Отчёт технического интервью

Кандидат: Candidate_1

Направление: JavaScript
Количество задач: 3

1. История выполнения задач

Попытка #1

Задача: js-easy-6

Время выполнения: 42 ms

stdout: Сумма цифр 12345: 15 Сумма цифр -987: 24 Сумма цифр 0: 0

stderr:

Попытка #2

Задача: js-med-6

Время выполнения: 36 ms

stdout: 'hello world example' -> helloWorldExample ' first name' -> firstName 'CSS is awesome'
-> cssIsAwesome 'JavaScript' -> javascript

stderr:

Попытка #3

Задача: js-hard-9

Время выполнения: 35 ms

stdout: '2 + 3 * (4 - 1)' = 11 '10 / 2 + 5' = 10 '(1 + 2) * (3 + 4)' = 21 '5 + -2 * 3' = NaN
'1.5 + 2.5 * 2' = 6.5

stderr:

2. Решения кандидата

js-easy-6 — Сумма цифр числа

Напишите функцию sumDigits(num), которая возвращает сумму всех цифр числа. Учтите, что num может быть отрицательным.

```
function sumDigits(num) {
  // 1. Преобразуем число в строку
  const numStr = String(num);
  let sum = 0;
  // 2. Проходим по каждому символу строки
  for (const char of numStr) {
    // 3. Проверяем, является ли символ цифрой
    if (char >= '0' && char <= '9') {
      // 4. Преобразуем символ обратно в число и добавляем к сумме
      // Используем parseInt() или унарный плюс (+)
      sum += parseInt(char, 10);
    }
    // Если символ — это '.', ',' или '-', мы его игнорируем
  }
  return sum;
}
// Примеры использования:
```

js-med-6 — Форматирование строки в camelCase

Реализуйте функцию camelCase(str), которая преобразует строку вида 'hello world example' в 'helloWorldExample'.

```
function camelCase(str) {
 // 1. Обрезаем лишние пробелы в начале/конце и разбиваем строку по пробелам
 const words = str.trim().split(' ');
 // 2. Преобразуем каждое слово, начиная со второго
 const camelCasedWords = words.map((word, index) => {
  // Первое слово остается как есть (в нижнем регистре)
  if (index === 0) {
   return word.toLowerCase();
  }
  // Для остальных слов делаем первую букву заглавной
  // word.charAt(0).toUpperCase() + word.slice(1)
  return word.charAt(0).toUpperCase() + word.slice(1).toLowerCase();
 });
 // 3. Объединяем слова обратно в одну строку
 return camelCasedWords.join('');
}
// Примеры использования:
console.log(`'hello world example' -> ${camelCase('hello world example')}`);
// Вывод: helloWorldExample
console.log(`'  first name' -> ${camelCase('  first name')}`);
// Вывод: firstName
console.log(`'CSS is awesome' -> ${camelCase('CSS is awesome')}`);
// Вывод: cssIsAwesome
console.log(`'JavaScript' -> ${camelCase('JavaScript')}`);
// Вывод: javaScript
```

js-hard-9 — Parser mini-JS выражений

Реализуйте функцию parseExpression(expr), которая принимает строку вида '2 + 3 * (4 - 1)' и вычисляет результат. Требуется реализовать парсер с поддержкой скобок и приоритетов операций.

```
/**
 * Вычисляет математическое выражение со скобками и приоритетами операций.
 * @param {string} expr Входное выражение (например, '2 + 3 * (4 - 1)').
 * @returns {number} Результат вычисления.
 */
function parseExpression(expr) {
   // Определяем приоритет операций
   const precedence = {
      '+': 1,
      '-': 1,
      '*': 2,
      '/': 2,
   };
   const operators = []; // Стек операторов
   const output = [];    // Выходная очередь (хранит числа и операторы в ОПЗ)
   // Используем регулярное выражение для токенизации строки (числа, операторы, скобки)
   const tokens = expr.match(/(\d+\.?\d*|\+|\-|\*|\/|\(|\))/g);
   if (!tokens) {
      throw new Error("Некорректное выражение.");
   }
   for (const token of tokens) {
      if (/\d/.test(token)) {
         // Если токен — число, добавляем его в выходную очередь
         output.push(parseFloat(token));
      } else if (token === '(') {
         // Если открывающая скобка, помещаем ее в стек операторов
         operators.push(token);
      } else if (token === ')') {
```

3. Анализ решений (LLM)

js-easy-6 — 69.0/100

Комментарий:

> Итоговая оценка за задачу: 69.0/100 Оценка кода: 90/100 Функция корректно обрабатывает положительные и отрицательные числа, а также работает с нулём. Логика прохождения по строке и проверки символов понятна. Однако можно упростить код, избегая лишних преобразований и проверок. Также стоит уточнить, что символы '.' и ',' не обрабатываются, хотя в задаче не указано их игнорирование. Оценка коммуникации: 20/100 Ответ кандидата 'Next please' не отвечает на поставленный вопрос. Интервьюер спрашивал пояснение к выбранному подходу, но кандидат не предоставил никакого объяснения, логики или мотивации behind his solution. Это указывает на непонимание сути задачи, отсутствие способности аргументировать свой выбор, а также на низкий уровень коммуникации. Ответ не демонстрирует понимания того, как работает код, почему выбраны определённые конструкции, или какие альтернативы рассматривались. Такой ответ не соответствует ожиданиям на уровне технического интервью и свидетельствует о слабом понимании материала.

Проблемы не обнаружены.

js-med-6 — 68.5/100

Комментарий:

> Итоговая оценка за задачу: 68.5/100 Оценка кода: 85/100 Функция корректно реализует преобразование строки в camelCase. Учитывает обработку пробелов, регистр и корректную обработку первого слова. Есть небольшая оптимизация в части проверки пустых строк и лишних пробелов, но общая логика верна. Работает для всех примеров. Оценка коммуникации: 30/100 Ответ кандидата 'I dont know' не соответствует поставленному вопросу. Интервьюер спрашивал о мотивации и логике выбора подхода, но кандидат не продемонстрировал понимания своей логики или обоснования выбора конкретных действий в коде. Он не объяснил, почему выбран метод split(), как работает map с условием, почему используется trim() и join(), а также не раскрыл логику преобразования регистра. Ответ не показывает структурированного мышления или владения концепциями, используемыми в решении. Это указывает на слабое понимание собственного кода и отсутствие способности аргументировать свои действия, что снижает оценку.

Проблемы не обнаружены.

js-hard-9 — 65.5/100

Комментарий:

> Итоговая оценка за задачу: 65.5/100 Оценка кода: 85/100 Решение корректно реализует парсер выражений с поддержкой приоритетов операций и скобок. Использован алгоритм Шейнера для перевода в ОПЗ и вычисления. Есть небольшая проблема с обработкой унарного минуса, но это не критично для задачи. Код читаемый, структурирован, покрыт примерами. Оценка коммуникации: 20/100 Ответ кандидата не соответствует вопросу. Вопрос был: 'Поясните, почему вы выбрали именно такой подход?', то есть интервьюер ожидал объяснения выбора алгоритма, структур данных и логики реализации. Однако кандидат просто написал 'Вот так вот', что является абсолютно непонятным и неполным ответом. Он не раскрыл ни одного аспекта своего решения: почему выбрана схема Шейнера (токенизация + ОПЗ), почему используется стек для операторов, как работает обработка приоритетов и скобок. Также отсутствует логическая структура и понимание сути реализации. Это указывает на недостаточное понимание собственного кода и неумение его аргументировать. Такой

ответ не отражает профессионального уровня и не демонстрирует знания алгоритмов.

Проблемы не обнаружены.

4. Итоговая оценка кандидата

Средняя оценка за код: 67.67/100

Средняя оценка за коммуникацию: 23.33/100

Сильные стороны:

• Кандидат демонстрирует способность писать функциональный код, который корректно решает поставленные задачи. В каждой из задач реализована логика, соответствующая требованиям.

• В задачах с числами и строками кандидат использует стандартные методы JavaScript, такие как String(), parseInt(), split(), map(), что указывает на базовое знание языка.

• В сложной задаче реализован алгоритм Шейнера для парсинга выражений, что демонстрирует знакомство с алгоритмами перевода в ОПЗ и вычисления выражений.

Слабые стороны:

• Кандидат не демонстрирует понимание своего кода и не может объяснить выбор подхода, что указывает на поверхностное знание решаемых задач.

• Отсутствует способность аргументировать выбор технических решений, обосновать выбор алгоритмов или структур данных, что критично для позиции разработчика.

• Низкий уровень коммуникации: ответы на вопросы интервьюера не соответствуют поставленным вопросам, что свидетельствует о неумении структурировать мысли и объяснять свою логику.

Рекомендации:

• Рекомендуется углубить понимание алгоритмов и структур данных, а также развить навыки объяснения своих решений.

• Необходимо тренировать коммуникацию в технических интервью: уметь структурировать ответы, объяснять выборы и обосновывать решения.

