네이버 클론 코딩

https://www.youtube.com/playlist?list=PLcqDmjxt30Rsrct30BOyL5XYCsJC978Gn

[1-3강] html 만들어보기

vs code 실행 후, index.html 생성. 그리고 'doc' 입력 후 tab을 누르면 자동 완성이 된다.

[1-4강] shortcut icon 넣기 (브라우저 탭에 표시되는 아이콘)

naver에서 f12 누르고 shortcut icon 검색하면 숏컨 아이콘 코드가 나옴. 이를 'copy -> copy outerHTML'을 눌러 복사하고, 헤드 부분에 코드 추가

그리고 아이콘은 그 코드의 내용을 보면 '/favicon.ico'에 있다고 뜸. 네이버 주소 뒤에 쳐서 해당 아이콘을 웹 개발 폴더에 넣어주면 됨.

-> https://www.naver.com/favicon.ico (네이버 아이콘 위치)

* **주의 사항**: 크롬에서는 기본 폴더를 D 드라이브로 인식하기에 아이콘의 위치를 **'./favicon.ico'**로 변경해야 함.

```
<link rel="shortcut icon" type="image/x-icon" href="./favicon.ico">
```

[1-5강] block, inline-block, inline 차이점

사이트를 크게 4개의 구역으로 나눔. 각 구역을 'header', 'search', 'nav', 'main'이라 명명. (wrap은 전체 구역 의미. wrap-center는 이 사이트를 가운데 정렬할 것이라는 의미)

```
<body>
    <div id="wrap">
        <div id="wrap-center">
            <div id="header"></div>
            <div id="search"></div>
            <div id="nav"></div>
            <div id="main"></div>
        </div>
    </div>
</body>
```

그리고 가운데 정렬을 해주기 위해 css 활용. head 부분에 style 태그를 통해 함.

(#wrap은 id가 wrap인 곳을 가리킨다는 의미)

```
<style>
    #wrap {
        text-align: center; /*텍스트의 정렬 방향 의미*/
    }
    #wrap-center {
        width: 1280px;
        background-color: yellow;
    }
    #main {
        height: 2000px;
    }
</style>
```

main의 height를 준 것은 wrap-center의 height를 설정하기 위함.(자식의 height를 받아옴)

** block, inline-block, inline 차이점 **

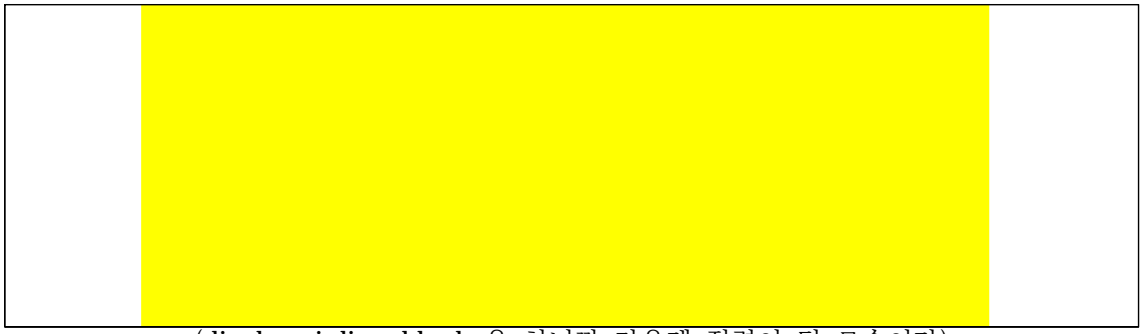
Q. 위 코드대로 웹사이트를 보면 노란색 부분이 가운데 정렬이 아니라 왼쪽으로 쏠려있다. 그 이유는?

div의 **display가 기본적으로 block**으로 되어있기 때문.

만약 가운데로 설정되게 하고 싶다면, display를 **inline-block**으로 바꾸어야 함.

* **참고**: **display: inline**으로 하면, 높이와 너비를 무시해서 아무것도 표시되지 않음.

```
<div id="wrap-center" style="
    display: inline-block;
"> == $0
```



(**display: inline-block** 을 하니까 가운데 정렬이 된 모습이다)

[1-6강] 브라우저 기본 css를 없앱시다.

