JCC 코드 가이드 라인!

공통(HTML/CSS)

- 각 구간 안에는 div로 구분하고 클래스명을 부여할 때 해당 구간에 '앞글자(소문자)'를 따고 '_'를 사용해 뒤에 '의미있는 이름'을 붙여줌 (ex: header 안에 menu =>h_menu))

```
<section class="main">
    <main>
        <div class="m_content">
            <!-- 메인 콘텐츠 내용 -->
        </div>
    </main>
</section>

<section class="footer">
    <footer>
        <div class="f_info">
            <!-- 푸터 정보 내용 -->
        </div>
    </footer>
</section
```

- 작업 중간 중간 W3C유효성 검사 필수.
- 주석으로 알아보기 쉽게 구간 표시

HTML

1. 구간**:**

- 구간 태그 header는 <header>,<main>,<footer>로 구분
- main을 구분할 때는 해당 페이지의 이름을 Class명으로 부여 (ex: works->works/contact->contact)
- css, js 작업은 외부 링크로만 작성
- 외부링크는 <title> 아래로 태그, og, favicon, font, reset, style 순으로 넣음

2. **들여쓰기:**

- 탭 2칸 (스페이스 사용x)

3. **파비콘 링크 추가하기:** 웹페이지의 `<head>` 섹션에 아래 코드를 추가하여 파비콘을 설정합니다. 예시는 파비콘 파일이 "favicon.ico"로 저장되어 있는 경우입니다:

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

이 코드는 파비콘 파일의 경로를 지정하고, 파비콘이 지원되지 않는 브라우저를 위해 타입을 지정합니다.

4. **유효성 검사**

- 작업 중간 중간 w3로 유효성 검사 필수

CSS

1. **선택자**

- 선택자는 클래스명을 기본으로
- 하위 선택자를 기본으로 작성하고 특수한 경우에 그 외 선택자를 사용함

  <header>

  </header>

  css = > .btn {스타일} / 특수한 경우 (btn이 겹칠때) h_container .btn {스타일}

2. **스타일 시트 구조:**

- 스타일 순서는 일관성 있게 header → main → footer 작업

3. **효율적인 스타일 구조**

- 이미지 단위 : px 고정 필요에따라 %사용
- font-family: Pretendard 사용
- color : 포인트컬러: #EA5028 사용, 백그라운드: black.white 사용

4. **미디어 쿼리**

- 모바일 기준으로 작업 후 pc 작업

JavaScript

1. **변수명:**

- 변수 선언은 지역변수로 작성을 기본
- 변수 이름은 명확한 이름을 사용

```
javascriptCopy code
// 예: 명확한 변수명과 한글 주석
const btn = document.querySelector('.h-btn');
// 버튼 엘리먼트를 선택
const inp = document.querySelector('.h-inp');
// 입력 엘리먼트를 선택

javascriptCopy code
// 예: 코드가 겹칠때
const mBtn = document.querySelector('.m-btn');
main > btn = mbtn = 명확한 표기를 위해 카멜표기법 mBtn
// 버튼 엘리먼트를 선택
const mInp = document.querySelector('.m-inp');
// 입력 엘리먼트를 선택
```

2. **event 변수**

이벤트 처리 및 addEventListener 사용:

- 코드에서 이벤트 처리는 `addEventListener` 함수를 통해 수행됩니다. 이 방법은 JavaScript 코드와 HTML을 명확하게 분리하여 코드의 구조를 개선하고 관리성을 향상시킵니다.

이벤트 객체와 event 변수 활용:

- `addEventListener` 함수를 활용할 때, 이벤트 핸들러 함수 내부에서 `event` 변수를 사용하여 이벤트 객체에 접근합니다. 이 객체에는 이벤트와 관련된 다양한 정보와 속성이 포함되어 있어 코드 내부에서 이 정보를 활용할 수 있습니다.

"onclick" 속성 사용의 제한:

- HTML 요소에 "onclick" 속성을 직접 사용하여 이벤트 핸들러를 할당하는 것은 지양하며, 대신 `addEventListener` 함수를 활용하여 이벤트 처리를 구현합니다. 이로써 코드의 유지 보수성을 향상시키고 가독성을 높입니다.