**Project planning**

Computing and displaying the leveled league table – a long task involving applying Paul’s formula to every match played and adding the results for each column.

Idea :

In many cases you can move pure computational work to [Web Workers](https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/basic_usage), if, for example, it doesn’t require DOM access. Data manipulation or traversal, like sorting or searching, are often good fits for this model, as are loading and model generation.

var dataSortWorker = new Worker("sort-worker.js");  
dataSortWorker.postMesssage(dataToSort);  
  
// The main thread is now free to continue working on other things...  
  
dataSortWorker.addEventListener('message', function(evt) {  
   var sortedData = evt.data;  
   // Update data on screen...  
});

Not all work can fit this model: Web Workers do not have DOM access. Where your work must be on the main thread, consider a batching approach, where you segment the larger task into micro-tasks, each taking no longer than a few milliseconds, and run inside of requestAnimationFrame handlers across each frame.

var taskList = breakBigTaskIntoMicroTasks(monsterTaskList);  
requestAnimationFrame(processTaskList);  
  
function processTaskList(taskStartTime) {  
  var taskFinishTime;  
  
  do {  
    // Assume the next task is pushed onto a stack.  
    var nextTask = taskList.pop();  
  
    // Process nextTask.  
    processTask(nextTask);  
  
    // Go again if there’s enough time to do the next task.  
    taskFinishTime = window.performance.now();  
  } while (taskFinishTime - taskStartTime < 3);  
  
  if (taskList.length > 0)  
    requestAnimationFrame(processTaskList);  
  
}

There are UX and UI consequences to this approach, and you will need to ensure that the user knows that a task is being processed, either by [using a progress or activity indicator](https://www.google.com/design/spec/components/progress-activity.html). In any case this approach will keep your app's main thread free, helping it to stay responsive to user interactions.

<https://developers.google.com/web/fundamentals/performance/rendering/optimize-javascript-execution>