**Введение. Whiteboard coding**

В первой теме из цикла подготовки к собеседованию мы упомянули, что одним из этапов технического интервью может стать практическая часть, на которой вас попросят написать код для решения какой-то задачи. Поэтому тренировка решения задач на JS повышает шансы пройти этап технического интервью успешно.

Решение задач на собеседовании называют whiteboard coding от английского «написание кода на маркерной доске». Всё потому, что писать код вам придётся на доске или листе бумаги, и возможности проверить, работает программа или нет, у вас не будет.

В этой теме вы будете учиться решать такие задачи. А также потренируетесь писать код на JavaScript. Сначала будет море дурацких ошибок вроде пропущенных скобок и запятых. Но со временем тренировка даст результат, и ошибок станет меньше.

От слов перейдём к делу. Вот пример задачи, которая может попасться на собеседовании:

Напишите функцию getProperty, принимающую два аргумента: объект и путь до свойства, которое нужно из объекта достать. Путь до свойства — это строка ключей, разделённых точками.

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

*// здесь предстоит написать код*

}

const object = {

one: 1,

two: {

three: 3

},

four: 4

};

getProperty(object, 'one'); *// 1*

getProperty(object, 'two.three'); *// 3*

Рекомендуем попытаться решить эту задачу самостоятельно. В следующем уроке мы разберём решение и опишем, какой ход мыслей поможет к этому решению прийти.

# Как решать задачи?

Итак, вы пришли на собеседование. Очень волнуетесь. Перед вами сидит интервьюер, он показывает условие задачи:

Скопировать кодJAVASCRIPT

*/\* Напишите функцию getProperty(obj, path), принимающую два аргумента:*

*объект и путь до свойства, которое нужно достать из объекта.*

*Путь до свойства — это строка ключей, разделённых точками. \*/*

function getProperty(obj, path) {

*// здесь предстоит написать код*

}

const object = {

one: 1,

two: {

three: 3

},

four: 4

};

getProperty(object, 'one'); *// 1*

getProperty(object, 'two.three'); *// 3*

Но с чего начать решение, если в голове от волнения нет ни одной идеи?

# 1. Расслабьтесь

От волнения не убежишь, оно будет присутствовать всегда. Просто подумайте, что это если и первое, то точно не последнее ваше собеседование. Более того, уже тем, что пришли на него, вы уже оставили многих других разработчиков позади. Ещё справиться с волнением поможет начало диалога с интервьюером. Следующий пункт как раз про то, как это сделать.

# 2. Задавайте вопросы

Не бросайтесь сразу же писать код. Цель интервью — понаблюдать за ходом ваших мыслей, а не получить мгновенный ответ. Поэтому условие может быть намеренно немного туманным, чтобы подтолкнуть вас к вопросам.

Какие вопросы стоит задать в нашей исходной задаче? Например, в условии сказано, что функция должна вернуть свойство. Но что если свойства по переданному пути нет? Что тогда возвращает функция?

На этот вопрос интервьюер может ответить по-разному. Предположим, он отвечает, что свойство по переданному пути есть в объекте всегда.

Это избавляет вас от ряда проблем, так что можно переходить к построению алгоритма. В конце этой темы мы разберём, как быть с иными ответами. А сейчас — идём дальше.

# 3. Рассуждайте вслух

Даже когда условие ясно, не стоит бросаться писать код. Порассуждайте вслух: как решать задачу, с чего начать?

В нашей задаче путь к свойству — строка ключей, разделённых точками. Будет удобно разбить её методом split. Это первый, довольно простой шаг, с которого удобно взяться за рассуждение.

Затем начнём проходить по этому массиву в цикле. Возьмём первый ключ и достанем из объекта соответствующее ему свойство. Результат сохраним в переменной: следующий ключ мы будем искать уже в ней, а не в исходном объекте. Когда ключи в массиве закончатся, вернём результат.

Такими могут быть ваши устные рассуждения. В следующем шаге начнём перекладывать их на бумагу.

# 4. Напишите псевдокод

Параллельно с устными рассуждениями можно начать писать так называемый «псевдокод». Псевдокод — это алгоритм программы, описанный естественным языком, а не программным. Это промежуточная ступень между нашими мыслями и кодом. Его задача — описать алгоритм, опустив детали реализации.