헤더 부분에 햄버거 아이콘을 넣어보자.

일단 햄버거 아이콘의 크기를 f12로 알아내고, header 밑에 hamburger를 만든다.

```
<div id="header">
    <div id="hamburger">

    </div>
</div>
```

그리고 top 값을 f12로 알아내서, css 부분에 넣는다.

```
#hamburger {
    margin-top: 18px;
    background-color: blue;
    width: 46px;
    height: 46px;
}
```

* **참고**: w46을 입력하면, width: 46px;
h46을 입력하면, height: 46px;
이 자동으로 완성된다.

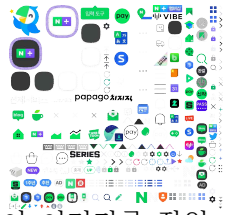
여기서 문제가 발생한다. 브라우저로 보면 body에 기본적으로 마진이 들어가 있다.

```
body {
    display: block;
    margin: 8px;
}
```

따라서 css에서 없애주어야 한다.

```
html, body {
    margin: 0;
    padding: 0;
}
```

햄버거 아이콘의 위치는 어디 있을까? f12에서 햄버거 아이콘을 눌러 '::after' 부분에서 'background-image' 부분에서 소스 이미지를 다운받을 수 있다.



이 이미지를 작업 폴더 안에 저장하자. (sp_main.png로 저장)

[1-7강] 이미지 스프라이트를 쓰는 이유, background로 위치 조절하기

**** 이미지 스프라이트를 쓰는 이유 ****

옛날 브라우저들은 이미지를 가져오는게 느렸다. 그래서 하나의 png에 여러 이미지를 모아놓은 것이 이미지 스프라이트이다.

이미지 스프라이트를 가져와야 하므로, **background-image: url(./sp_main.png);**로 가져온다.

이미지 스프라이트에서 내가 사용할 햄버거 이미지 부분을 가져오기 위해 **background-position, background-size** 값을 f12에서 가져온다.

그리고 햄버거 주변에 패딩을 넣어야하는데 바로 padding: 10px;를 하면 패딩이 의도한 대로 적용되지 않는다. 따라서 header-hamburger 아래에 div를 하나 만들어서 햄버거 이미지를 띄운다. 그리고 div의 부모인 header-hamburger에 패딩을 적용한다.

```css
#header-hamburger {
    margin-top: 18px;
    display: inline-block;
    padding: 10px;
}
#header-hamburger > div {
    background-image: url(./sp_main.png);
    width: 32px;
    height: 32px;
    background-size: 463px 442px;
    background-position: -311px -166px;
    background-repeat: no-repeat;
}
```

그리고 햄버거 아이콘을 왼쪽에 배치해야 하므로 header의 **text-align: left;** 로 설정한다.

```css
#header {
    text-align: left;
}
```

[1-8강] 패딩vs마진 결정하기, CSS에 변수가 생겼다?

햄버거 아이콘이 버튼이므로 가 아닌 <button>으로 바꾸자.

그런데 바꾸고 나니까 기본적인 설정 때문에 버튼의 배경이 흰색이 되는 문제가 발생하였다. 그래서 **background: none; border: none;** 으로 바꾸어 주었다. 그리고 버튼에 커서 올렸을 때, 커서 아이콘이 변하게 하기 위해 **cursor: pointer;** 로 설정해준다.

```css
background: none;
border: none;
cursor: pointer;
```

