Notes

Contents

1 Notes

1.1 Functions

```
// Calculate the volume of a prism using parameters length, width, and height
var prism = function(l,w,h)
{
    return l * w * h;
}

// Print the result of the prism function with specific values
console.log(prism(34,23,2));

// Define a curried function prisma to calculate the volume of a prism
var prisma = function (l){
    return function(w){
        return function(h){
            return l * w * h;
        }
    }
}
// Print the result of the prisma function with curried values
console.log(prisma(20)(12)(12));

// Immediately-invoked function expression (IIFE) to log a message
var foo = (function(){
    console.log("Here I am");
}());

// Constant variable bar storing the result of an IIFE
const bar = (function() {
    return 56;
}());

// Print the value of the constant variable bar
console.log(bar)
```

1.2 Named Function

```
// Define a named function sum and an anonymous function sumaEnoy for addition
var suma = function sum (a,b){
  return a + b;
}
var sumaEnoy = function (a,b){
  return a + b;
}

// Print the result of the named function sum and the anonymous function sumaEnoy
console.log(suma(1,2));
console.log(sumaEnoy(1,2));

// Attempt to call the undefined function sum (commented out)
// console.log(sum(1,2)); //(Is not Undefinated)

// Attempt to call the undefined function foo
foo();
// Define foo as an anonymous function and call it
var foo = function (){
    console.log("example");
}

// Call the function foo successfully
foo();

// Declare function foo and call it before its declaration
function foo (){
    console.log("example");
}

// Define a function say and call it recursively with decreasing times argument
var say = function say(times){
    if(times > 0 ){
        say = undefined;
        console.log("hello");
        say(times - 1 );
    }
}

// Assign the say function to saysay, then try to call the undefined say function
var saysay = say; //say is not a function
say = "opps";
saysay(10);
```

1.3 Variadic Functions

```
// Define a variadic function persons to log messages for a person
function persons(per, ...msg){
  msg.forEach(arg => {
      console.log(per + " say: " + arg);
  })
}

// Call the persons function with various arguments including an anonymous function
obj = {
  username : "bar",
  status: true
}
persons("fer","hello","world","js","React", ((x)=> x * x)(2))

// Define and call an anonymous function foo
var foo = function(){
  console.log("example");
}
foo();

// Define a function foo with a callArg parameter, call recursively with true
var foo = function(callArg){
  console.log("messages");

  if(callArg == true) foo(false)
}
foo(true)

// Define a function printMsg with a default parameter msg
function printMsg(msg ='default messages'){
  console.log(msg);
}

// Call printMsg with different arguments
printMsg("example")
printMsg(undefined)
printMsg(null)
printMsg()
```

1.4 Fetch Structure Examples

```
// Fetch data from the PokeAPI and log the JSON response
fetch('https://pokeapi.co/api/v2/pokemon/ditto')
.then(resp => resp.json())
.then(json => console.log(json))
.catch(err => console.log(err) )

// Set request headers in the fetch request
fetch('',{
    headers: new Headers({
        'Accept': "text/plain",
        'x-your-custom-header':  'example.values'
    })
})

// Fetch data from JSONPlaceholder and log details from the response
var url ='https://jsonplaceholder.typicode.com/posts';
fetch(url)
.then(response => response.json())
.then(response => {
    response.forEach(element => {
        console.log("ID" + element.id + " -- Title " + element.title );
    });
});

// Fetch data from JSONPlaceholder albums and log details
var url = 'https://jsonplaceholder.typicode.com/albums'
fetch(url)
.then(res => res.json())
.then(res => res.forEach(element => {
    console.log("userId "+ element.userId + "--- id "+ element.id+ "--- title " + elem
}))

// Create an unordered list element in the HTML body
const questionList = document.createElement('ul');
document.body.appendChild(questionList);

// Fetch data and process the response to access specific properties
const responseData = fetch(url).then(response => response.json());
responseData.then(({items, has_more, quota_max, quota_remaining}) => {
    for (const {title, score, owner, link, answer_count} of items) {
    }
});
```

1.5 Axios

```
// Require the Axios library
const axios = require('axios');

// Define a URL for Axios to fetch data
const url = "https://jsonplaceholder.typicode.com/users";

// Fetch data using Axios and log each element in the response
axios.get(url).then(res =>
    res.data.forEach(element => {
        console.log(element);
    })
)

// Perform a POST request with Axios, sending data and logging the response
axios.post(url,
    {
        username:"foo bar",
        email: "foo.bar.com"
    }
).then(res => console.log(res.data));
```