26. Реализуйте функцию capitalize(), которая принимает непустую строку и приводит первую букву строки к верхнему регистру:

|  |  |
| --- | --- |
|  | const name = 'arya'; |
|  | console.log(capitalize(name)); *// => "Arya"* |

Чтобы получить подстроку (или символ) из строки, используйте метод [slice()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/slice):

'welcome'.slice(2, 5); *// "lco"*

Для приведения строки к верхнему регистру используйте метод [toUpperCase()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase" \t "_blank)

'welcome'.toUpperCase(); *// "WELCOME"*

// BEGIN (write your solution here)

capitalize=()=>{

}

// END

27. Реализуйте функцию isInternationalPhone(), которая проверяет формат указанного телефона. Если телефон начинается с *+*, значит это международный формат.

|  |  |
| --- | --- |
|  | isInternationalPhone('89602223423'); *// false* |
|  | isInternationalPhone('+79602223423'); *// true* |

28. Реализуйте функцию isLeapYear(), которая определяет, является ли год високосным или нет. Год будет високосным, если он кратен (то есть делится без остатка) 400 или он одновременно кратен 4 и не кратен 100. Как видите, в определении уже заложена вся необходимая логика, осталось только переложить её на код:

|  |  |
| --- | --- |
|  | isLeapYear(2018); *// false* |
|  | isLeapYear(2017); *// false* |
|  | isLeapYear(2016); *// true* |

Кратность можно проверять так:

|  |  |
| --- | --- |
|  | *// % - возвращает остаток от деления левого операнда на правый* |
|  | *// Проверяем что number кратен 10* |
|  | number % 10 === 0 |
|  |  |
|  | *// Проверяем что number не кратен 10* |
|  | number % 10 !== 0 |

29. Реализуйте функцию getLetter(), которая извлекает из переданной строки указанный символ (по порядковому номеру, а не индексу) и возвращает его наружу. Если такого символа нет, то функция возвращает пустую строку.

Примеры вызова:

|  |  |
| --- | --- |
|  | const name = 'Hexlet'; |
|  |  |
|  | *// Обычное обращение возвращает undefined* |
|  | name[10]; *// undefined* |
|  |  |
|  | *// 11 символ соответствует 10 индексу* |
|  | getLetter(name, 11); *// ''* |
|  |  |
|  | getLetter(name, 1); *// 'H'* |
|  | getLetter(name, 0); *// ''* |
|  | getLetter(name, 6); *// 't'* |

30. normalizeUrl.js

Реализуйте функцию normalizeUrl(), которая выполняет так называемую нормализацию данных. Она принимает адрес сайта и возвращает его с *https://* в начале.

Функция принимает адреса в виде *АДРЕС* или *https://АДРЕС*, но всегда возвращает адрес в виде *https://АДРЕС*

Можно использовать метод [startsWith()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/startsWith), чтобы проверить, начинается ли строка с префикса *https://*. А потом на основе этого добавлять или не добавлять *https://*.

Примеры вызова:

|  |  |
| --- | --- |
|  | normalizeUrl("google.com"); *// "https://google.com"* |
|  | normalizeUrl("https://ai.fi"); *// "https://ai.fi"* |

31. Реализуйте функцию convertText(), которая принимает на вход строку и, если первая буква не заглавная, возвращает перевернутый вариант исходной строки. Если первая буква заглавная, то строка возвращается без изменений. Если на вход передана пустая строка, функция должна вернуть пустую строку.

Примеры вызова:

|  |  |
| --- | --- |
|  | convertText('Hello'); *// 'Hello'* |
|  | convertText('hello'); *// 'olleh'* |
|  |  |
|  | *// Не забудьте учесть пустую строку* |
|  | convertText(''); *// ''* |

Перевернуть строчку можно используя функцию reverse(). В качестве аргумента в неё нужно передать строку, которую мы хотим перевернуть:

|  |  |
| --- | --- |
|  | const result = reverse('Hello!'); |
|  | console.log(result); *// => '!olleH'* |

Есть разные подходы к решению этой задачи. Возможно, вам пригодится метод [toUpperCase()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase" \t "_blank) и возможность получения символа из строки (например, str[0]).

