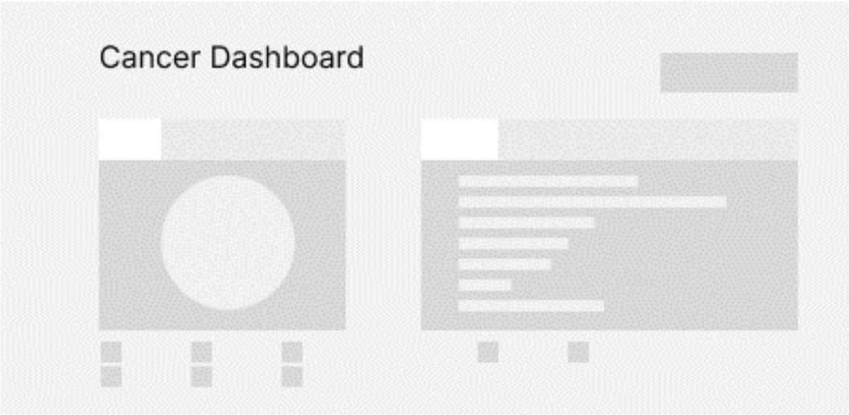
사이트 바로가기

DESIGN

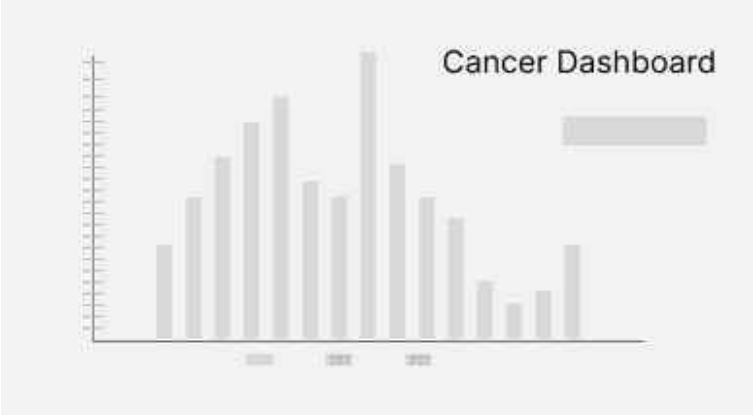
사이트 바로가기

DESIGN

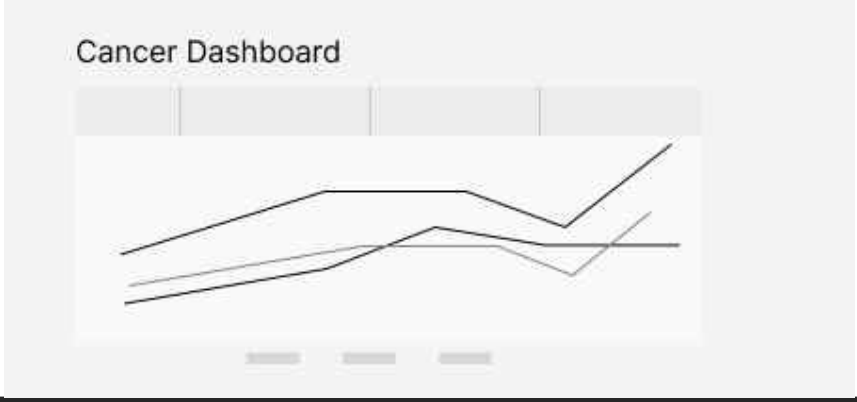
1 요약 Page



3 연령별 Page



2 연도별 Page



사용자 화면

- 각 페이지의 표현 형태를 대략적으로 구성
- 사용자의 직관적 이해에 초점을 두고 화면을 구성

사이트 바로가기

DESIGN

01 Folder color

각 폴더를 구분하는 색채를 사용함으로 시각적 효과를 얻을 수 있도록 함.

#FAC090 #8EB4E3 #A67D9D #796B8D #4E5B76

02 Font

사용자가 정보에 집중할 수 있도록 폰트 적용

NanumBarum Gothic

Montserrat Bold

03 Graph color

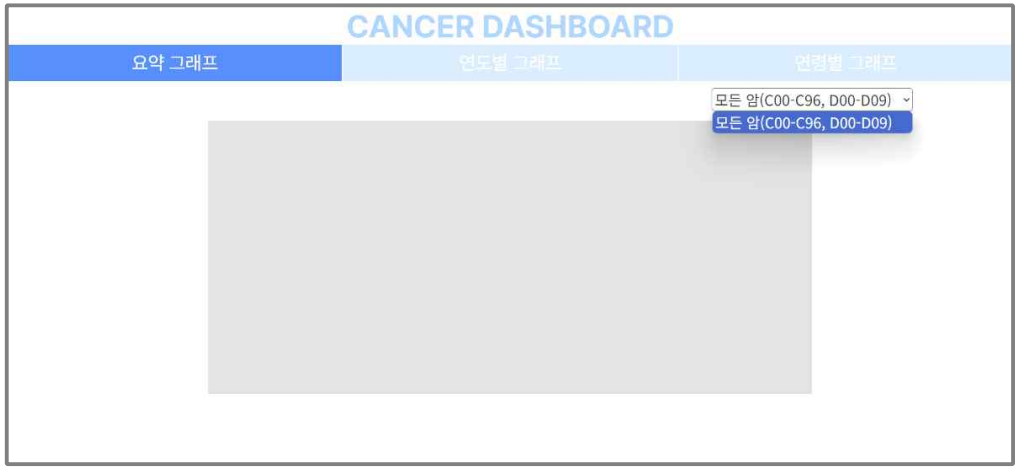
#D81818 #86D1FF #CD8585 #FF8888 #86FF86 #FFB486

