Javascript Notes by anshumancodes

Variables
Now lets understand variables this way , Variable is like a container or jar in which you can store anything ! like a jar can store liquid , solid or anything else. A variable can store different type of data types.

Variable keywords

Var keyword

var is the keyword that tells JavaScript you're declaring a variable.
Var variables can be re-declared ,updated

This means that we can do this within the same scope and won't get an error.

```
var greeter = "hey hi";
var greeter = "say Hello instead";
```

and this also

```
var greeter = "hey hi";
greeter = "say Hello instead";
```

Pic source:freecodecamp.org

The problem with var keyword is it can be over written causing errors while writing 100s lines of code.

Let keyword
Let is now preferred for variable declaration. It's no surprise as it comes as an improvement to `var` declarations. It also solves the problem with `var` that we just covered. Let's consider why this is so.

let can be updated but not re-declared.

Just like `var`, a variable declared with `let` can be updated within its scope. Unlike `var`, a `let` variable cannot be re-declared within its scope. So while this will work:

```
let greeting = "say Hi";
greeting = "say Hello instead";
```

this will return an error:

```
let greeting = "say Hi";
let greeting = "say Hello instead"; // error: Identifier 'greeting' has already been declare
```

Credit: freecodecamp.org

However, if the same variable is defined in different scopes, there will be no error:

```
let greeting = "say Hi";
if (true) {
    let greeting = "say Hello instead";
    console.log(greeting); // "say Hello instead"
}
console.log(greeting); // "say Hi"
```

Why is there no error? This is because both instances are treated as different variables since they have different scopes.

Const

Variables declared with the `const` maintain constant values. `const` declarations share some similarities with `let` declarations.

Like `let` declarations, `const` declarations can only be accessed within the block they were declared.

const cannot be updated or re-declared

This means that the value of a variable declared with `const` remains the same within its scope. It cannot be updated or re-declared. So if we declare a variable with `const`, we can neither do this:

```
const greeting = "say Hi";
greeting = "say Hello instead";// error: Assignment to constant variable.
```

Read more about const , let , var here

Data types:

- Primitive
- reference types

Primitive Types

- String
- Numbers
- Boolean
- Undefined
- Null

Types of language

- Statically-typed - declared value can't be changed
- Dynamically-typed- declared value can be changed

Javascript is a dynamic lang i.e; values assigned can be changed in runtime!

```
> typeof name
< 'string'
> name=1256
< 1256
> typeof name
< 'number'
> //this shows that value once assigned can be changed in runtime
< undefined
```

Reference types

- Array
- object
- function

Objects:

JavaScript variables can also contain many values. Objects are variables too. But objects can contain many values.

Here is a example

```
> let empX={name:"Anshuman",Age:34,Role:"frontend dev",salary:150000};
< undefined
> // example of a object , here empX is a object that have multiple value assigned
< undefined
> //so lets check what empX returns
< undefined
> empX
< ▼{name: 'Anshuman', Age: 34, Role: 'frontend dev', salary: 150000} i
    Age: 34
    Role: "frontend dev"
    name: "Anshuman"
    salary: 150000
  ▶[[Prototype]]: Object
> typeof empX
< 'object'
```

But what if i want just one value or property from object?
Here is how you can do that by using the dot notation method!
{check next page for code snippet}

```
> //How to change or read value of a single property from a object
< undefined
> //so lets consider a example Employee
< undefined
> let employee={name:"Anshuman",Age:34,Role:"frontend dev",salary:150000};
< undefined
> // lets read every single property
< undefined
> employee.name
< 'Anshuman'
> // As you can see it returned the name assigned
< undefined
> // Now lets try to change the name property
< undefined
> employee.name="Rohan"
< 'Rohan'
> //lets check the value of name again
< undefined
> employee.name
< 'Rohan'
> // as you can see this time it returned "Rohan" insted of predefined value"Anshuman"
< undefined
> //this method is called dot notation
< undefined
>
```

Now lets do the same by using bracket notation method!

```
> // again we will take our old example employee
< undefined
> let employee={name:"Anshuman",Age:34,Role:"frontend dev",salary:150000};
< undefined
> employee['Age']
< 34
> //with this method we will use sqr bracket with a quotation
< undefined
> //alright let change the value again with bracket notation method
< undefined
> employee['Age']="69"
< '69'
> // lets read again
< undefined
> employee['Age']
< '69'
> //it returned 69 insted of 34 ! misson passed++
< undefined
```

Arrays in javaScript:

the array is a single variable that is used to store different elements. It is often used when we want to store a list of elements and access them by a single variable.

Here is how to use , read , manipulate a array(basic)

```
> // lets make a array
< undefined
> let BgColors=['red','pink','yellow','green']
< undefined
> // so here BgColors is a array now lets read and manupulate it
< undefined
> // syntax for reading or using any value of a array is arrayNmae[index]
< undefined
> BgColors[0]
< 'red'
> BgColors[3]
< 'green'
> // *index starts from 0
< undefined
> // lets manupulate the array
< undefined
> BgColors[2]="grey"
< 'grey'
> // now if we use 2nd index we will get:
< undefined
> BgColors[2]
< 'grey'
> // we got grey not yellow
```

Lets do some advanced stuff-

```
> // Lets do some advanced concepts

  let BgColors=[['red','pink','yellow','green'],['darkyellow','greypink','black']]

  // alright so in this array we have 2 non-identical lists inside our array

  // lets read this
< undefined
> BgColors[0][1]
< 'pink'
> // syntax : Array[list][value]
< undefined
> // how to add a value into a array in runtime
< undefined
> BgColors[0][4]="darkgreen"
< 'darkgreen'
> // now lets check if we inserted it right or not
< undefined
> BgColors[0][4]
< 'darkgreen'
> // yo sucess!
< undefined
```

There are lot of array methods that you will need while writing code in js you can learn about them here

Functions

A JavaScript function is a block of code designed to perform a particular Task.

syntax:

```
Function name(params){
 Your logic

};
name(params);
```

Lets see some examples to understand how functions work

```
// lets see another example of function

function sayHelloto(name) {
    console.log("hello " + name)
}
sayHelloto("Anshumancodes")
```

Output:

```
hello Anshumancodes                                    index.js:27
> |
```

Here sayHelloto() is the function name while “Anshumancodes” is the given argument to the params name.

Functions are reusable and thats the reason they are highly used

Lets see that with a example

```
function sayHelloto(name) {
    console.log("hello " + name)
}

sayHelloto("stephen")
sayHelloto("rahulcodes")
```

Output:

```
hello stephen                                          index.js:27
hello rahulcodes                                       index.js:27
> |
```

Also a function can have multiple parameters lets understand that by:

```
function sayhello(firstName , lastName){
    console.log("hello "+ firstName+ lastName)
};
undefined
// lets input values
undefined
sayhello("anshuman")
hello anshumanundefined                                    VM13702:2
undefined
// hello anshumanundefined is the ouput because we didnt inserted a value for lastName
undefined
// now lets do both
undefined
sayhello("anshuman","codes")
hello anshumancodes                                        VM13702:2
undefined
// sucesss!
undefined
```

---------- end ----------

Learn more about javascript here (link to MDN docs)