HTML, CSS, JavaScript 상세 분석

HTML 분석 상세 설명

HTML은 웹 페이지의 뼈대 역할을 하며, 여기서는 장바구니 목록을 구현하는 기본 구조를 설정했습니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>장바구니</title>
    <script src="https://kit.fontawesome.com/8347ef8aff.js"
crossorigin="anonymous"></script>
    <link rel="stylesheet" href="./shop1.css">
    <script src="./shop1.js" defer></script>
</head>
<body>
    <section class="list">
        <header class="header">Shopping List</header>
        <ul>
            <li>물건1 <i class="fa-solid fa-trash-can delete"></i></li>
            <li>물건2 <i class="fa-solid fa-trash-can delete"></i></li>
            <li>물건3 <i class="fa-solid fa-trash-can delete"></i></li>
            <li>물건4 <i class="fa-solid fa-trash-can delete"></i></li>
            <li>물건5 <i class="fa-solid fa-trash-can delete"></i></li>
        </ul>
        <footer class="footer">
            <input type="text" class="footer_input">
            <button class="footer_plus">
                <i class="fa-solid fa-plus"></i>
            </button>
        </footer>
    </section>
</body>
</html>
```

구성 요소 설명

1. `<!DOCTYPE html>` **및** `<html>` **태그**
    - `<!DOCTYPE html>`은 HTML5 문서임을 명시합니다.
    - `<html lang="ko">`는 문서의 언어를 한국어로 설정합니다. 이렇게 하면 검색 엔진이나 스크린 리더 등이 문서의 언어를 올바르게 인식하게 됩니다.
2. `<head>` **태그**
    - `<meta charset="UTF-8">`는 문서의 인코딩을 UTF-8로 설정하여 한글을 포함한 모든 언어를 제대로 표시할 수 있도록 합니다.

- `<meta name="viewport"...>`는 모바일 환경에서도 페이지가 적절히 보이도록 화면 크기를 제어합니다.
- `<title>장바구니</title>`는 브라우저 탭에 표시될 제목을 설정합니다.
- `<script src="https://kit.fontawesome.com/8347ef8aff.js" crossorigin="anonymous"></script>`는 Font Awesome을 불러와 아이콘을 사용할 수 있게 합니다. 장바구니에 들어갈 휴지통 아이콘과 추가 버튼의 `+` 아이콘을 사용하는데 이 라이브러리를 사용합니다.
- `<link rel="stylesheet" href="./shop1.css">`는 CSS 파일을 연결하여 스타일을 정의합니다.
- `<script src="./shop1.js" defer></script>`는 JavaScript 파일을 연결하여 동적인 기능을 추가합니다. `defer` 속성은 HTML이 모두 로드된 후 스크립트를 실행하도록 합니다.

3. `<body>` **태그**

- `<section class="list">`: 전체 장바구니를 감싸는 컨테이너입니다. 전체 쇼핑 목록의 레이아웃을 구성합니다.
- `<header class="header">Shopping List</header>`: 페이지의 제목을 나타냅니다. 사용자에게 이 섹션이 쇼핑 리스트임을 알려줍니다.
- `<ul>`**와** `<li>`:
  - `<ul>` 태그는 목록의 컨테이너 역할을 하며 `<li>` 태그는 각각의 아이템을 나타냅니다.
  - `<li>` 내부에는 물건 이름과 휴지통 아이콘이 있습니다.
  - `<i class="fa-solid fa-trash-can delete"></i>` 는 삭제 버튼으로, Font Awesome에서 제공하는 휴지통 모양의 아이콘을 사용합니다. `delete` 클래스는 JavaScript에서 클릭 이벤트를 처리할 때 사용됩니다.
- `<footer class="footer">`:
  - 이 섹션은 사용자가 새로운 물건을 추가할 수 있는 부분입니다.
  - `<input type="text" class="footer_input">` 는 사용자가 물건 이름을 입력할 수 있는 텍스트 입력창입니다.
  - `<button class="footer_plus">` 는 항목을 추가하는 버튼입니다. 버튼 안에 `<i class="fa-solid fa-plus"></i>` 를 사용하여 추가할 수 있는 `+` 아이콘을 표시합니다.

CSS 분석 상세 설명

CSS 파일은 HTML 파일에 디자인을 추가하여 웹페이지를 더 보기 좋게 만듭니다. 여기서는 레이아웃, 색상, 정렬 등을 설정합니다.