사이트 바로가기

DESIGN

Tab menu 사용

Tab menu를 사용하여 사용자 편이와 클릭 편의성을 도모함.



그래프 표시 영역

Object의 성격에 맞는 차트 유형을 선정하여 시각적 접근성을 향상 시켰다.

Select submenu

Select 클릭 시 사용자가 모든 암 종류를 확인할 수 있도록 함.

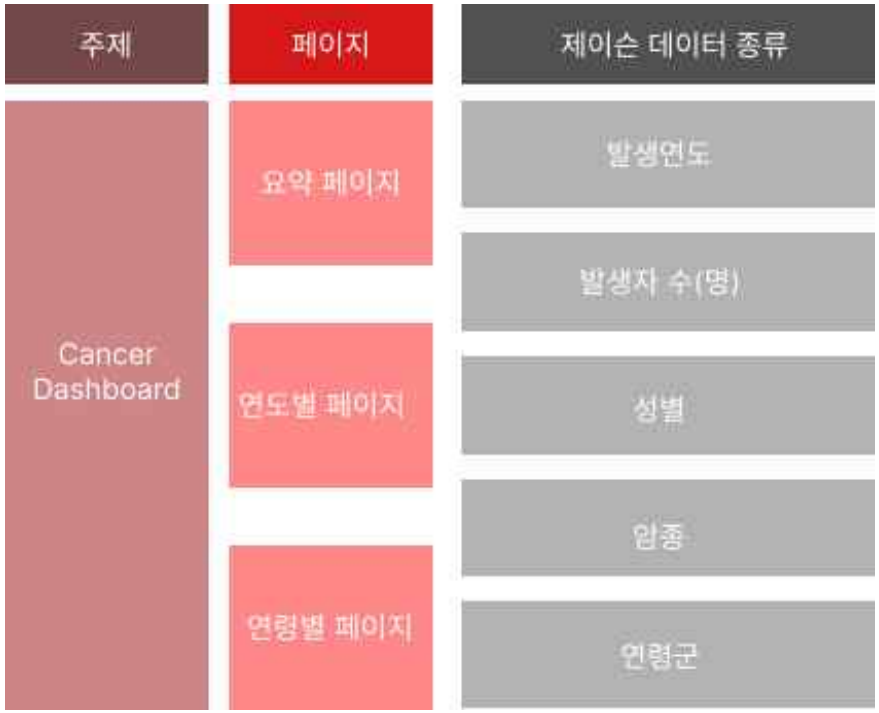
사이트 바로가기

SCHEMA

사이트 바로가기

SCHEMA

01 Data Modeling



{json}

발생 연도
발생자 수(명)
성별
암종
연령군

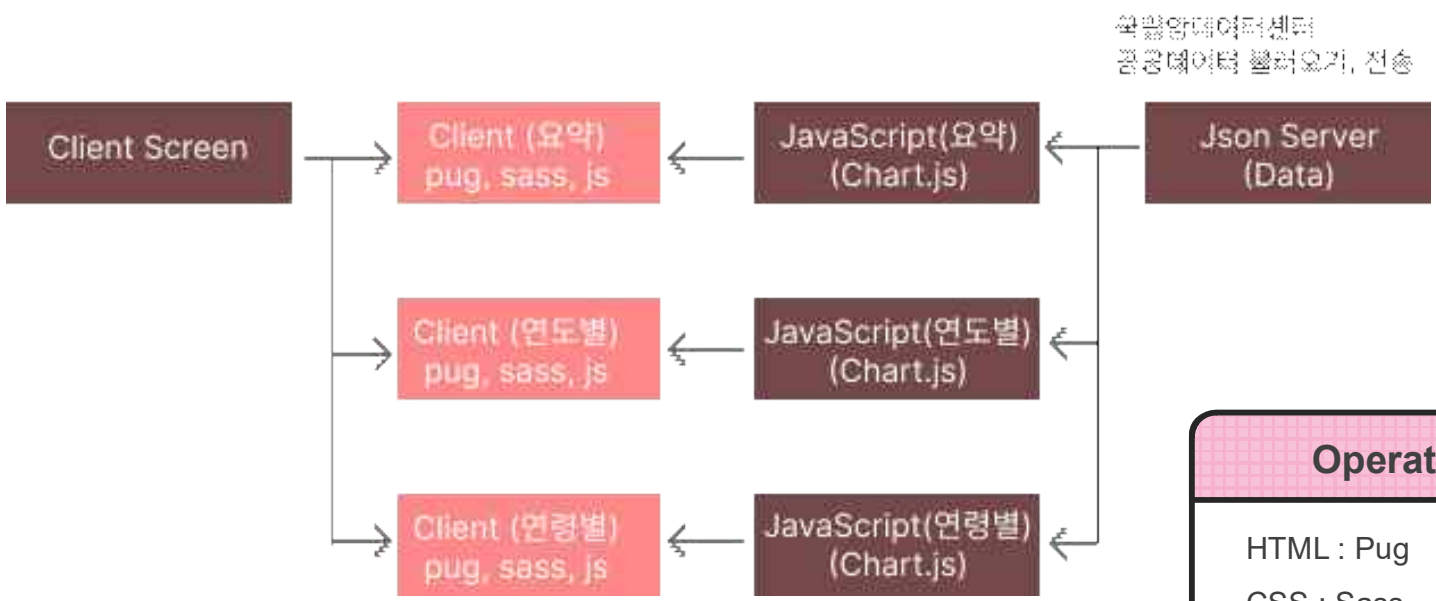
Data Modeling

국립암데이터센터 데이터를 기준으로 발생 연도, 발생자 수, 성별, 암종, 연령군 요소들을 JSON으로 작성

사이트 바로가기

SCHEMA

02 Operating Model



Operating Model

HTML : Pug

CSS : Sass

Script : Javascript(+ Chart.js)

