Annotated Implementation of a Calendar and Event List Web Application

- **day.js**
- **month.js**
- **Calendar.js**
- **App.js**
- **Index.html**

day.js:

```
// Represents a single day in the calendar
function Day(weekday, date) {
   this.date = date;         // Numeric day of the month (e.g., 25)
   this.weekday = weekday;   // Name of the weekday (e.g., "Monday")
   this.tasks = [];          // Array to hold task <li> elements for this day
}

// Returns an HTML <ul> element containing all tasks for this day
Day.prototype.getHTML = function() {
   // Try to find an existing <ul> for the todo list
   let list = todoContainer.querySelector("ul");
   if (!list) {
      // If not found, create a new <ul> and set its ID
      list = document.createElement("ul");
      list.id = "todo-list";
   } else {
      // If found, clear its contents so we can re-render
      list.innerHTML = "";
   }

   // Add each task <li> element to the list
   for (let i = 0; i < this.tasks.length; i++) {
      list.appendChild(this.tasks[i]);
   }
   return list;
```

```
}

// Adds a new task to this day
Day.prototype.addTask = function(task) {
  // Create a new <li> element for this task
  const taskItem = this.createTask(task);
  this.tasks.push(taskItem);

  // Find the day cell in the calendar grid that matches this date
  const allDays = document.querySelectorAll(".day");
  for (let dayEl of allDays) {
    // Match by date and make sure it's not an unused cell
    if (parseInt(dayEl.textContent.trim()) === this.date &&
!dayEl.classList.contains("unused")) {
      // Only add a pin if one doesn't already exist
      if (!dayEl.querySelector(".event-pin")) {
        const pin = document.createElement("div");
        pin.classList.add("event-pin");
        dayEl.appendChild(pin);
      }
    }
  }
}

// Creates a new <li> element representing a task, with a remove button
Day.prototype.createTask = function(task) {
  const li = document.createElement("li");
  const taskItem = document.createElement("span");
  taskItem.textContent = task;
  const removeButton = document.createElement("button");
  removeButton.className = "remove";
  removeButton.textContent = "X";
  li.appendChild(taskItem);
  li.appendChild(removeButton);
  return li;
```

```
}

// Removes a task from this day
Day.prototype.removeTask = function(task) {
   const index = this.tasks.indexOf(task);
   this.tasks.splice(index, 1);

   // If there are no more tasks, remove the event pin from the calendar cell
   if (this.tasks.length === 0) {
      const allDays = document.querySelectorAll(".day");
      for (let dayEl of allDays) {
         if (parseInt(dayEl.textContent.trim()) === this.date) {
            const pin = dayEl.querySelector(".event-pin");
            if (pin) {
               pin.remove();
            }
         }
      }
   }
}
```

month.js

```
// Represents a month in the calendar
function Month(year, month) {
   const weekDays = [
      "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"
   ];

   const monthNames = [
      "January", "February", "March", "April", "May", "June", "July",
      "August", "September", "October", "November", "December"
   ];
```

```
    this.year = year;                 // The year (e.g., 2025)
    this.date = new Date(year, month);  // JS Date object for the first day of the
month
    this.name = monthNames[this.date.getMonth()]; // Month name (e.g., "June")
    this.days = [];                   // Array of Day objects for each day in the month

    // Calculate how many days are in this month
    this.MAXDAYS = new Date(year, month + 1, 0).getDate();

    // Figure out which weekday the month starts on (0=Monday, 6=Sunday)
    this.currentWeekDay = this.date.getDay() - 1;
    if (this.currentWeekDay === -1) {
        this.currentWeekDay = 6; // If Sunday, wrap to end
    }

    // Create a Day object for each day in the month
    for (let i = 0; i <= this.MAXDAYS - 1; i++) {
        this.days[i] = new Day(weekDays[this.currentWeekDay % 7], i + 1);
        this.currentWeekDay++;
    }
    this.currentDay = this.days[0]; // Default to the first day
}

// Renders the calendar grid for this month
Month.prototype.renderCalendar = function () {
    //  Remove all event pins from previous month
    const allPins = document.querySelectorAll('.event-pin');
    allPins.forEach(pin => pin.remove());

    //  Fade in the calendar grid for a smooth effect
    divDays.style.opacity = 0.0;
    fadeIn(divDays);

    // Calculate which cell the first day of the month should appear in
    let startDay = this.date.getDay() - 1;
```

```
if (startDay === -1) {
   startDay = 6;
}

const dayCards = document.getElementsByClassName("day");
let dayNumber = 1;

// --- Check if this is the real-life current month ---
const today = new Date();
const isCurrentMonth =
   today.getMonth() === this.date.getMonth() &&
   today.getFullYear() === this.date.getFullYear();

// --- Loop through every cell in the calendar grid ---
for (let i = 0; i < dayCards.length; i++) {
   if (i >= startDay && dayNumber <= this.MAXDAYS) {
      dayCards[i].className = "day"; // Reset class
      dayCards[i].firstChild.textContent = dayNumber; // Set date number

      // Highlight today if applicable
      if (isCurrentMonth && dayNumber === today.getDate()) {
         dayCards[i].classList.add("today");
      }

      // Add event pin if this day has tasks
      const dayObj = this.days[dayNumber - 1];
      if (dayObj && dayObj.tasks.length > 0) {
         const pin = document.createElement('div');
         pin.classList.add('event-pin');
         dayCards[i].appendChild(pin);
      }

      dayNumber++;
   } else {
         // Mark unused cells (before/after month) as empty and styled
differently
         dayCards[i].className = "day unused";
         dayCards[i].firstChild.textContent = "";
      }
   }
};
```