** 패딩 vs 마진 차이 **
패딩은 버튼에 포함되어서 빈 부분도 버튼으로 인식됨.
마진은 바깥 부분을 의미하기 때문에 빈 부분은 버튼으로 인식되지 않음.

버튼에 커서 올렸을 때 음영이 지는 효과를 구현해보자. 일단 f12의 버튼 부분에서 '우클릭 -> force state -> hover'를 눌러 음영이 계속 지도록 할 수 있다. (음영 분석 가능해짐)
그래서 코드를 보면 ::before 부분의 **color: var(--color_option_bg);** 이라 되어 있다. (--color_option_bg는 css의 변수다.) **--color_option_bg**를 사용하기 위해 네이버의 색상 설정을 복사해서 css 부분의 :root에다가 추가하자.

```
:root {
    --color_title: #080808;
    --color_title_rgb: 8, 8, 8;
    --color_body: #101010;
```
(...)
그리고 css에서 **#header-hamburger:hover** 를 추가하고 **background-color: var(--color_option_bg); border-radius: 50%;** 로 설정한다.

```
#header-hamburger:hover {
        background-color: var(--color_option_bg);
        border-radius: 50%;
}
```

[1-9강] 각종 position의 차이 알기(+ ::before, ::after)
기본값은 **position: static;** 이다.
position: relative; 는 left:*px 또는 top:*px 등의 옵션을 통해 위치를 조금씩 옮길 때 사용할 수 있다.
position: absolute; 는 기본적으로는 static의 위치에 있다. left:0, bottom:0 와 같은 옵션을 사용하여 원래 위치에서 벗어날 수 있다. 만약 자신의 부모 혹은 조상 중에 position이 static이 아닌 것이 존재 한다면, **position: absolute;** 는 그 조상을 기준으로 움직인다.
position: fixed; 는 스크롤을 해도 항상 원래 위치를 유지한다.
position: sticky; 는 스크롤 초반에는 원래 위치에 있다가, 스크롤을 계속 내리면 fixed처럼 동작함.

::before 와 **::after**는 css에 의해 삽입되는 가상 요소이다. HTML의 태그는 각각 ::before 와 ::after를 하나씩 가진다.

[1-10강] z-index와 display: none을 쓰면 안 될 때(웹접근성)

z-index는 형제끼리만 사용 가능.

z-index는 position: static에서는 사용 불가. 그래서 버튼의 z-index를 적용하기 위해 position::relative로 설정.

```
#header-hamburger:hover::before {
    position: absolute;
    content: '';
    z-index: 0;
    left: 1px;
    top: 1px;
    width: 50px;
    height: 50px;
    background-color: var(--color_option_bg);
    border-radius: 50%;
}
#header-hamburger > div {
    background-image: url(./sp_main.png);
    width: 32px;
    height: 32px;
    position: relative;
    z-index: 1;
```

네이버 f12를 보면 **확장 영역** 코드가 있다. (시각 장애인을 위함)

그런데 일반 사람들에게 시각적으로 안 보이는 구역이라고 css 영역에서 .blind 부분에 **display: none;**을 해버리면 시각 장애인용 리더기가 해당 글씨를 안 읽어준다. 그래서 별도의 코드를 작성해준다.

```
.blind {
    position: absolute;
    clip: rect(0 0 0 0);
    width: 1px;
    height: 1px;
    margin: -1px;
    overflow: hidden;
}
```

[1-11강] 길어진 css 코드 정리하는 법

우선 모든 개체의 box-sizing: border-box; 로 설정하여 width와 height의 값이 패딩을 포함한 전체의 크기가 되게끔 설정한다. 그러면 앞으로는 padding 값만 빼주면 해당 아이콘의 크기가 되기에 좀 더 편하다. (일일이 패딩 계산 안 해도 됨)

```
* {
    box-sizing: border-box;
}
```

코드를 보면 겹치는 부분이 많다는 것을 알 수 있다. 이 겹치는 부분들은 하나로 모아서 작성하고, 차이가 나는 부분만 따로 작성해주면 된다.

```
#header-hamburger, #header-naverpay, #header-notice {
    position: absolute;
    top: 5px;
    display: inline-block;
    padding: 8px;
    background: none;
    border: none;
    cursor: pointer;
}
```

```css
#header-hamburger {
    left: -12px;
}
```

[1-12강] 검색창 만들기 시작!