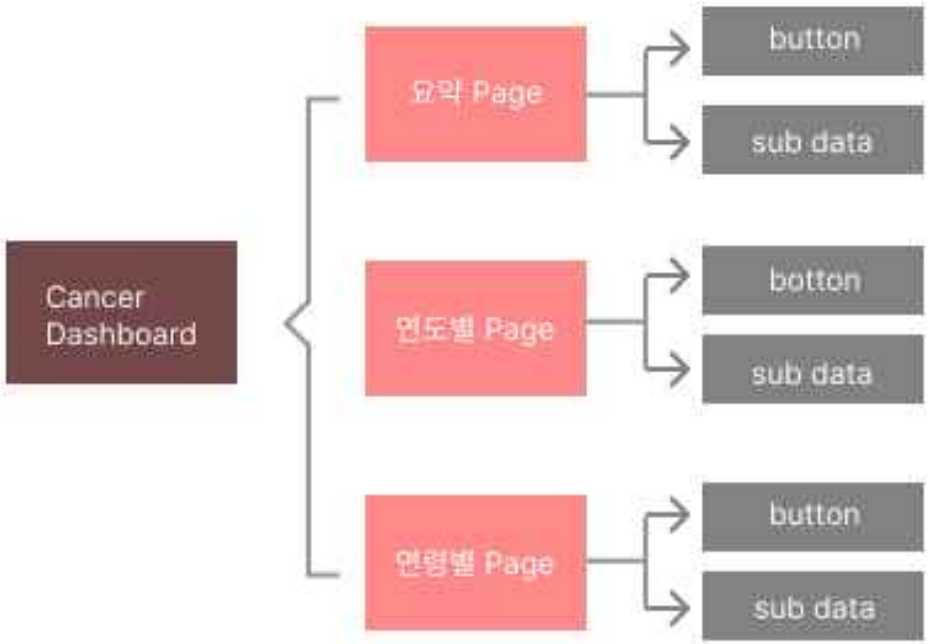
Data Transfer : REST

사이트 바로가기

PAGE STRUCTURE

사이트 바로가기

Page Structure



Page Structure

요약 Page: 모든 암에 대한 정보를 한 눈에보여주는 Page

연도별 Page: 연도별로 남녀별 암 발생 자 수를 보여주는 Page

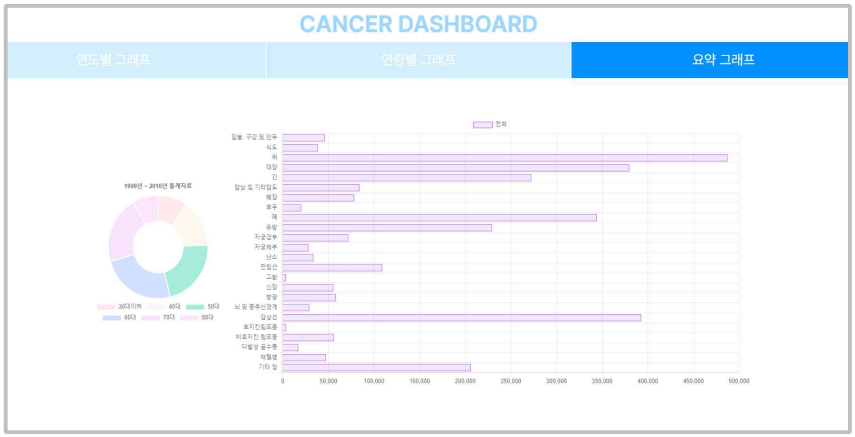
연령별 Page: 연령별 발생자 수를 보여주는 Page

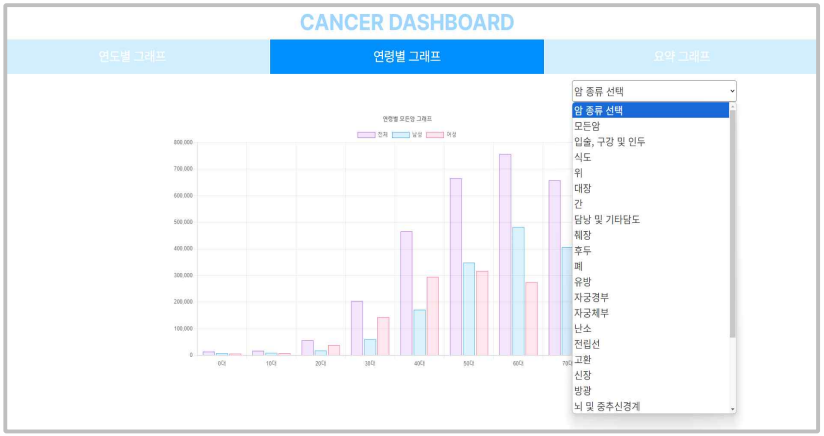
Button: 모든 종류의 암을 선택할 수 있는 Select button