Изложим наши рассуждения псевдокодом:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

разбить path в массив и положить результат в переменную arr

создать переменную res и присвоить ей obj

for (пройтись по массиву от начала и до конца) {

взять элемент массива

достать из переменной результата свойство по соответствующему ключу

записать полученное свойство в переменную res

начать всё сначала

}

вернуть res

}

Алгоритм готов. Осталось сделать из него код.

# 5. Напишите код

Когда алгоритм готов, код писать значительно проще. Начинаем «переводить» псевдокод на JavaScript:

Скопировать кодJAVASCRIPT

*// разбить path в массив и положить результат в переменную arr*

const arr = path.split('.');

В итоге код получится таким:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

*// разбить path в массив и положить результат в переменную arr*

const arr = path.split('.');

*// создать переменную res и присвоить ей obj*

let res = obj;

*// пройтись по массиву от начала и до конца*

for (let i = 0; i < arr.length; i += 1) {

*// взять элемент массива*

*// достать из переменной результата свойство по соответствующему ключу*

*// записать полученное свойство в переменную res*

res = res[arr[i]];

}

*// вернуть res*

return res;

}

Задача решена. Но это ещё не всё. В следующем уроке разберём, как решение можно улучшить.

**Когда задача решена**

В прошлом уроке мы получили такой код:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

const arr = path.split('.');

let res = obj;

for (let i = 0; i < arr.length; i += 1) {

res = res[arr[i]];

}

return res;

}

Редко получается найти идеальное решение с первого раза. Подумайте, как можно улучшить наш код, а в следующем уроке мы разберём это вместе.

# Анализируем и улучшаем решение

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

const arr = path.split('.');

let res = obj;

for (let i = 0; i < arr.length; i += 1) {

res = res[arr[i]];

}

return res;

}

Итак, наше решение работает, но его можно улучшить. Давайте попробуем это сделать.

## Сломайте решение

Это прозвучит варварски, но попробуйте сломать написанный код. Подумайте, какие аргументы нужно передать функции на вход, чтобы решение перестало работать. Вначале мы условились, что свойство по ключу обязательно есть в объекте, но если передать ключ 'five.six', программа сломается, что не очень хорошо. Поэтому большим плюсом будет, если вы сможете это исправить:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

const arr = path.split('.');

let res = obj;

for (let i = 0; i < arr.length; i += 1) {

res = res[arr[i]];

}

return res;

}

const object = {

one: 1,

two: {

three: 3

},

four: 4

};

getProperty(object, 'five.six'); *// TypeError: Cannot read property 'six' of undefined*

Ошибка Cannot read property 'six' of undefined . Действительно, сначала мы обращаемся к свойству five объекта obj. Такого свойства в объекте нет, поэтому мы получаем undefined. Дальше в цикле пытаемся обратиться по ключу six в undefined:

Скопировать кодJAVASCRIPT

const object = {

one: 1,

two: {

three: 3

},

four: 4

};

object.five; *// undefined*

object.five.six; *// TypeError*

Если попытаться обратиться по ключу в undefined или null, будет ошибка. Давайте учтём это в нашем коде. Для этого в цикле будем проверять переменную res:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

const arr = path.split('.');

let res = obj;

for (let i = 0; i < arr.length; i += 1) {

*// проверим, что по ключу обратиться можно*

if (res === undefined || res === null) {

console.log('Такого свойства нет');

*// и выйдем из функции*

return undefined;

}

res = res[arr[i]];

}

return res;

}

Теперь функция будет прекращать выполнения, если свойства по переданному пути нет. Это лишь один из способов сделать наш код крепче. Можно развивать этот ход мыслей дальше и подумать, что будет, если вместо obj в функцию придёт не объект, а вместо path — не строка.

## Подумайте, как сделать решение более изящным

Это необязательный шаг, но он добавит вам очков. В задаче необходимо получить значение из объекта на основе массива ключей. Помните, какой метод в JS позволяет получить единственное значение на основе массива? Это метод reduce — идеальный кандидат для использования в этой задаче. Перепишем наш код с использованием reduce:

Скопировать кодJAVASCRIPT

function getProperty(obj, path) {

return path.split('.').reduce(function (res, key) {

if (res === undefined || res === null) {

console.log('Такого свойства нет');

return undefined;

}

return res[key];

}, obj);

}

Есть и другие способы анализа и улучшения кода. Например, можно оценить вычислительную сложность алгоритма. Эту тему мы затронем в следующем спринте.

