24. задание Реализуйте функцию truncate(), которая обрезает переданную строку до указанного количества символов, добавляет в конце многоточие и возвращает получившуюся строку. Подобная логика часто используется на сайтах, чтобы отобразить длинный текст в сокращенном виде.

Функция принимает два параметра:

1. Строка, которую нужно обрезать
2. Число символов, которые нужно оставить

|  |  |
| --- | --- |
|  | *// В функцию могут быть переданы в качестве параметров* |
|  | *// любая строка и любое количество символов* |
|  |  |
|  | *// Обрезаем текст 'hexlet', оставляя 2 символа* |
|  | truncate('hexlet', 2); *// "he..."* |
|  |  |
|  | *// Обрезаем текст 'it works!', оставляя 4 символа* |
|  | truncate('it works!', 4); *// 'it w...'* |
|  |  |
|  | *// Обрезаем текст 'hello', оставляя 3 символа* |
|  | truncate('hello', 3); *// 'hel...'* |

Подсказки

* Вам нужно самостоятельно реализовать функцию truncate(), в JS такой функции не существует. Реализовать эту функцию можно различными способами, подскажем лишь один из них. Для решения этим способом вам понадобится взять подстроку из строки, переданной первым параметром в функцию. Используйте для этого метод [slice()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/slice). Подумайте, исходя из задания, с какого индекса и по какой вам надо извлечь подстроку?

|  |  |
| --- | --- |
|  | const word = 'welcome'; |
|  | *// Передавать параметры в метод можно через переменные* |
|  | const index = 3; |
|  | word.slice(0, index); *// "wel"* |

С точки зрения проверочной системы не имеет значения, каким из способов будет реализована функция truncate() внутри, главное – чтобы она выполняла поставленную задачу

* Вам не нужно задавать конкретные значения для параметров, они будут переданы в функцию во время её тестирования.
* Чтобы функция могла работать с любой строкой, используйте внутри функции переменные – параметры функции. После вызова функции эти переменные примут конкретные значения. Так функция сможет обрезать любую строку, которую мы передадим ей при вызове

РЕШЕНИЕ

const truncate = (text, length) => {

  // BEGIN

  const result = `${text.slice(0, length)}...`;

  return result;

};

// END

export default truncate;

25. getHiddenCard.js

Реализуйте функцию getHiddenCard(), который принимает на вход номер кредитки (состоящий из 16 цифр) в виде строки и возвращает его скрытую версию, которая может использоваться на сайте для отображения. Если исходная карта имела номер *2034399002125581*, то скрытая версия выглядит так *\*\*\*\*5581*. Другими словами, функция заменяет первые 12 символов, на звездочки. Количество звездочек регулируется вторым необязательным параметром. Значение по умолчанию — 4.

|  |  |
| --- | --- |
|  | *// Кредитка передается внутрь как строка* |
|  | getHiddenCard('1234567812345678', 2); *// "\*\*5678"* |
|  | getHiddenCard('1234567812345678', 3); *// "\*\*\*5678"* |
|  | getHiddenCard('1234567812345678'); *// "\*\*\*\*5678"* |
|  | getHiddenCard('2034399002121100', 1); *// "\*1100"* |

Для выполнения задания вам понадобится метод строки repeat(), который повторяет строку указанное количество раз

|  |  |
| --- | --- |
|  | '+'.repeat(5); *// "+++++"* |
|  | 'o'.repeat(3); *// "ooo"* |

решение

1 вариант const getHiddenCard=(numberCard,star=4)=>{

  const result='\*'.repeat(star);

  const cut = numberCard.slice(12);

  return (`${result}${cut}`)

}

// END

2 вариант // BEGIN

const getHiddenCard = (cardNumber, starsCount = 4) => {

const visibleDigitsLine = cardNumber.slice(12);

return `${'\*'.repeat(starsCount)}${visibleDigitsLine}`;

};

// END