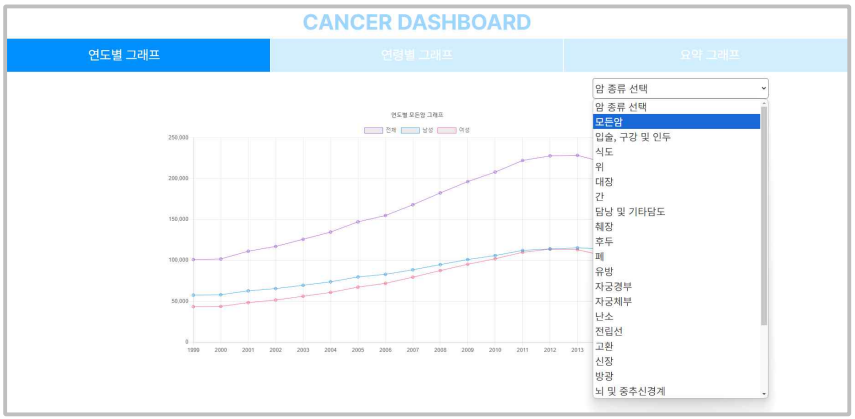
사이트 바로가기

Page Structure

대시보드 구현 결과물



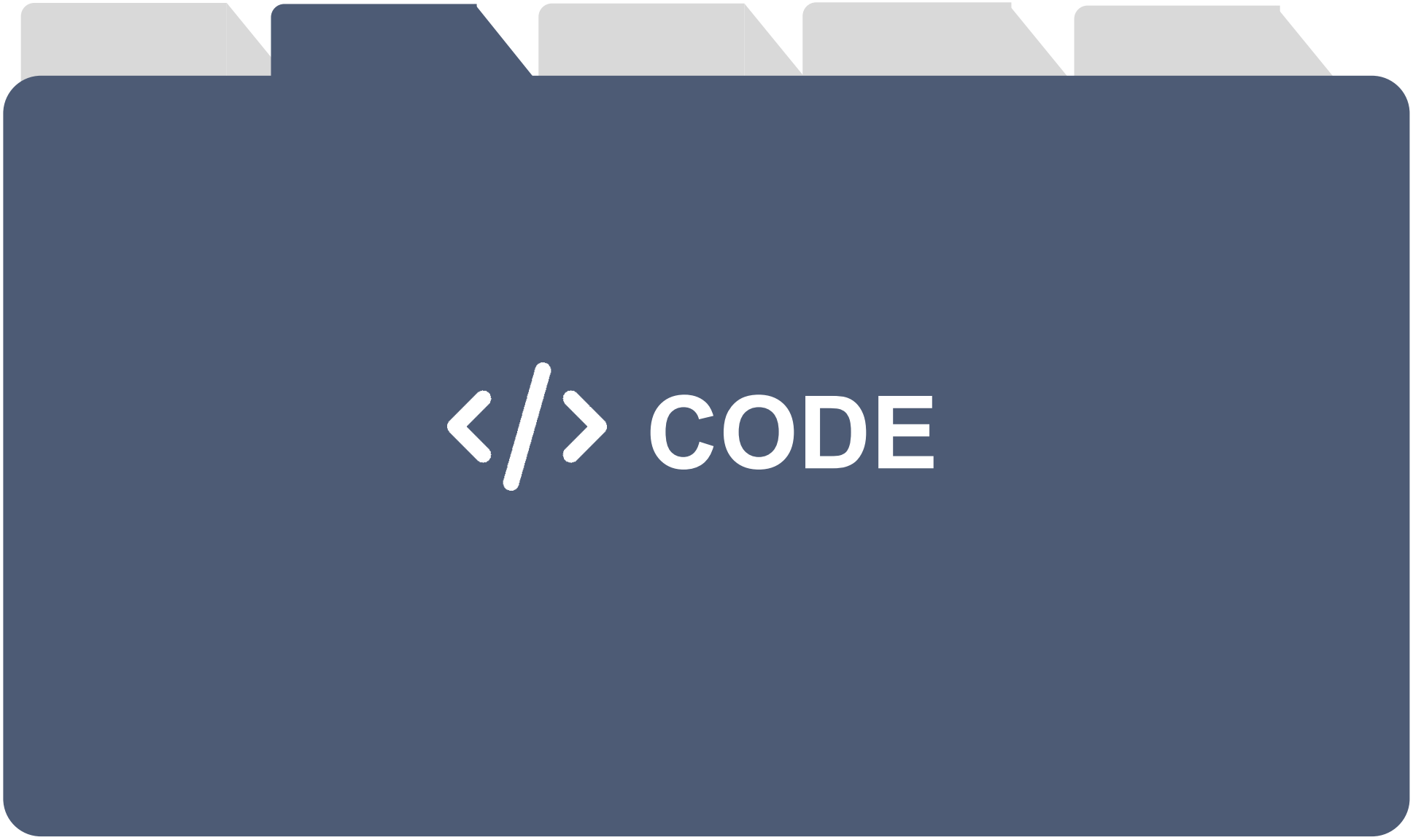




대시 보드 화면 구성

1. 좌측 상단 : 요약 Page
2. 우측 상단 : 연령별 Page
3. 좌측 하단 : 연도별 Page

사이트 바로가기



사이트 바로가기

CODE

01 Select 요소

```
// select에 옵션 추가하는 메소드
addCancerTypes() {
    const cancerTypes = new Set(fetchData.allData.map(entry => entry.암종));
    const selectElements = document.querySelectorAll('.cancerTypeSelect');

    // 암 종류를 각 select 요소에 옵션으로 추가
    selectElements.forEach(selectElement => {
        cancerTypes.forEach(cancerType => {
            const option = document.createElement('option');
            option.value = cancerType;
            option.textContent = cancerType;
            selectElement.appendChild(option);
        });
    });
}
```

Select 요소

암 종류를 선택할 수 있는 select 요소에 암 종류를 추가하고, 선택된 암 종류에 따라 데이터를 변경 및 표시합니다.

사이트 바로가기

CODE

02 Sass 작업

```scss
.sumLi, .yearLi, .ageLi{
    width:33.3%;
    height:100%;
    text-align:center;
    background: #CCE8FF;
    border:1px solid #fff;
    transition: background-color 0.3s, color 0.3s;
    &.active {

        background-color: #008CFF;
        color: white;


    }
    .sumA, .yearA, .ageA{
        width:100%;
        height:100%;
        font-size:1.8rem;
        display:flex;
        justify-content:center;
        align-items:center;
        color: white;
    }
}
```