Calendar.js

```
// Represents the calendar for a year, with navigation and rendering logic
function Calendar(year, month) {
   this.year = year;        // Current year
   this.months = [];        // Array of Month objects for the year

   // Create Month objects for each month in the year
   while (month < 12) {
      this.months.push(new Month(year, month));
      month++;
   }

   // Set the current month and day based on today's date
   const currentDate = new Date();
   this.currentMonth = this.months[currentDate.getMonth()];
   this.currentMonth.currentDay = this.currentMonth.days[currentDate.getDate() -
1];

   // Set the headline for the todo section (e.g., "Wednesday - June 25")
   const todoHeadline = document.getElementById("todoHeadline");
   todoHeadline.textContent = this.currentMonth.currentDay.weekday + " - " +
                        this.currentMonth.name + " " +
                        this.currentMonth.currentDay.date;

   // Render the task list for the current day
   const taskList = this.currentMonth.currentDay.getHTML();
```

```
    todoContainer.appendChild(taskList);

    // Render the calendar grid
    this.renderTemplate();
}

// Move to the next month, creating a new year if needed
Calendar.prototype.nextMonth = function() {
    let index = this.months.indexOf(this.currentMonth);
    if (index >= this.months.length - 1) {
        // If at the end of the year, add next year's months
        this.months = this.createYear(this.year + 1);
    }
    this.currentMonth = this.months[index + 1];
    this.renderTemplate();
    return true;
}

// Move to the previous month, creating a new year if needed
Calendar.prototype.previousMonth = function() {
    let index = this.months.indexOf(this.currentMonth);
    if (index <= 0) {
        // If at the start of the year, add previous year's months
        this.months = this.createYear(this.year - 1);
        index = this.months.indexOf(this.currentMonth);
    }
    this.currentMonth = this.months[index - 1];
    this.renderTemplate();
    return true;
}

// Create a new set of 12 Month objects for a given year
Calendar.prototype.createYear = function(year) {
    let newMonths = [];
    let month = 0;
```

```
  while (month < 12) {
    newMonths.push(new Month(year, month));
    month++;
  }
  // Add new months to the start or end of the months array as needed
  if (this.year < year) {
    return this.months.concat(newMonths);
  } else {
    return newMonths.concat(this.months);
  }
}

// Render the calendar grid and update the month banner
Calendar.prototype.renderTemplate = function() {
  this.year = this.currentMonth.year;
  const monthName = document.getElementById("month-name");
  monthName.textContent = this.currentMonth.name + " - " +
this.currentMonth.year;
  this.currentMonth.renderCalendar();
}
```