```css
body {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Arial', sans-serif;
    display: flex;
    justify-content: center;
```

```
    align-items: center;
    height: 100vh;
    background: #e9e9e9;
}

.list {
    background: linear-gradient(to right, #2b0f59, #a94ca5, #00c8ff);
    border-radius: 20px;
    width: 400px;
    overflow: hidden;
    box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
    display: flex;
    flex-direction: column;
    height: 650px;
}

.header {
    padding: 20px;
    text-align: center;
    font-size: 24px;
    color: white;
    background: linear-gradient(to right, #2b0f59, #a94ca5, #00c8ff);
    border-radius: 20px 20px 0 0;
}

ul {
    list-style: none;
    padding: 0;
    margin: 0;
    background: #f7f6f9;
    flex: 1;
    overflow-y: auto;
    scroll-behavior: smooth;
}

li {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px;
    border-bottom: 1px solid #e5e5e5;
    font-size: 16px;
}

li:last-child {
    border-bottom: none;
}

li .delete {
    color: #444;
    cursor: pointer;
    font-size: 16px;
    transition: color 300ms ease-in;
}
```

```
li .delete:hover {
    color: #f00;
    transform: scale(1.1);
}

.footer {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 20px;
    background: linear-gradient(to right, #2b0f59, #a94ca5, #00c8ff);
    border-radius: 0 0 20px 20px;
}

.footer_input {
    flex: 1;
    height: 40px;
    padding: 0 15px;
    font-size: 16px;
    border: none;
    border-radius: 20px;
    outline: none;
    margin-right: 10px;
}

.footer_plus {
    background-color: #000;
    color: #fff;
    border: none;
    width: 50px;
    height: 50px;
    border-radius: 50%;
    cursor: pointer;
    display: flex;
    justify-content: center;
    align-items: center;
}

.footer_plus:hover {
    transform: scale(1.1);
}
```

주요 스타일 설명

1. `body` **스타일링**
    - `margin: 0;`, `padding: 0;`
      : 브라우저 기본 여백 제거.
    - `box-sizing: border-box;`
      : 패딩과 테두리를 포함해 크기를 계산하도록 설정하여 레이아웃 관리 용이.

- `display: flex;`
  , `justify-content: center;`
  , `align-items: center;`
  : 모든 콘텐츠가 화면의 정중앙에 오도록 배치.

2. `.list` **스타일링**

- `background: linear-gradient(to right, #2b0f59, #a94ca5, #00c8ff);`
  : 오른쪽으로 색상이 섞이는 그라디언트 배경 적용.
- `border-radius: 20px;`
  : 모서리를 둥글게 하여 부드러운 느낌.
- `overflow: hidden;`
  : 내부 콘텐츠가 넘칠 경우 잘리도록 설정.
- `box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);`
  : 그림자를 추가해 입체감을 줌.

3. `.header` **스타일링**

- `padding: 20px;`
  : 요소 안쪽에 여백 추가.
- `text-align: center;`
  : 텍스트 가운데 정렬.
- `color: white;`
  : 텍스트 색상 설정.

4. `ul` **(리스트 항목을 담는 컨테이너)**

- `list-style: none;`
  : 기본 불릿 포인트 제거.
- `overflow-y: auto;`
  : 세로 스크롤 필요할 때만 표시.
- `scroll-behavior: smooth;`
  : 스크롤 동작을 부드럽게 설정.

5. `li` **(각각의 쇼핑 항목)**

- `display: flex;`
  , `justify-content: space-between;`
  : 항목을 좌우로 분배하여 배치.
- `border-bottom: 1px solid #e5e5e5;`
  : 각 항목 아래 구분선을 추가.

6. `.footer` **(입력창과 버튼을 감싸는 부분)**

- `display: flex;`
  , `justify-content: center;`
  , `align-items: center;`
  : 입력창과 버튼을 중앙에 배치.
- `background`
  : 그라디언트 배경 적용.

7. `.footer_input` **(입력창)**
    - `flex: 1;`
      : 입력창이 가용할 수 있는 최대 공간을 차지하도록 설정.
    - `border-radius: 20px;`
      : 모서리를 둥글게 하여 부드러운 느낌.

8. `.footer_plus` **(추가 버튼)**
    - `border-radius: 50%;`
      : 버튼을 원형으로 디자인.
    - `cursor: pointer;`
      : 마우스를 올렸을 때 포인터 모양으로 바뀌게 설정.
    - `transform: scale(1.1);` (hover 상태)
      : 마우스를 올렸을 때 크기가 1.1배 확대하여 시각적 반응 제공.

JavaScript 분석 상세 설명

JavaScript는 웹 페이지에 동적인 기능을 추가합니다. 여기에서는 쇼핑 목록에 항목을 추가하고 삭제하는 기능을 구현했습니다.