검색창을 만들기 위해 보통 **<input>**을 사용하는데 **<form>**으로 감싸주는 것이 좋다.
그리고 <input> 위에 **<label>**을 추가하여 input에 대한 설명을 해준다.
검색 버튼을 검색 창 안에 넣어야하므로 버튼도 하나 만들어 준다.

css를 작성해야 하는데 스크롤을 너무 많이 해야하므로 임시로 body 안에 <style>을 삽입하도록 하겠다.

**** body 안에 <style>을 삽입하는 것이 비추천되는 이유 ****

<style> 태그가 <body> 안에 있으면, 브라우저가 페이지 콘텐츠를 읽다가 중간에 스타일 정보를 발견하고 다시 파싱해야 하므로 성능이 떨어질 수 있다.

검색 구역 왼쪽에 네이버 이미지가 있으므로 삽입해주어야 한다. 다만, 네이버는 네이버 이미지를 <svg>(벡터 이미지, 많이 복잡함)로 사용하고 있기에 복사해서 가져오도록 하겠다.
그리고 <a> 태그 안에 넣어서 아이콘의 패딩 공간을 확보하도록 하겠다. (참고로 <a> 태그는 링크를 연결하는 역할을 한다.)

```html
<div id="search">
    <!-- <form>의 가장 중요한 역할은 사용자가 입력한 데이터를 서버로 전송하는 것 -->
    <form action="">
        <a href="#">
            <svg viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg"><path
        </a>
        <!-- 폼 컨트롤(예: <input>, <textarea>)에 대한 **레이블(설명)**을 정의하는 태그 -->
        <label for="search-input" class="blind">검색어 입력</label>
        <input id="search-input" type="text" />

        <button class="blind">검색</button>
    </form>
</div>
<style>
    #search > form {
        display: inline-block;
        width: 708px;
        height: 60px;
        border: 1px solid #03c75a;
        border-radius: 33px;
        text-align: left;
    }
    #search > form > input {
        width: 480px;
        height: 100%;
        padding: 17px 0;
        border: none;
        outline: none;
    }
    #search > form a {
        width: 67px;
        height: 100%;
        padding: 17px;
    }
    #search > form svg {
        height: 24px;
        width: 24px;
    }
</style>
```

[2-1강] inline-block의 문제점과 vertical-align에 대한 오해

inline block은 자신의 왼쪽에 있는 개체에 영향을 받는다.
vertical-align은 자신의 왼쪽 개체의 세로를 기준으로 정렬되는 것이다.

[2-2강] Flex로 간편하게 배치하기(justify-content, align-items)

inline-block으로 했을 때 삐뚤빼뚤하던 문제가 **#search > form** 의 **display: inline-flex;**로 바꾸어주니 일단 가로로는 모두 반듯하게 정렬되었다.

```
#search > form {
    display: inline-flex;
```

부모가 **display: flex 또는 inline-flex**일 경우 css 옵션으로 **flex:1;**을 사용할 수 있다. flex: 1;을 사용하면 나머지 부분의 width는 보장하면서 나머지 부분을 모두 차지한다.
(만약 **형제끼리 flex:2; / flex:1;** 옵션을 주면 **2:1 비율**로 남은 공간을 차지한다.)

display: flex; 옵션을 주면, 크롬 f12 창에서 flex 옆에 있는 버튼을 눌러 **justify-content, a lign-items**를 조정할 수 있다.

```
#search-right {
    flex: 1;
    display: flex;
    justify-content: flex-end;
    align-items: stretch;
}
```

[2-3강] AI한테 도움받기, css 파일 들여다보기

#search-button > div의 **background:#03c75a**(naver 색깔)로 설정한다.
#search-keyboard:hover::before 의 **filter: brightness(0.7);**로 설정하여, 키보드 아이콘에 마우스를 올리면 색이 조금 더 진해지도록 해준다.
#search-input::placeholder의 **color: white;** 로 설정하고,
#search-input:focus::placeholder의 **color: #e4e4e4;**로 설정하여
검색창을 클릭했을 때에만 검색어를 입력하라는 문구가 보이게 설정한다.