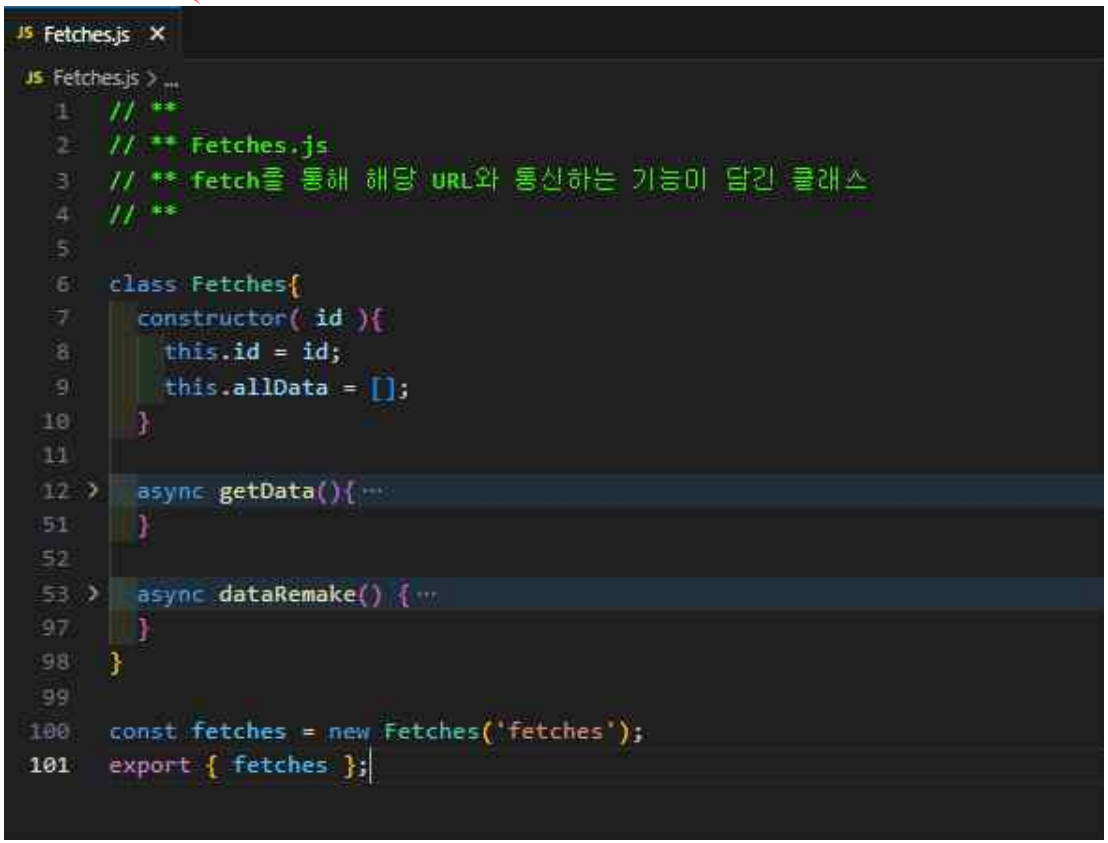
Sass 사용

Sass를 사용하여 부모 요소 선택자의 중괄호 안에 선택자를 선언하고 다시 중괄호로 묶었다.

사이트 바로가기

CODE

03 Fetches.js



Class 작성

API에서 데이터를 받아와서 그 데이터를 사용자가 필요한 형태로 재가공하기 위한 클래스를 작성하였다.

사이트 바로가기

CODE

03 Fetches.js

```
async getData(){
  try {
    // ** 총 데이터를 1만개씩 3파트로 나눠 그 배열을 가져오기
    const urls = [ '1','2','3' ];
    const responses = await Promise.all( urls.map( url => fetch(
      `https://api.odcloud.kr/api/3039563/v1/
      uddi:d9dad564-5ef2-426e-bdbe-fd00aa8c7f23_201906120952?page=${url}&perPage=10000&
      serviceKey=Leyebd2MfA41YgrXmOdVU67BWZOnY6lTAlNCeK7kXHU4KcsScOd8%2FH%2B1NfFJYeLJXVeV2
      rDdwzx%2FzAh4GpuaMg%3D%3D`
    ) ) );

    for (const response of responses) {
      if ( !response.ok ) {
        throw new Error('네트워크 응답이 좋지 못합니다. ' + response.statusText);
      }
    }

    const dataSets = await Promise.all(responses.map(response => response.json()));
```

getData( )

API가 제공하는 총 데이터의 양이 한 번에 가져올 수 있는 개수를 초과하기 때문에 여러 번에 나눠서 데이터를 가져온다.

사이트 바로가기

CODE

03 Fetches.js

```
  // ** 모든 data 평탄화하여 배열에 넣기
  {
    let tempArr = [];
    this.allData = dataSets.map( data => data.data );
    this.allData.forEach( data =>{
      data.forEach( data2 =>{
        tempArr.push( data2 );
      })
    })
    this.allData = tempArr;
  }

  // ** 데이터를 가져온 후에 dataRemake 메소드 호출 및 데이터 전달
  this.dataRemake( this.allData );
} catch (error) {
  console.error("fetch에 문제가 있어요", error);
}

// ** 실행한 결과값을 Promise반환
return new Promise((resolve) => {
  resolve();
});
}
```

getData( )

가져온 데이터를 하나의 배열에 평탄화하여 넣고, dataRemake( ) 메소드를 실행하여 App에 필요한 형태로 재가공한다.

사이트 바로가기

CODE

03 Fetches.js

```
async dataRemake() {
  if ( !this.allData || this.allData.length === 0 ) {
    console.error("아직 fetch 안됐음");
    return;
  }

  const cancerDataByAgeAndYear = {};

  this.allData.forEach (entry => {
    const {
      암종,
      발생연도,
      연령군,
      성,
      "발생자수(명)": 발생자수
    } = entry;
```

dataRemake( )

fetch가 아직 완료가 안 됐을 경우에는 error가 발생하도록 하고, 완료가 되면 그 데이터 덩어리를 forEach를 통해 traveling 하는데, 이때 각 객체가 가지고 있는 키 값들을 비구조화하여 할당한다.