• Полезно будет практиковать объяснение своих решений вслух, чтобы развить навык аргументирования и умение отвечать на технические вопросы.

Итог: Кандидат обладает базовыми техническими навыками для написания кода, но не демонстрирует понимание собственных решений и не может аргументировать выбор подходов. Это указывает на необходимость развития технической глубины и коммуникативных навыков.

5. Коммуникативные ответы

Задача: js-easy-6

Вопрос: Поясните, почему вы выбрали именно такой подход?

Ответ: Next please

Оценка: 20/100

Ответ кандидата 'Next please' не отвечает на поставленный вопрос. Интервьюер спрашивал

пояснение к выбранному подходу, но кандидат не предоставил никакого объяснения, логики или мотивации behind his solution. Это указывает на непонимание сути задачи, отсутствие способности аргументировать свой выбор, а также на низкий уровень коммуникации. Ответ не демонстрирует понимания того, как работает код, почему выбраны определённые конструкции, или какие альтернативы рассматривались. Такой ответ не соответствует ожиданиям на уровне технического интервью и свидетельствует о слабом понимании материала.

Задача: js-med-6

Вопрос: Поясните, почему вы выбрали именно такой подход?

Ответ: I dont know

Оценка: 30/100

Ответ кандидата 'I dont know' не соответствует поставленному вопросу. Интервьюер спрашивал о мотивации и логике выбора подхода, но кандидат не продемонстрировал понимания своей логики или обоснования выбора конкретных действий в коде. Он не объяснил, почему выбран метод split(), как работает map с условием, почему используется trim() и join(), а также не раскрыл логику преобразования регистра. Ответ не показывает структурированного мышления или владения концепциями, используемыми в решении. Это указывает на слабое понимание собственного кода и отсутствие способности аргументировать свои действия, что снижает оценку.

Задача: js-hard-9

Вопрос: Поясните, почему вы выбрали именно такой подход?

Ответ: Вот так вот

Оценка: 20/100

Ответ кандидата не соответствует вопросу. Вопрос был: 'Поясните, почему вы выбрали именно такой подход?', то есть интервьюер ожидал объяснения выбора алгоритма, структур данных и логики реализации. Однако кандидат просто написал 'Вот так вот', что является абсолютно непонятным и неполным ответом. Он не раскрыл ни одного аспекта своего решения: почему выбрана схема Шейнера (токенизация + ОПЗ), почему используется стек для операторов, как работает обработка приоритетов и скобок. Также отсутствует логическая структура и понимание сути реализации. Это указывает на недостаточное понимание собственного кода и неумение его аргументировать. Такой ответ не отражает профессионального уровня и не демонстрирует знания алгоритмов.

6. Анализ античита

Вставок: 3, Смена вкладки: 7, Анализов LLM: 10

Задача: js-easy-6

Вероятность списывания: 65% (уровень: medium)

Кандидат написал корректное решение задачи, но есть несколько подозрительных факторов. За один снимок кода произошло значительное увеличение (672 символа), что может указывать на вставку большого блока кода. Также зафиксирована одна вставка из буфера обмена, что может быть признаком списывания. Четыре выхода из вкладки могут свидетельствовать о переключении на другие вкладки или окна, однако не являются однозначным признаком списывания. Общая структура кода выглядит логично, без явных признаков готового решения из интернета.

Подозрительные события:

- code_growth (medium): Резкое увеличение размера кода на 672 символа за один снимок
- clipboard_paste (low): Одна вставка из буфера обмена
- tab_switches (low): Четыре выхода из вкладки

Задача: js-med-6

Вероятность списывания: 35% (уровень: low)

Код написан самостоятельно, без признаков списывания. Изменения в коде происходили постепенно, без резких скачков. Единственная вставка из буфера обмена и один выход из вкладки не указывают на подозрительное поведение. Структура кода логична и соответствует задаче. Нет признаков использования готовых решений из интернета.

Подозрительные события:

- clipboard_paste (low): Одна вставка из буфера обмена
- tab_switch (low): Один выход из вкладки

Задача: js-hard-9

Вероятность списывания: 65% (уровень: medium)

Код значительно увеличился за один снимок (3659 символов), что указывает на возможное списывание или использование готового решения. Одна вставка из буфера обмена и два выхода из вкладки также повышают подозрительность. Однако структура кода соответствует стандартной реализации парсера выражений, без явных признаков копирования из интернета.

Подозрительные события:

- code_growth (high): Резкое увеличение размера кода на 3659 символов за один снимок
- clipboard_paste (medium): Одна вставка из буфера обмена
- tab_switch (medium): Два выхода из вкладки во время работы