```
document.addEventListener("DOMContentLoaded", () => {
    const inputField = document.querySelector(".footer_input");
    const addButton = document.querySelector(".footer_plus");
    const itemList = document.querySelector("ul");

    // 항목 추가 함수
    function addItem() {
        const itemText = inputField.value.trim(); // 입력값에서 앞뒤 공백을 제거
        if (itemText !== "") {
            const newItem = document.createElement("li"); // 새로운 <li> 요소 생성
            newItem.innerHTML = `${itemText} <i class="fa-solid fa-trash-can
delete"></i>`; // 리스트 아이템과 삭제 아이콘 추가
            itemList.appendChild(newItem); // <ul>에 새 항목 추가
            inputField.value = ""; // 입력창 비우기
            inputField.focus(); // 입력창에 다시 포커스
            scrollToBottom(); // 스크롤을 가장 아래로 이동
        }
    }

    // 삭제 버튼 클릭 시 항목 삭제
    itemList.addEventListener("click", (event) => {
        if (event.target.classList.contains("delete")) {
            const listItem = event.target.parentElement; // 클릭한 아이콘의 부모
<li>를 찾음
            itemList.removeChild(listItem); // <li>를 삭제
        }
    });

    // 추가 버튼 클릭 이벤트
    addButton.addEventListener("click", () => {
```

```
        addItem(); // 추가 버튼 클릭 시 항목 추가
    });

    // 엔터키로 항목 추가
    inputField.addEventListener("keypress", (event) => {
        if (event.key === "Enter") {
            addItem(); // 엔터키 누르면 항목 추가
        }
    });

    // 스크롤 하단 고정 함수
    function scrollToBottom() {
        itemList.scrollTop = itemList.scrollHeight; // 스크롤을 가장 아래로 설정
    }
});
```

구성 요소 설명

1. `document.addEventListener("DOMContentLoaded", () => {...})`:
    - 이 코드는 HTML이 모두 로드된 후 JavaScript를 실행하도록 설정합니다. 이렇게 해야 JavaScript가 HTML 요소를 찾지 못하는 상황을 방지할 수 있습니다.
2. **변수 정의**:
    - `const inputField = document.querySelector(".footer_input");`
      : 입력창 요소를 선택하여 변수 `inputField`에 저장합니다.
    - `const addButton = document.querySelector(".footer_plus");`
      : 추가 버튼을 선택하여 변수 `addButton`에 저장합니다.
    - `const itemList = document.querySelector("ul");`
      : 장바구니 목록을 담고 있는 `<ul>` 요소를 선택하여 변수 `itemList`에 저장합니다.
3. `addItem()` **함수**:
    - 이 함수는 사용자가 새로운 항목을 추가할 때 호출됩니다.
    - `const itemText = inputField.value.trim();`
      : 입력된 텍스트의 앞뒤 공백을 제거합니다.
    - `if (itemText !== "") { ... }`
      : 입력된 내용이 빈 문자열이 아닌 경우에만 항목을 추가합니다.
    - `const newItem = document.createElement("li");`
      : 새로운 `<li>` 요소를 생성합니다.
    - `newItem.innerHTML = ...`
      : 새로 만든 항목에 입력된 텍스트와 휴지통 아이콘을 추가합니다.
    - `itemList.appendChild(newItem);`
      : 새로운 항목을 목록의 끝에 추가합니다.

- `inputField.value = "";`
  : 항목을 추가한 후 입력창을 비웁니다.

- `inputField.focus();`
  : 다시 입력창에 포커스를 두어 사용자가 바로 다음 항목을 입력할 수 있게 합니다.

- `scrollToBottom();`
  : 항목을 추가한 후 스크롤을 자동으로 아래로 이동시켜 새 항목이 보이도록 합니다.

4. **항목 삭제 기능**:

- `itemList.addEventListener("click", (event) => { ... })`
  : 사용자가 목록에 있는 항목의 삭제 버튼을 클릭할 때 작동하는 이벤트입니다.

- `if (event.target.classList.contains("delete"))`
  : 클릭한 요소가 삭제 버튼인지 확인합니다.

- `const listItem = event.target.parentElement;`
  : 클릭한 삭제 버튼의 부모 요소인 `<li>`를 찾습니다.

- `itemList.removeChild(listItem);`
  : 해당 `<li>`를 목록에서 제거하여 항목을 삭제합니다.

5. **추가 버튼 클릭 이벤트**:

- `addButton.addEventListener("click", () => { addItem(); });`
  : 사용자가 추가 버튼을 클릭하면 `addItem()` 함수가 호출되어 새로운 항목이 추가됩니다.

6. **엔터키로 항목 추가**:

- `inputField.addEventListener("keypress", (event) => { ... })`
  : 사용자가 입력창에 포커스를 두고 엔터키를 누르면 항목이 추가됩니다.

- `if (event.key === "Enter")`
  : 누른 키가 엔터키인지 확인한 후, 맞다면 `addItem()` 함수를 호출합니다.

7. **스크롤 하단 고정 함수 (`scrollToBottom`)**

- `function scrollToBottom() { ... }`
- `itemList.scrollTop = itemList.scrollHeight;`
  : 목록의 스크롤 위치를 항목의 전체 높이로 설정하여 스크롤이 자동으로 하단으로 이동하게 합니다. 이로써 새로운 항목이 추가될 때마다 사용자가 항상 마지막 항목을 볼 수 있도록 합니다.