# Advance JavaScript

## 1) Write the code, one line for each action:

### a) Create an empty object user.

~ let user = {};

### b) Add the property name with the value John.

~ user.name = “john”;

### c) Add the property surname with the value Smith.

~ user.surname = “smith”;

### d) Change the value of the name to Pete.

~ user.name = “pete”;

### e) Remove the property name from the object.

~ delete user.name

## 2) Is the array copied?

```
let fruits = ["Apples", "Pear", "Orange"]; // push a new value into
the "copy" let shoppingCart = fruits;
shoppingCart.push("Banana"); // what's in fruits?

alert( fruits.length ); // ?
```

```
~ let fruits = ["Apples", "Pear", "Orange"];
  alert (fruits [0] ) ;// Apples
```

```
   alert (fruits [1] ) ;// Pear
  alert (fruits [2] ) ;//  Orange

Fruits [3] Banana;// Apples,Pear,Orange,Banana

alert(fruits.length);//3
```

## 3) Map to names let john = { name: "John", age: 25 }; let pete = { name: "Pete", age: 30 }; let mary = { name: "Mary", age: 28 }; let users = [ john, pete, mary ]; let names = /* ... your code */ alert( names ); // John, Pete, Mary

```
~ let john = { name: john, age: 25};
   let pete = { name: pete, age:30};
   let mary = { name: mary, age:28};

  let users = [ john,pete,mary ] ;
  let names =  user.map (item => item.name);
  alert (names); //john,pete,mary
```

## 4) Map to objects

```
 let john = { name: "John", surname: "Smith", id: 1 };
 let pete = { name: "Pete", surname: "Hunt", id: 2 };
 let mary = { name: "Mary", surname: "Key", id: 3 }; let users = [ john, pete, mary ]; let usersMapped = /* ... your code ... */

/*
 usersMapped = [
 { fullName: "John Smith", id: 1 },
 { fullName: "Pete Hunt", id: 2 },
 { fullName: "Mary Key", id: 3 } ]
```

```
 */ alert( usersMapped[0].id ) // 1 alert( usersMapped[0].fullName ) //
John Smith

~  let john = { name: "John", surname: "Smith", id: 1 };
   let pete = { name: "Pete", surname: "Hunt", id: 2 };
   let mary = { name: "Mary", surname: "Key", id: 3 };

   let users =  [ john, pete , mery ];
   let userMapped = user.map ( user => ({
       fullName:`${ user.name} $ {user.surname}`,
       id : user.id
}));

/* users mapped = [
  { fullName: “john smith”, id :1},
  { fullName:  "Pete Hunt", id: 2 },
  { fullName: "Mary Key", id: 3 } ]
*/

alert ( usersMapped[0].id) ;//1
alert ( usersMapped[0].fullName) ;// john smith.
```

**5) Sum the properties There is a salaries object with an arbitrary number of salaries. Write the function sumSalaries(salaries) that returns the sum of all salaries using Object.values and the for..of loop.If salaries is empty, then the result must be 0.**

```
let salaries =
 { "John": 100, "
Pete": 300, "
Mary": 250
 };
alert( sumSalaries(salaries) ); // 650
```

```
~ function sumSalaries (salaries) {
    let sum = 0 ;
    for ( let salary of object . values (salaries)) {
    sum+= salary ;}

    return sum ; // 650
   }
   let salaries = {
    “John” : 100;
    “pete”  : 300;
    “Mary” : 250;
};
alert(sumSalaries(salaries)   ); //650
```

**6) Destructuring assignment We have an object: Write the Destructuring assignment that reads:**

**a) Name property into the variable name.**
**b) Year’s property into the variable age.**
**c) isAdmin property into the variable isAdmin (false, if no such property)**
**d) let user = { name: "John", years: 30};**

```
~ let user = {
     name: “john”,
     year : “30”
};
let { name, years : age, isAdmin = false} = user;

alert (name);// john
alert (age) ; // 30
```

```
alert( isAdmin); // false
```

**7) Write a program to Show an alert**

~ 
```
<script>
function ShowAlert(){
 alert (“hello World !”);
 }
 </script>
```

**8) Will an alert be shown?**

```
if ("0") { alert( 'Hello'); }
```

~ yes, any string except an empty one (and “0” is not empty) becomes true in the logical context. We can run and check.

```
If (“0”) {
alert (‘hello’);
}
```

**9) What is the code below going to output?**

```
alert( null || 2 || undefined );
```

~ The answer is 2 , that is the first truthy value.

```
alert( null || 2 || undefined );
```

**10) The following function returns true if the parameter age is greater than**

**18. Otherwise it asks for a confirmation and returns its result:**

```
function
checkAge(age)
        {
       if (age> 18) { return true;
        }
      else {
      // ...return confirm (‘did parents allow you?'); } }
```

```
~ function CheckAge(Age){
   If (age > 18 ) {
    return  true;
   } else {
    return confirm (‘ Did present allow you ? ‘);
    }
    }
```

## 11)  Replace Function Expressions with arrow functions in the code below: Function ask(question, yes, no)

```
~  function ask (question , yes , no ) {
    if ( confirm (question) ) (yes);
    else no () ;
    }
    ask (
    “Do you agree?”),
      () => alert (“you agreed .”),
      () => alert (“ you canceled the execution.”)
      );
```

## 12) What is JSON

**~  JSON stands for JavaScript Object Notation**

**JSON is a lightweight format for storing and transporting data**

**JSON is often used when data is sent from a server to a web page**

**JSON is "self-describing" and easy to understand**

## 13) What is promises

**~  A promise is a proxy for a value not necessarily known when the promise is created. it  allows you  to associate handlers with an asynchronous action's eventual success value or failure reason. This lets asynchronous methods return values like synchronous methods: instead of immediately returning the final**

**value, the asynchronous method returns a *promise* to supply the value at some point in the future.**

## 14) Write a program of promises and handle that promises also

```
~    new promise ((resolveOuter) => {

      resolveOuter (

      new promise ((resolveInner) => {

     setTimeout (resolveInner , 1000);

  });

);

});
```

## 15) Use fetch method for calling an api https://fakestoreapi.com/products

```
~   fetch ('https:// fakestoreapi.com/products/1')

     .then(res => res.json () );

    .then (json => console.log(json) );
```

## 16)  Display all the product from the api in your HTML page

**~  Javascript is a powerful programming language that is consitently used with HTML in web programming. One of the many things that you can do with javascript is that you can Fetch API Results from their respective endpoints. In this walkthrough , you are going to learn how to extract Rick  and morty data from an API and dispaly it on an HTML page.**