JavaScript Problem Solving

Problem Solving

We are going to solve a problem step by step.

Problem Solving

We will first present an exercise and then go through the steps to solve it.

Problem Solving

We will present the steps in plain English.

Problem Solving

You will have to translate it to JavaScript.

Exercise

Write a function that decodes a secret message hidden in an array of words.

Exercise

Each word contains one letter of the message. Go through the words in order and retrieve one character from each of them.

Exercise

For the 1st word, grab the 1st character, for the 2nd word, the 2nd character and so on.

Exercise

When you get to the 6th word, start from the 1st character again.

Exercise

Take each of those characters and add them to a new string to form the message. Finally, return the message.

Exercise

```
var words, message;

words = [
    "dead",          // 1st -> d
    "bygone",        // 2nd -> y
    "landing",       // 3rd -> n
    "cheaply",       // 4th -> a
    "assumed",       // 5th -> m
    "incorrectly",   // 1st -> i
    "attention",     // 2nd -> t
    "agent"          // 3rd -> e
];

// message should be "dynamite"
message = decoder(words);
console.log(message);
```

Steps

Create a function called `decoder`.

Steps

The function will take one argument: the list of words.

Steps

We will need 2 variables inside the function.

Steps

First, the `index` to retrieve the 1st, 2nd, 3rd, 4th or 5th character from the word.

Steps

Second, the `secretMessage` that will store the pieces as of the message as we decode it.

Steps

```
function decoder (words) {
  var index = 0;
  var secretMessage = "";
}
```

Steps

Let’s iterate through all the words with a for loop.

Steps

At each iteration, retrieve the character we want from the word and add it to the `secretMessage` variable.

Steps

Update the index at the end of each iteration. That means increase it by 1.

Steps

What happens when we hit 6?
We need to start again at 0.

Steps

We can use an if statement that checks if the index is 6, and sets it to 0.

Steps

BONUS: A more elegant solution, though.

```
var index = (index + 1) % 5;
```

Steps

Return `secretMessage` at the end of the function.

Exercise

What is the secret message in these words?

```
var words2 = [
  "January", "lacks", "caveats",
  "hazardous", "DOORS", "crying",
  "arrogantly", "climate", "proponent",
  "rebuttal"
];
```

Solution

```
function decoder (words) {
   var index = 0;
   var secretMessage = "";
   for (var i = 0; i < words.length; i++) {
      secretMessage += words[i].charAt(index);
      index = (index + 1) % 5;
   }

   return secretMessage;
}
```

Exercise part 2

Now suppose that the secret message is a sentence. And that is decoded starting from the last word rather than the first.

Exercise part 2

So we don't decode from beginning to end, but from end to beginning. Backwards.

Exercise part 2

We would like the function to be able to handle either direction though.

Exercise part 2

We also want the function to be able to use either the odd words of the sentence, the even words of the sentence or all the words.

Exercise part 2

Let's create a function that takes the sentence and then a string that tells the function how to behave.

Exercise part 2

```
var sentence, message;

sentence = "fill the proper tank for the cow";
// we are taking only the even words
// "fill proper for cow"
// reverse them ["cow", "for", "proper", "fill"]
// call the previous decode function

// message should be "cool"
message = super_decode(sentence, "even-backwards");
console.log(message);
```

Exercise part 2

We should also use the function that we already created to avoid repeating ourselves.

Module Exports

In order to keep our files clean, we are going to keep the “decoding” function in one file, and the “super decoding” in another file

Module Exports

However, we will want to use the “decode” function in our new function.
How can we do that?

Module Exports

Using `module.exports` and `require`.

Module Exports

Similar to `require_relative` in Ruby.

Module Exports

However, you need to specify what value the file will be exporting.

Module Exports

In the decoder.js file add:

```
module.exports = decode;
```

Create a super_decoder.js file and add:

```
var decode = require("./decoder.js");
```

Module Exports

With `module.exports` we are telling what we want to retrieve when we use:

```
require("./some_file.js").
```

It can be anything, function, object, string. For example:

```
module.exports = "Export a stupid string";
```

Solution

file `decoder.js`

```
function decoder (words) {
   var index = 0;
   var secretMessage = "";
   for (var i = 0; i < words.length; i++) {
      secretMessage += words[i].charAt(index);
      index = (index + 1) % 5;
   }

   return secretMessage;
}

module.exports = decoder;
```

Solution

file `super_decoder.js`

```
var decoder = require("./decoder.js");
```

Steps

Create a function “super_decoder” in “super_decoder.js” file.

Steps

The function should take two strings: the sentence and the behavior string (e.g. “even-backwards”).

Steps

The logic should check the behavior string to figure out what needs to be done.

Steps

Depending on the value, you need to create a different array.

Remember `.split()` to create an array from a string

Steps

It might be the same elements but backwards.

Remember `.split()` to create an array from a string

Steps

It might be a new array with just the even words.

Remember `.split()` to create an array from a string

Steps

For example: when “backwards” is passed, we want to create the array and then reverse it, so that we can call the decoder on it.

Steps

Example for “backwards”

```
var sentence1 = "Attack her nose under here with an itch"
// split the sentence and reverse it
// ["itch, "an", "with", "here", "under, "nose", "her", "Attack"]
// call the previous decode function
```

Steps

Example for “even-backwards”

```
var sentence = "fill the proper tank for the cow";
// we are taking only the even words
// "fill proper for cow"
// reverse them ["cow", "for", "proper", "fill"]
// call the previous decode function
```

Steps

file super_decoder.js

```
var decoder = require("./decoder.js");

function super_decoder(sentence, type) {
   var types = type.split("-");
   var words = [];
   if (types[0] === "every") {
      words = sentence.split(" ");
   }
   ...

}
```

Remember `.filter()` to select from an array

Steps

Call the decoder with the new array to get the `secretMessage`.

Steps

Return the `secretMessage`

Steps

file super_decoder.js

```
var decoder = require("./decoder.js");

function super_decoder(sentence, type, forwards ) {
    var forwards = forwards || false;
    var words = [];
    if (type === "every") {
        words = sentence.split(" ");
    }
    ...

    if (forwards) {
        return decoder(words);
    }
}
```

Exercise

Download the list of sentences from the gist and discover the secret message!

*Possibilities: forwards or backwards, every, even or odd words. And their combinations. First check for every, even and odd words. Then check backwards or forwards

* Bonus: create a function that checks all the possibilities automatically with all the list of words.

Solution

Solution