사이트 바로가기

CODE

03 Fetches.js

```
    if (연령군 !== "전체") {
      // ** 연령을 10단위로 묶기
      const ageGroup = `${Math.floor(parseInt(연령군) / 10) * 10}대`;

      // ** 객체 초기화
      if (!cancerDataByAgeAndYear[암종]) {
        cancerDataByAgeAndYear[암종] = {};
      }

      if (!cancerDataByAgeAndYear[암종][발생연도]) {
        cancerDataByAgeAndYear[암종][발생연도] = {};
      }

      if (!cancerDataByAgeAndYear[암종][발생연도][ageGroup]) {
        cancerDataByAgeAndYear[암종][발생연도][ageGroup] = {
          남자: 0,
          여자: 0
        };
      }

      if( 성 === "남녀전체" ){ return };
      let accrualsNumber = ( 발생자수 === "" ) ? ( 0 ) : ( parseInt( 발생자수 ) );
      cancerDataByAgeAndYear[암종][발생연도][ageGroup][성] += accrualsNumber;
    }
  });

  return cancerDataByAgeAndYear;
}
```

dataRemake( )

비구조화하여 할당된 변수들을 객체의 키로 사용하여 자료를 설계하는데 이때 "연령군"의 값이 전체가 아닐 경우에만 작성하도록 한다.

모든 객체들의 초기값을 빈 객체로 만들어주고 가장 내부의 객체의 값을 { 남자 : 0, 여자 : 0 }으로 설정해준다. 그 후 '성'이 '남녀 전체'가 아닌 '남자', '여자'일 경우에만 '남자'와 '여자'의 값이 설정될 수 있도록 한다. 그렇게 하여 최종 완성된 JSON을 반환 시켜준다.

사이트 바로가기

CODE

04 DrawVerticalbar.js

```
async function fetchDataAndProcess() {
  try {
    await fetchData.getData(); // 데이터를 가져오는 비동기 함수 호출
    const cancerData = await fetchData.dataRemake(); // 데이터를 처리하는 비동기 함수 호출

    let selectInfo = document.getElementById('ageCancers');
    let horizontalBarChart;

    selectInfo.addEventListener('change', () => {
      drawChart(selectInfo.value);
    });

    const drawChart = (cancerType) => {
      const totalGenderCountsByAgeGroup = {}; // 이곳에서 초기화

      const dataForCancerType = cancerData[cancerType];
      for (const year in dataForCancerType) {
        for (const ageGroup in dataForCancerType[year]) {
          if (!totalGenderCountsByAgeGroup[ageGroup]) {
            totalGenderCountsByAgeGroup[ageGroup] = { 전체: 0, 남자: 0, 여자: 0 };
          }

          const data = dataForCancerType[year][ageGroup];
          totalGenderCountsByAgeGroup[ageGroup].전체 += (data.남자 + data.여자);
          totalGenderCountsByAgeGroup[ageGroup].남자 += data.남자;
          totalGenderCountsByAgeGroup[ageGroup].여자 += data.여자;
        }
      }
```

Chart.js 활용

파싱해온 데이터를 차트를 그리기 위해 checkbox 의 값을 불러와 해당하는 데이터만 파싱하여 가져온 후 그 값들을 저장한다.

사이트 바로가기

CODE

05 DrawVerticalbar.js

```
const ctxVerticalbar = document.getElementById('ageCanvas').getContext('2d');
horizontalBarChart = new Chart(ctxVerticalbar, {
  type: 'bar',
  data: {
    labels: ['0대', '10대', '20대', '30대', '40대', '50대', '60대', '70대', '80대'],
    datasets: [
      {
        label: '전체',
        data: allData,
        backgroundColor: 'rgba(166, 94, 217, 0.2)',
        borderColor: 'rgba(166, 94, 217, 1)',
        borderWidth: 1
      },
      {
        label: '남성',
        data: mensData,
        backgroundColor: 'rgba(54, 162, 235, 0.2)',
        borderColor: 'rgba(54, 162, 235, 1)',
        borderWidth: 1
      },
      {
        label: '여성',
        data: womenData,
        backgroundColor: 'rgba(255, 99, 132, 0.2)',
        borderColor: 'rgba(255, 99, 132, 1)',
        borderWidth: 1
      }
    ]
```

```
    options: {
      plugins: {
        legend: {
          position: 'top'
        },
        title: {
          display: true,
          text: `연령별 ${selectInfo.value} 그래프`
        }
      }
    }
  });
};
```

Chart.js 활용

재 파싱해와서 저장한 데이터들을 Chart.js 라이브러리를 이용해 원하는 차트로 그려준다.

사이트 바로가기

TOOLS

Editor

Drawing

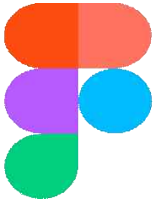
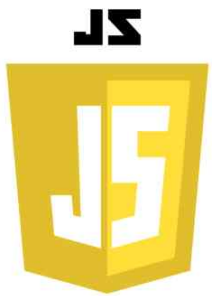
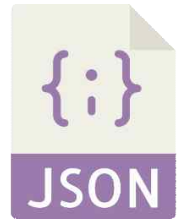


Chart.js

Skills





사이트 바로가기

REVIEW

사이트 바로가기

REVIEW

이 름: 배 석 찬

담 당: Project scheduling /Management
JSON parsing

이번 프로젝트에서 공공 데이터 포털의 암 관련 API를 활용해 대시보드를 구성했습니다. 프로젝트 초반, 회의를 통해 각자의 의견을 적극적으로 나누며 역할을 배분했습니다. 개인의 강점을 살려 역할을 나누었고, 팀원들은 맡은 바를 열정적으로 수행했습니다.