# Как тренировать алгоритмическое мышление?

Сначала будет тяжело. Код предательски откажется работать, приводя к неправильному результату и засыпая консоль ошибками. Это похоже на бег: сложно сразу одолеть 5 километров без подготовки. Но через год регулярных пробежек и полумарафон покажется посильной задачей.

И навык программирования, и алгоритмическое мышление можно развить, если регулярно тренироваться. Старайтесь несколько раз в неделю уделять этому время. А чтобы тренировки приносили больше плодов, соблюдайте простые правила.

# 1. Сначала думать — потом кодить

Ещё раз: не бросайтесь сразу писать код. Сначала постройте алгоритм в голове. Этот этап хорош ещё и тем, что думать над решением можно пока едешь в метро или сидишь в очереди в паспортном столе.

После того, как алгоритм придуман — пишите код. Начинайте с псевдокода, а затем переводите его. Когда всё готово, переходите к следующему пункту.

# 2. Анализируйте написанное

Не спешите сразу проверять код в браузере. Ищите в нём ошибки и мысленно ломайте его. Если удалось что-то найти — переработайте код.

# 3. Запускайте код

Когда код готов, запустите его в браузере. Можно прямо скопировать код в консоль и тестировать функцию, вызывая её с разными аргументами.

Код отрабатывает без ошибок? Поздравляем: вы один из немногих, чей код сработал с первого раза. Если есть ошибка — не расстраивайтесь. Попытайтесь её найти. Очень удобно выводить значения переменных в разных частях программы вызовом console.log. Есть и другие, более продвинутые способы отладки: мы расскажем о них позже, в отдельном курсе.

Когда ошибка найдена — исправьте её и тщательно проанализируйте, чтобы не допустить того же промаха снова.

# 4. Повышайте сложность постепенно

Подбирайте задачи своего уровня. Пробежать марафон без тренировок — идея, чреватая провалом. С программированием так же: если вы чувствуете, что задача сложная, найдите для начала пример попроще и постепенно увеличивайте сложность.

# 5. Не переживайте из-за неудач

Это самое главное правило. Когда сталкиваешься с трудной задачей, может возникнуть куча дурацких мыслей из разряда «я гуманитарий, это не моё». Все мы через это прошли. Уверяем вас: дело не в каком-то особом складе ума, а в упорстве и практике. Тренируйтесь — всё получится.

# Решите несколько задач

Решите три задачи. Перед тем, как нажимать «Проверить», убедитесь что:

* **Решение работает без ошибок на большинстве оговоренных входных данных.** То есть, если в условии сказано, что задача принимает массив целых чисел, то при проверке на множестве разных массивов с целыми числами всё должно работать.
* **Решение работает в краевых случаях.** Например, мы можем передать в функцию не массив целых чисел, а пустой массив, и решение не должно ломаться.

Удачи! И не забывайте: сначала подумать, а потом написать код.

Задачки

Напишите функцию reverseWords, принимающую на вход строку. Функция должна вернуть новую строку, расставив слова в обратном порядке. Если в строке есть знаки препинания, их можно удалить или оставить — на ваше усмотрение.

function reverseWords(str) {

const arr = str.split(' ');

newArr = arr.reverse();

res = newArr.join(' ');

return res

}

reverseWords('всегда много путей достичь цель есть'); // "есть цель достичь путей много всегда"

reverseWords('испробовать их все должны вы'); // "вы должны все их испробовать"

Напишите функцию capitalize. Функция должна возвращать новую строку, в которой каждое слово начинается с прописной буквы.

capitalize('Надоел, надоел, надоел ты'); *// Надоел, Надоел, Надоел Ты*

capitalize('Надоели, надоели, надоели вы все'); *// Надоели, Надоели, Надоели Вы Все*

capitalize('Прошу уходи, уходи'); *// Прошу Уходи, Уходи*

capitalize('Уходи прошу, уходи совсем'); *// Уходи Прошу, Уходи Совсем*

*function capitalize(str) {*

*const arr = str.split(' ');*

*const newArr = arr.map((item) => {*

*return(item.slice(0, 1).toUpperCase() + item.slice(1));*

*});*

*return newArr.join(' ');*

*}*