[2-4강] 키워드의 중요성(:focus-within)

#search-svg::after의 **display: none;**으로 설정하고,
#search > form:focus-within #search-svg::after에서 회색 선을 하나 띄우도록 해서, inp ut 창에 focus가 되었을 때 svg 오른쪽 부분에 회색 선을 하나 띄우도록 하였다.
이 선을 자유롭게 움직이도록 하기 위해서 **#search-svg**의 **position: relative;**으로 설정하고,
#search > form:focus-within #search-svg::after의 **position: absolute;**로 설정하였다.
(position: absolute;는 부모/조상 중에 position: static이 아닌 곳을 기준으로 정렬되므로)

```
#search-svg {
    position: relative;
```

```
#search-svg::after {
    display: none;
}
#search > form:focus-within #search-svg::after {
    content: "";
    width: 1px;
    height: 20px;
    background-color: #e4e4e4;
    position: absolute;
```

[2-5강] 리스트를 활용해서 메뉴 만들기(nth-child, nth-of-type)

이제 네비게이션 메뉴들을 리스트로 만들어보자.
메뉴들이 숫자처럼 순서가 있는 것이 아니므로 **<ul>** 태그를 사용하겠다.(unordered list)
리스트 내부의 메뉴들은 클릭하면 어디론가 이동되므로 **<a>** 태그를 사용하겠다.

리스트에서 id 없이 지정하는 방법이 nth-child와 nth-of-type이 있다.
nth-child는 리스트에서 같은 태그가 끼여있을 때, 도 n번째 자식에 포함시킨다.
nth-of-type은 리스트에서 외의 태그들은 무시한다.

```
#nav li:nth-of-type(1) {
    margin-left: 0;
}
#nav li:nth-of-type(2) {
```

#nav > ul에서 display: flex;로 설정 후, **justify-content: center;**를 통해 가운데 정렬시킨다. **list-style-type: none;**을 통해 앞쪽에 붙은 점을 제거한다.

```
#nav > ul {
    display: flex;
    list-style-type: none;
    justify-content: center;
    font-size: 1.4rem;
    line-height: 20px;
}
```

#nav a에서 **text-decoration: none;** 를 하여 글자 밑의 선을 제거한다.
#nav a:visited에서 **color: black;**으로 설정하여 글자 색을 검정으로 한다.

```
#nav a {
    text-decoration: none;
}
#nav a:visited {
    color: black;
}
```

[2-6강] 내비게이션 메뉴 완성하기

아이콘을 넣을 div의 **display: block;**으로 기본 설정 되어 있어서 **text-align: center;가 인식되지 않는 문제**가 생긴다.
#nav a div::after에다가 아이콘 설정을 해주고, **#nav a div::before**에다가 아이콘 테두리 아이콘을 넣어준다.
여기서 before의 테두리가 각 아이콘 위에 오는 문제가 발생하였는데, 이를 해결하기 위해 **z-index**를 사용한다. **after**의 **position: relative;**로 설정하고 **z-index: 1;**로 설정한다.
(z-index는 position: static일 경우에는 z-index 설정이 안 되므로)

```
#nav a div::after {
    position: relative;
    z-index: 1;
```

[3-1강] 시맨틱 태그와 괴상한 CSS 선택자(^=, $=)

모든 구역을 로만 나타내면 나중에 헷갈릴 수 있다. 따라서 **<nav>**, **<main>**과 같은 시맨틱 태그를 사용하도록 하겠다. **<aside>** 태그를 이용해 광고를 표시하고,
주요 내용들은 **<section>** 태그를 통해 넣도록 하겠다.

```
<main id="main">


</main>
```

"^="를 사용하면 뒤에 오는 문자로 **시작하는** 모든 개체를 의미하고,
"$="을 사용하면 뒤에 오는 문자로 **끝나는** 모든 개체를 의미한다.

```
section[id^=main] {
    box-shadow: 0 0 0 1p
    border-radius: 8px;
}
```

[3-2강] 로그인 섹션 만들기

#main-login의 **display: flex;**로 설정 후, **flex-flow: column;**으로 하여 내부의 아이콘들이 세로로 정렬되도록 하였다.

[3-3강] 저는 HTML을 먼저 짝 써놓고 CSS를 나중에 입힙니다.

