JCC 코드 가이드 라인!

HTML

1. **시맨틱 HTML 사용:**

- 구간 태그를 사용하여 문서 구조를 나타냅니다. (예: **<header>**, **<nav>**, **<section>** 등)
- <section>태그로는 header,main,footer 구분.

  예:)<section class=”header>

  <header><header>

  </section>
- <header>,<main>,<footer>안에 태그 구분

  예) <header>

  내용

  </header>
- class명 가독성을 위해 해당 구간 앞자리를 사용(

  예: header의 메뉴부분은 h_menu, main의 메뉴부분은 m_menu, footer의 메뉴 부분은 f_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
```

- <main>구간은 해당 페이지 이름으로 정의.(예: inside,works…)

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.

CSS

1. **의미 있는 클래스와 ID 이름 사용:**

- 클래스 및 ID 이름은 의미가 있어야 하며 일관성을 유지합니다.

  예) <header>

  <img>

  </header>

  띄어쓰기는 언더바로 구분. header .h_img { style 값 }

2. **스타일 시트 구조:**

- 스타일 순서는 일관성 있게 header → main → footer 작업
- <header>,<footer> 스타일 통일성을 위해 링크css로 설정

3. **효율적인 스타일 구조**

- 이미지 단위 : px 고정 필요에따라 %사용
- font-family: Pretendard 사용

- color : 포인트컬러: #EA5028 사용, 백그라운드: black.white 사용

4. **링크 css**

- reset.css , font.css 적용

5. **주석**

- 구간 확인을 위해 사용 (예:/*header*/ …~~~~~)

JavaScript

1. **의미 있는 변수명:**

- 변수 이름은 명확하고 의미 있는 이름을 사용합니다.

```
javascriptCopy code
// 예: 의미 있는 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.

1. **주석:**

- 코드에 주석을 추가하여 중요한 부분이나 복잡한 로직을 설명합니다.