GlintsAcademy
Powered by BINAR ACADEMY

Code Challenge

Isumi
Jakarta, June 2020

BINAR ACADEMY

GOALS

1. JS (Easy): Palindrome
2. JS (Easy): Power of Two
3. JS (Easy): Other Products
4. JS (Medium): Letter Capitalize
5. JS (Medium): Letter Changes
6. JS (Medium): Recursion
7. JS (Hard): Roman Numerals
8. JS (Hard): Numbers in Words
9. React (Easy): React Button Toggle
10. React (Easy): React Simple Counter
11. Bonus: React List

1. Palindrome

Have the function `Palindrome(str)` take the `str` parameter being passed and return the string **true** if the parameter is a palindrome, (the string is the same forward as it is backward) otherwise return the string **false**. For example: "racecar" is also "racecar" backwards. Punctuation and numbers will not be part of the string.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test = Palindrome("racecar");
console.log(test)
// Output: true
```

2. Powers of Two

Have the function `PowersofTwo(num)` take the `num` parameter being passed which will be an integer and return the string **true** if it's a power of two. If it's not return the string **false**. For example if the input is 16 then your program should return the string true but if the input is 22 then the output should be the string false.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test1 = PowersofTwo(22);
console.log(test)
// Output: false
let test2 = PowersofTwo(16);
console.log(test2)
// Output: true
```

3. Other Products

Have the function `OtherProducts(arr)` take the array of numbers stored in **`arr`** and return a new list of the products of all the other numbers in the array for each element. For example: if **`arr`** is [1, 2, 3, 4, 5] then the new array, where each location in the new array is the product of all *other* elements, is [120, 60, 40, 30, 24]. The following calculations were performed to get this answer: [(2*3*4*5), (1*3*4*5), (1*2*4*5), (1*2*3*5), (1*2*3*4)]. You should generate this new array and then return the numbers as a string joined by a hyphen: **120-60-40-30-24**. The array will contain at most 10 elements and at least 1 element of only positive integers.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test = OtherProducts([1,2,3,4,5]);
console.log(test)
// Output: 120-60-40-30-24
```

4. Letter Capitalize

Have the function `LetterCapitalize(str)` take the `str` parameter being passed and capitalize the first letter of each word. Words will be separated by only one space.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test = LetterCapitalize("hello world");
console.log(test)
// Output: Hello World
```

5. Letter Changes

Have the function `LetterChanges(str)` take the `str` parameter being passed and modify it using the following algorithm. Replace every letter in the string with the letter following it in the alphabet (**ie.** c becomes d, z becomes a). Then capitalize every vowel in this new string (a, e, i, o, u) and finally return this modified string.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test = LetterChanges("hello world");
console.log(test)
// Output: Ifmmp xpsmE
```

6. Recursion

Have the function `Recursion(num)` take the `num` parameter being passed which will be an integer and return the product of integer parameters, and return the product of integer parameters entered and recursively until the digits are only one number. For example:

`Recursion(39)` // Output: 4 (Because of 3*9 = 27 | 2*7 = 14 | 1*4 = **4 -> the only digit left**)
`Recursion(999)` // Output: 2 (Because of 9*9*9 = 729 | 7*2*9 = 126 | 1*2*6 = 12 | 1*2 = **2 -> the only digit left**)
`Recursion(3)` // Output: 3 (Because of **3 -> the only digit left**)

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
let test1 = Recursion(39);
let test2 = Recursion(999);
let test3 = Recursion(3);
console.log(test1) // Output: 4
console.log(test2) // Output: 2
console.log(test3) // Output: 3
```

7. Roman Numerals

Have the function `RomanNumerals(num)` take the `num` parameter being passed which will be an integer and convert from integer into roman numerals.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
console.log('Input | Expected | Actual')
console.log('------|----------|--------')
console.log('4     | IIII     | ', RomanNumerals(4)) // Output: IV
console.log('9     | VIIII    | ', RomanNumerals(9)) // Output: IX
console.log('14    | XIIII    | ', RomanNumerals(14)) // Output: XIV
console.log('1454  | MCDLIIII | ', RomanNumerals(1454)) // Output: MCDLIV
console.log('1644  | MDCXLIIII| ', RomanNumerals(1644)) // Output: MDCXLIV
```

8. Numbers in Words

Have the function `NumbersinWords(num)` take the **`num`** parameter being passed which will be an integer and convert from integer into words in Indonesian currency.

Use the **Parameter Testing** feature in the box below to test your code with different arguments.

```
console.log(Numbers_in_Words(9)); // Output: sembilan
console.log(Numbers_in_Words(987654321234567));
// Output: sembilan ratus delapan puluh tujuh triliun enam ratus lima puluh empat miliar
tiga ratus dua puluh satu juta dua ratus tiga puluh empat ribu lima ratus enam puluh tujuh
console.log(Numbers_in_Words(7000000000000000)); // Output: tujuh kuadriliun
```

9. React Button Toggle

We provided some simple React template code. Your goal is to modify the component so that you can properly toggle the button to switch between an ON state and an OFF state. When the button is on and it is clicked, it turns off and the text within it changes from `ON` to `OFF` and vice versa. Make use of component state for this challenge.

You are free to add classes and styles, but make sure you leave the element ID's as they are.

```
import React from 'react';
import ReactDOM from 'react-dom';

class Toggle extends React.Component {
 constructor(props) {
   super(props);
 }

 handleClick() {
   // todo
 }

 render() {
   return (
     <button>ON</button>
   );
 }
}

ReactDOM.render(
 <Toggle />,
 document.getElementById('root')
);
```

10. React Simple Counter

We provided some simple React template code. Your goal is to modify the component so that the counter correctly displays and it increments by one whenever the button is pressed. You are free to add classes and styles, but make sure you leave the element ID's as they are.

```
import React from 'react';
import ReactDOM from 'react-dom';

class Counter extends React.Component {
 constructor(props) {
  super(props);
 // todo
  }
 }
 countAdd = () => {
  // todo
 }

 render() {
  return (
    <div id="mainArea">
      <p>button count: <span></span></p>
      <button id="mainButton">Increase</button>
    </div>
  );
 }
}

ReactDOM.render(
 <Counter />,
 document.getElementById('root')
);
```

11. React List

We provided some simple React template code. Your goal is to display an unordered list (UL) with list items (LI) within it. The content of each list item should contain two spans (SPAN), one with the name and the other with the age passed in to the DataList function. The span elements should be separated by a single space.

Data Students

- Iqfar 20
- Udhin 23
- Ary 22
- Krisna 26
- Afnur 22

```
import React from 'react';
import ReactDOM from 'react-dom';

function DataList(props) {
 return (
   <div>
     <h2>Data Students</h2>
     <ul>
         // todo
     </ul>
   </div>
 );
}

const data = [
    // todo
];

ReactDOM.render(
 <DataList datas={ data } />,
 document.getElementById('root')
);
```

References

- Coderbyte
- `Publish to NPM`