Попробуйте написать два варианта функции: с обычным if-else, и с тернарным оператором.

Подсказки

* Подумайте какую проверку нужно написать первым делом: проверку заглавности буквы или является ли пустой строка. Что первичнее?

import reverse from './reverse';

// BEGIN (write your solution here)

// END

export default convertText;

32. Реализуйте функцию getNumberExplanation(), которая принимает на вход число и возвращает объяснение этого числа. Если для числа нет объяснения, то возвращается just a number. Объяснения есть только для следующих чисел:

|  |  |
| --- | --- |
|  | \* 666 - devil number |
|  | \* 42 - answer for everything |
|  | \* 7 - prime number |

Примеры вызова функции:

|  |  |
| --- | --- |
|  | getNumberExplanation(8); *// just a number* |
|  | getNumberExplanation(666); *// devil number* |
|  | getNumberExplanation(42); *// answer for everything* |
|  | getNumberExplanation(7); *// prime number* |

33. Модифицируйте функцию printNumbers() так, чтобы она выводила числа в обратном порядке. Для этого нужно идти от верхней границы к нижней. То есть счётчик должен быть инициализирован максимальным значением, а в теле цикла его нужно уменьшать до нижней границы.

printNumbers(4);

|  |  |
| --- | --- |
|  | 4 |
|  | 3 |
|  | 2 |
|  | 1 |
|  | finished! |

// @ts-check

/\* eslint-disable no-console \*/

const printNumbers = (initialNumber) => {

  // BEGIN (write your solution here)

  // END

};export default printNumbers;

34. Реализуйте функцию joinNumbersFromRange(), которая объединяет все числа из диапазона в строку и возвращает её:

|  |  |
| --- | --- |
|  | joinNumbersFromRange(1, 1); *// '1'* |
|  | joinNumbersFromRange(2, 3); *// '23'* |
|  | joinNumbersFromRange(5, 10); *// '5678910'* |

// BEGIN (write your solution here)

// END

export default joinNumbersFromRange;

35. Реализуйте функцию mySubstr(), которая извлекает из строки подстроку указанной длины. Она принимает на вход два аргумента (строку и длину) и возвращает подстроку, начиная с первого символа:

|  |  |
| --- | --- |
|  | const text = 'If I look back I am lost'; |
|  | console.log(mySubstr(text, 1)); # => 'I' |
|  | console.log(mySubstr(text, 7)); # => 'If I lo' |

В этом задании аргумент, обозначающий длину извлекаемой подстроки, не может быть длиннее самой строки.

Используйте тот же подход, что в функции для переворота строки из урока: собирайте строку-результат в цикле, перебирая начальную строку до определённого момента.

36. Функция countChars() из теории считает, сколько раз входит буква в предложение и при этом учитывает регистр букв. То есть A и a с её точки зрения разные символы. Реализуйте вариант этой же функции, так чтобы регистр букв был не важен:

|  |  |
| --- | --- |
|  | countChars('HexlEt', 'e'); *// 2* |
|  | countChars('HexlEt', 'E'); *// 2* |

37. Напишите функцию makeItFunny(), которая принимает на вход строку и возвращает её копию, у которой каждый n-ный элемент переведен в верхний регистр. n – задается на входе в функцию.

Для определения каждого n-ного элемента понадобится остаток от деления %. Подумайте, как его можно использовать.

|  |  |
| --- | --- |
|  | const text = 'I never look back'; |
|  | *// Каждый третий элемент* |
|  | makeItFunny(text, 3); *// 'I NevEr LooK bAck'* |

Пример работы оператора остатка от деления:

|  |  |
| --- | --- |
|  | console.log(20 % 3) *// => 2* |
|  | *// Так как 3 \* 6 = 18* |

38. Сэмвелл обнаружил, что его сообщения перехватываются в замке «Близнецы» и там читаются. Из-за этого их атаки перестали быть внезапными. Немного подумав, он разработал программу, которая бы шифровала сообщения по следующему алгоритму. Она бы брала текст и переставляла в нем каждые два подряд идущих символа.