특히, 처음에 찾았던 API가 우리가 필요로 하는 데이터를 제공하지 않아 새로운 API를 찾는 데 시간이 걸렸습니다. 그러나 결국 원하는 데이터를 발견했고, 유용한 참고 사이트 덕분에 대시보드 구성을 비교적 수월하게 진행할 수 있었습니다.

이번 프로젝트의 성공은 팀원들의 헌신과 협업 덕분입니다. 모두의 노력에 깊이 감사드립니다.

사이트 바로가기

REVIEW

이 름: 박 진 서

담 당: Main publishing / Front-end
(Pug / Sass / Javascript)

이번 프로젝트에서 공공 데이터 API를 받아와 값을 뿌려주는 화면 틀을 만드는 작업을 했습니다. Pug와 Sass를 사용하였고, 작업의 효율성과 코드의 가독성이 크게 향상되었습니다.

작업 중 가장 어려웠던 점은 select 요소에 기록된 암 종류를 사용자가 선택할 때 그 종류에 따라 데이터를 실시간으로 변경하는 기능을 구현하는 것이었습니다.

처음에는 Pug로 select를 만들어보려 했으나, 좀더 고민한 후 다른 방법으로 시도해 보자는 생각이 났고, select에 option으로 암 종류를 넣어주는 메소드를 만들어 해결하였습니다.

이번 프로젝트를 통하여 한 걸음 더 성장할 수 있는 계기가 되었습니다.

사이트 바로가기

REVIEW

이 름: 유 흥 일

담 당: JSON parsing
Data structuring

데이터를 재구성하여 사용하는 방법을 배웠습니다.

지금까지는 비어 있는 JSON에 자료구조를 설계하거나 기존에 있던 JSON을 그대로 이용하는 경우가 대부분이었습니다.

하지만 이번 기회를 통해 공공 API를 처음 사용해 보는 경험도 해봤으며, API로부터 받은 데이터에서 저에게 필요한 부분을 필터링하여 새로운 자료구조를 설계해 보았습니다.

그리고 API와의 통신에서 비동기 작업을 동기 작업으로 처리해 주기 위해 async / await 문법과 Promise 문법을 처음 사용해 봤습니다.

자주 사용하는 문법이 아니라 너무 어렵기도 했지만 이번 프로젝트를 통해 여러 번 반복하여 사용하면서 익숙해지는 기회를 가졌습니다.

앞으로도 비동기 작업을 처리할 기회가 있어도 크게 당황하지 않도록 여러 번 반복하여 완벽하게 숙달해야겠다고 다짐했습니다.

사이트 바로가기

REVIEW

이 름: 윤 종 식

담 당: Project documentation
Design

프로젝트를 통해 JSON DATA를 불러와서 시각화 하는 과정을 배우게 되었습니다. 여기에 필요한 도구로 Chart.js를 새로 접한 것도 소득이었습니다.

팀 프로젝트를 진행하다 보면 서로의 의견과 능력 차이로 삐걱대기도 하는데 아무런 불협화음 없이 순탄하게 프로젝트를 이끌어준 팀장님 그리고 최선을 다해 협력해준 팀원들에게 감사합니다.

코딩 실력이 부족해 팀 프로젝트를 앞두고 걱정이 많이 되었습니다. 다행히 팀원들의 배려로 내가 할 수 있는 과제를 부여 받았습니다. 내 부족함을 이해해주고, 필요한 부분에 도움을 주며 인내로 동행해 준 팀장님과 팀원들에게 감사의 마음을 전합니다.

사이트 바로가기

REVIEW

이 름: 전 영 진

담 당: Main publishing
Pug, Javascript, Chart.js

이번 프로젝트에서 저는 파싱 해온 공공 API 데이터를 원하는 구조로 재 파싱 하여 그 데이터를 시각화 하는 역할을 맡았습니다.

공공 API 데이터를 받아오는 과정에서 팀원들과의 협업을 통해 데이터들을 깔끔하게 정리하고 시작했기 때문에 쉽게 작업할 수 있었습니다.

또한 동기 / 비동기 작업 관련 에러, 데이터 파싱 관련 에러, select 값을 불러왔을 때 키 값에 적용이 안되는 에러, canvas가 표시되지 않는 에러 등 문제가 생길 때마다 이전 프로젝트에서 경험해 봤던 것들이라 과거의 경험을 토대로 비교적 수월하게 해결할 수 있었습니다.

의사소통을 통해 각자 잘할 수 있는 부분을 분담해서 빠르게 프로젝트를 마무리할 수 있었습니다. 이번 작업을 통해 협력, 의사소통이 프로젝트에서 얼마나 중요한지 다시 한 번 깨닫게 되었습니다.

사이트 바로가기

감사합니다.

사이트 바로가기