# Code Explanation: To-Do List - JavaScript Elements

## JavaScript Code

<script>  
 function addTask() {  
 const taskInput = document.getElementById('task');  
 const taskText = taskInput.value.trim();  
 if (taskText === '') return;  
  
 const task = document.createElement('li');  
 task.textContent = taskText;  
 task.addEventListener('click', () => task.classList.toggle('done'));  
 document.getElementById('tasks').appendChild(task);  
 taskInput.value = '';  
 }  
</script>

## 1. `<script>` Tag

- \*\*`<script>`\*\*: Defines a section for JavaScript code within the HTML document.  
- Placed before the closing `</body>` tag to ensure the DOM is fully loaded before the script runs.

## 2. `function addTask()`

- \*\*`function`\*\*: Declares a reusable block of code.  
- \*\*`addTask`\*\*: The name of the function that handles adding a new task to the to-do list.

## 3. `document.getElementById()`

- \*\*`document.getElementById('task')`\*\*: Retrieves the HTML element with the ID `task` (the input field).  
- \*\*`document.getElementById('tasks')`\*\*: Retrieves the unordered list element where tasks are added.

## 4. `taskInput.value.trim()`

- \*\*`taskInput.value`\*\*: Accesses the current text entered in the input field.  
- \*\*`.trim()`\*\*: Removes leading and trailing whitespace from the input value to avoid adding empty tasks.

## 5. `if (taskText === ) return;`

- \*\*`if`\*\*: Conditional statement that checks if the task text is empty.  
- \*\*`return;`\*\*: Exits the function without performing any further actions if the condition is true.

## 6. `document.createElement()`

- \*\*`document.createElement('li')`\*\*: Creates a new `<li>` element to represent a task in the list.

## 7. `task.textContent = taskText;`

- \*\*`task.textContent`\*\*: Sets the text content of the `<li>` element to the value of `taskText`.

## 8. `task.addEventListener()`

- \*\*`addEventListener('click', ...)`\*\*: Attaches a click event listener to the task element.  
- \*\*`task.classList.toggle('done')`\*\*: Toggles the `done` class on the task element when clicked, adding or removing styles (e.g., line-through).

## 9. `document.getElementById('tasks').appendChild(task)`

- \*\*`appendChild()`\*\*: Adds the newly created `<li>` element as a child of the `<ul>` element with the ID `tasks`.

## 10. `taskInput.value = ;`

- Resets the input field to an empty string after a task is added, clearing the field for the next entry.

## Key Flow

1. The user enters a task in the input field and clicks the "Add" button.  
2. The `addTask` function is triggered by the `onclick` event.  
3. The function retrieves and validates the input value.  
4. A new `<li>` element is created, styled, and appended to the `<ul>`.  
5. Each task can be marked as "done" by clicking on it, toggling the `done` class.