|  |  |
| --- | --- |
|  | encrypt('move'); *// 'omev'* |
|  | encrypt('attack'); *// 'taatkc'* |
|  | encrypt('car!'); *// 'ac!r'* |
|  |  |
|  | *// Если число символов нечётное,* |
|  | *// то последний символ остаётся на своём месте* |
|  | encrypt('go!'); *// 'og!'* |

Реализуйте функцию encrypt(), которая принимает на вход исходное сообщение и возвращает зашифрованное.

Подумайте. Может ли эта функция расшифровать зашифрованное сообщение?

26. ответ // BEGIN (write your solution here)

const capitalize = (text) => `${text[0].toUpperCase()}${text.slice(1)}`;

// END

export default capitalize;

27. // BEGIN

isInternationalPhone = (phone) => phone[0,1]==+7;

// END

export default isInternationalPhone;

28. // BEGIN (write your solution here)

const isLeapYear=(year)=>

  {  const result = year % 400 === 0 || (year % 4 === 0 && year % 100 !== 0);

  return result;

}

// END

29. // BEGIN

const getLetter = (text, position) => text[position - 1] || '';

// END

30. // BEGIN

const normalizeUrl = (site) => {

  let normalizedUrl;

  if (site.startsWith('https://')) {

    normalizedUrl = site;

  } else {

    normalizedUrl = `https://${site}`;

  }

  return normalizedUrl;

};

// END

export default normalizeUrl;

31. // BEGIN

const convertText = (text) => {

if (text === '') {

return '';

}

const reversable = text[0] !== text[0].toUpperCase();

return reversable ? reverse(text) : text;

};

// END

32. // BEGIN

const getNumberExplanation = (number) => {

  switch (number) {

    case 666:

      return 'devil number';

    case 7:

      return 'prime number';

    case 42:

      return 'answer for everything';

    default:

      return 'just a number';

  }

};

// END

export default getNumberExplanation;

33. // @ts-check

/\* eslint-disable no-console \*/

const printNumbers = (initialNumber) => {

  // BEGIN

    let i = initialNumber;

    while (i >= 1) {

      console.log(i);

      i = i - 1;

    }

    console.log('finished!');

  };

  // END

export default printNumbers;

34. // BEGIN

const joinNumbersFromRange = (start, finish) => {

  let i = start;

  let result = '';

  while (i <= finish) { result = `${result}${i}`;

    i = i + 1;

  }

  return result;

};

// END

export default joinNumbersFromRange;

35. // BEGIN (write your solution here)

const mySubstr=(str,length)=>{

let i = 0;

let result = '';

while (i < str.length) {

  result=`${result}${str[i]}`;

  i=i+1;

  return result;

}

// END

// BEGIN

const mySubstr = (string, length) => {

  let resultString = '';

  let i = 0;

  while (i < length) {

    resultString += string[i];

    i += 1;

  }

  return resultString;

};

// END

export default mySubstr;

36. // BEGIN

const countChars=(word,letter)=>{

  let i=0;

  let numberletter=0;

  while (i<word.length){

   if (word[i].toLowerCase()===letter.toLowerCase()){

     numberletter=numberletter+1

     }

    i=i+1;

  }

  return numberletter;

}

// END

export default countChars;

37. /\* eslint-disable no-plusplus \*/

// BEGIN

const makeItFunny = (text, n) => {

  let i = 0;

  let result = '';

  while (i < text.length) {

    const current = text[i];

    if ((i + 1) % n === 0) {

      result = `${result}${current.toUpperCase()}`;

    } else {

      result = `${result}${current}`;

    }

    i++;

  }

  return result;

};

// END

export default makeItFunny;

38. const encrypt = (str) => {

  let result = '';

  for (let i = 0; i < str.length; i += 2) {

    const nextSymbol = str[i + 1] || '';

    result = `${result}${nextSymbol}${str[i]}`;

  }

  return result;

};

// END

export default encrypt;