App.js

```
// Get references to important DOM elements
const divDays = document.getElementById("days");
const todoContainer = document.getElementById("todo");
const calendar = new Calendar(2025, 0); // Start with January 2025

const addTaskButton = document.getElementById("add-task");
const todoList = document.getElementById("todo-list");
const nextMonthButton = document.getElementById("next-month");
const previousMonthButton = document.getElementById("previous-month");

// When a day is clicked in the calendar grid...
divDays.addEventListener("click", (e) => {
  // Make sure the clicked element is a valid day cell
  if (e.target.firstChild.textContent) {
    const index = parseInt(e.target.firstChild.textContent) - 1; // Get day index
    calendar.currentMonth.currentDay = calendar.currentMonth.days[index]; //
Set as current day
    const weekday = calendar.currentMonth.currentDay.weekday;
    todoHeadline.textContent = weekday + " - " +
        calendar.currentMonth.name +
        " " + (index + 1); // Update the todo headline
    const taskList = calendar.currentMonth.currentDay.getHTML(); // Get tasks
for the day
    todoContainer.appendChild(taskList); // Show them in the todo panel
  }
});

// When the "Add Task" button is clicked...
addTaskButton.addEventListener("click", () => {
  const inputField = document.getElementById("task-input");
  const task = inputField.value; // Get the text from the input
  inputField.value = "";         // Clear the input
  inputField.focus();            // Focus for convenience
  calendar.currentMonth.currentDay.addTask(task); // Add the task to the
current day
  const taskList = calendar.currentMonth.currentDay.getHTML(); // Get updated
task list
  todoContainer.appendChild(taskList); // Show updated list
});

// When a task's remove button is clicked...
todoList.addEventListener("click", (e) => {
  if(e.target.className === "remove") {
    calendar.currentMonth.currentDay.removeTask(e.target.parentNode); //
Remove from data
    todoList.removeChild(e.target.parentNode); // Remove from UI
  }
```

```
});

// When the next month button is clicked, go to the next month
nextMonthButton.addEventListener("click", function() {
   calendar.nextMonth();
});

// When the previous month button is clicked, go to the previous month
previousMonthButton.addEventListener("click", function() {
   calendar.previousMonth();
});

// Helper function to fade in an element (for smooth UI transitions)
function fadeIn(element) {
   let opacity = 0.0;
   let timer = setInterval(function() {
      if (opacity >= 0.9) {
         clearInterval(timer);
         element.style.display = "block";
      } else {
         element.style.opacity = opacity;
         element.style.filter = "alpha(opacity=" + opacity * 100 + ")";
         opacity += 0.15;
      }
   }, 100);
}
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <!-- The title that appears in the browser tab -->
```

```
  <title>Calendar</title>
  <!-- Importing Google Fonts for nicer typography -->
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600&family=Lor
a:wght@500&display=swap" rel="stylesheet">
  <!-- Link to the main CSS file for styling -->
  <link href="styles.css" rel="stylesheet" />
</head>
<body>
  <!-- Main container for the entire app -->
  <div class="container">
    <!-- Month navigation banner at the top -->
    <div id="month-banner">
      <!-- Button to go to the previous month -->
      <button id="previous-month" class="arrow">&#8592;</button>
      <!-- The name of the current month and year will be displayed here -->
      <h2 id="month-name"></h2>
      <!-- Button to go to the next month -->
      <button id="next-month" class="arrow">&#8594;</button>
    </div>

    <!-- The to-do/task section for the selected day -->
    <div id="todo">
      <!-- Headline showing the selected day (e.g., "Monday - June 25") -->
      <h3 id="todoHeadline"></h3>
      <!-- Input box for entering a new task -->
      <input type="text" placeholder="Enter a task" autofocus id="task-input"/>
      <!-- Button to add the task to the list -->
      <button id="add-task">Add Task</button>
    </div>

    <!-- The main calendar grid section -->
    <div class="calendar">
      <!-- Banner showing the days of the week (Monday to Sunday) -->
      <div class="day-banner">
    <h3>Monday</h3>
    <h3>Tuesday</h3>
    <h3>Wednesday</h3>
    <h3>Thursday</h3>
    <h3>Friday</h3>
    <h3>Saturday</h3>
    <h3>Sunday</h3>
</div>

<!-- The calendar grid: 6 weeks, 7 days per week (to fit all months) -->
<div id="days">
    <!-- Each "week" is a row in the calendar -->
    <div class="week">
        <!-- Each "day" is a cell in the row -->
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
    </div>
    <!-- Repeat for up to 6 weeks to cover all possible month layouts -->
    <div class="week">
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
    </div>
    <div class="week">
        <div class="day"><p></p></div>
        <div class="day"><p></p></div>
```

```
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
        </div>
        <div class="week">
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
        </div>
        <div class="week">
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
        </div>
        <div class="week">
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
            <div class="day"><p></p></div>
        </div>
    </div> <!-- End of days grid -->
</div> <!-- End of calendar section -->
```

```
    </div> <!-- End of main container -->

    <!-- JavaScript files that power the app logic (order matters!) -->
    <script src="day.js"></script>       <!-- Handles individual days and tasks -->
    <script src="month.js"></script>     <!-- Handles month structure and rendering -->
    <script src="calendar.js"></script>  <!-- Handles calendar navigation and logic -->
    <script src="app.js"></script>       <!-- Main app logic and event handling -->
</body>
</html>
```