뉴스 부분을 **<header>**, ** **, **<footer>**로 나누었다. (시멘틱 태그 이용)

[3-4강] 드디어 Grid! (복잡한 그리드는 다다음 시간에)

언론사들이 격자로 정렬되어 있으므로 **display: grid;** 를 사용해보자.
display: grid; 후에 행과 열의 너비를 **grid-template-rows**, **grid-template-column**에 기재해준다. "1fr" 을 써주면 너비를 각각 동일하게 분배한다. **repeat(n, 1fr)** 을 사용해도 된다.

```
#main-newstand-grid {
    display: grid;
    grid-template-rows: 56px 56px 56px 56px;
    grid-template-columns: 131.27px 131.27px 131.27px 131.27px 131.27px 131.27px;
```

```
height: 224px;
grid-template-rows: 1fr 1fr 1fr 1fr;
width: 790px;
grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr;
```

==

```
height: 224px;
grid-template-rows: repeat(4, 1fr);
width: 790px;
grid-template-columns: repeat(6, 1fr);
```

그리고 내부 아이템을 가운데로 정렬하고 싶으므로 **justify-content**, **align-items: center**로 설정한다.

```
justify-content: center;
align-items: center;
```

그리드의 내부에도 선을 주고 싶을 때에는 그리드의 백그라운드 색을 회색으로 하고, 각 div를 흰색으로 설정하면 된다.

```
#main-newstand-grid {
    display: grid;
    grid-gap: 1px;
    background-color: var(--color_border_in);
#main-newstand-grid > div {
    height: 100%;
    background-color: white;
```

[3-5강] 텍스트가 길 때 ...으로 줄이는 기술(ellipsis)

뉴스 제목이 너무 길 때, "..."으로 줄여보도록 하자.

우선 제목이 여러 줄이 되지 않도록 **white-space: nowrap;**을 해준다.

그리고 **text-overflow: ellipsis;** 와 **overflow: hidden;**을 통해 기사 제목이 너무 길면 ...으로 나타낼 수 있다.

```
#main-newstand-animation a:last-of-type {
    width: 450px;
    text-overflow: ellipsis;
    white-space: nowrap;
    display: inline-block;
    overflow: hidden;
}
```

[3-6강] 가운데 정렬할 때 방해되는 요소는 absolute로

리스트 보기와 격자 보기 아이콘은 별개로 빼서, "언론사 더보기" 부분이 가운데로 오도록 하겠다.

리스트 보기와 격자 보기 부분의 **display: absolute;**로 하고, **부모** 부분인 footer의 **display: relative;**로 하여서 두 아이콘은 별개로 두었다.

```
#main-newstand footer .list , #main-newstand footer .grid {
    position: absolute;
    border: none;
    cursor: pointer;
    top: 0;
    background-color: transparent;
}
```

글씨를 여러 칸을 띄우고 싶을 때에는 ** **를 활용한다.

[3-7강] 중복 제거할 때 CSS 우선순위 생각하기

css 작성 시, 기본적으로는 아래에 작성된 코드가 우선 적용된다.

하지만 **id를 이용해서 작성한 코드**는 class를 이용해 작성한 코드보다 **우선순위가 높다**.

```
.left-section footer .text {
    padding: 0 16px;
    min-width: 156px;
}
```

우선순위 <

```
#main-shopping footer .text {
    min-width: 176px;
}
```

[3-8강] 딱 떨어지지 않는 grid 맞추기

display: grid; 에서 **grid-auto-flow: column;** 으로 설정하면, 세로 순으로 배치된다.

```
#main-shopping-grid {
    display: grid;
    grid-auto-flow: column;
```

만약 몇몇 요소가 칸을 2칸씩 차지하게 하고 싶다면, **grid-row: 1 / 3;**를 하면 된다.

(1칸<= item <3칸) 또는 **grid-row: 1 / span 2;**도 동일하게 동작한다.

[3-9강] 수업의 끝
그리드 부분 완성

[3-10강] animation과 @keyframes로 js 없이 css만으로 애니메이션 하기