Variables:

we used variable to store data temporarily.
we can look about it like box which store some data and we label on it to recognize it

Naming Variables rules:

- they cannot be reserved keyword
- they should be meaningful
- cannot start with a number
- they cannot contain a space or hyphen
- if the name have 2 words you need to use camel case
- they case-sensitive mean that: firstName and Firstname isn't the same variables
- if you need undefined variables you can place them at same row with comma
- semmi colon = ;, comma = ,

we have two kinds of variable:
const and let
const cannot reassign if we try to change const throw the code we will have an error
and let is flexible

Primitive Types:

in Javascript we have two categories of types:

- Reference Types
- Primitives / Value Types which are: String, Number, Boolean, undefined and null

var name = ‘Mosh’; // String Literal

var age = 30; // Number Literal

var isApproved = true; // Boolean Literal the value of Boolean can be true or false

var firstName; // Undefined

var selectedColor = null; // we used null when we want to clear the value of the variable
console.log(typeof selectedColor) = Object

Dynamic Typing:

Javascript is dynamic language
we have two type of languages Static and Dynamic

in static language when we declare a variable the type of this variable cannot be chnaged in the future

in Dynamic language the type of the variable can change at run time.

var name = ‘Asaf’
console.log(typeof name) = string
name = 1;
console.log(typeof name) = number

Reference Types:

Reference Types are Object Array and Function

Object in Javascript its like object in the real life think of person:
have name, age, and address and more:

var person = {}; // object literal
Object can include properties

var person = {‘asaf’, 28}

console.log(person) = { ‘asaf’, 28}

var person = {}; // Object literal
Object can include keys: values

var person = {
name: ‘Asaf’
age: 38
};

console.log(person) = {name: ‘Asaf’, age:28}

there is two ways to reach object data:

Dot Notation - person.name // Dot notation is your defualt because it's shorter
person.name = ‘Asaf Yeshoa’ // will change the value of the key ‘name’ from Asaf to Asaf yeshoa

Bracket Notation - when we don't know the name of the target property
person[name] = ‘Asafi’ will change the value of the key ‘name’ from Asaf to Asafi
if user might be selecting the name of the target property in this case we don't know what property we going to access so we will use another variable may called

var selection = ‘name’;
person[selection] = ‘Mary’;

Arrays

example to empty Array:

var selectedColors = []; // Array Literal

initialize array:

var selectedColors = [red, blue]
console.log(selectedColors) = 2 [‘red’, ‘blue’] // this is array with 2 elements
note that every element have an index. // red = index 0, blue = index 1
index determines the position of the element in that array

console.log(selectedColors[0]) = red
console.log (selectedColors[1]) = blue

to add element to the array will look like that:

selectedColor[2] = ‘green’
console.log(selectedColor) =[‘red’, ‘blue’, ‘green’]

remember Javascript is dynamic language so we can also do that

selectedColor[2] = 1
console.log(selectedColor) = [‘red’, ‘green’, 1]

typeof seletectedColor = object

to know the number of elements inside the array or the object we use the data structure call LENGTH

console.log(selectedColors.length) = 3

Functions

function is set of statements that either performs a task or calculates and return a value.

application is a collection of hundreds or thousands of functions working together to provide the functionality of that application

function are one of the fundamental building blocks in JavaScript.
a function is basically a set of statements that performs a task or casualties a value.
inside the function you have statement that terminate it.

example:

```
function hello () {
console.log('hello world')
}

hello();  = hello world (in the console)
```

function have parameter which make more useful // function hello (PARAMETER)

parameter is like variable that meaningful only to the function. parameter won't be accessible outside of the function

```
function hello(name) {          // now name is an input to this function
console.log('hello' + name)
}

hello('asaf');  = hello asaf (in the console)
hello('asafi');  = hello asafi (in the console)
hello('asafush');  = hello asafush (in the console)
```

// outside of the function the parameter will called argument

function can have multiple parameters that separate by comma.
if function that have multiple parameters wont have 2 arguments when you call the fuction the result will be undefined

```
function hello(name, lastName) {
console.log('hello' + name + lastName)
}

hello('asaf');  = undefined

hello('asaf', 'yeshoa') = hello asafyeshoa

function hello(name, lastName) {
console.log('hello' + name + '  ' +lastName)       // using space in the statement
}

hello('asaf', 'yeshoa') = hello asaf yeshoa
```

Type Of Functions:

there is few kinds of functions:

Performing a task

this task is to display something
example:

```
function hello(name, lastName) {
console.log('hello' + name + lastName)
}
```

another one is Calculating a value

this function is calculated something and **return** the function to how call the function
example:

```
function square(number) {
return number * number
}

square(2);
```

// note that if you wanna see the result of the function in console in need enter her to a var

```
var displayConsle = square(2);    // this particular example isn't nassery to use console
console.log(displayConsle)        // console.log(square(2)) we will give the same result
```

good to know that " console.log(displayConsle) " include 2 function log() and square()

JavsScript Operrators

in Javascript we have different kinds of operators.

we use these operators along with our variables to create expressions.

with these expression we can implement logic and algorithms.

we have type of operators:

- arithmetic
- assignment
- comparison
- logical
- bitwise

Arithmetic Operators

we use this for performing calculation just like calculations in mathematics
example:

```
var x = 10;
var y = 5;

console.log(x + y);  = 15   //  (x +y) is an expression and + is the operators
console.log(x - y)         // -  is the operators
console.log(x * y)    //  * is the operators
console.log(x / y)      //   /  is the operators
console.log(x % y)    //  % is the operators
console.log(x ** y)    //  // **  is the operators
```

we have two additional arithmetic operators that a little bit more tricky.

increment(++) and decrement(--)

increment example:

x++ = x + 1

```
var x = 5;
console.log(x) = 5
```

if the increment coming before the variable it will raise him by 1

```
console.log(++x); = 6
```

if we put the increment after x the value of the variable will change after printing to console

```
var x = 5;

console.log(x++)   =  5
console.log(x)      =  6
```

decrement example:

x-- = x +1

```
var x = 5;
console.log(x) = 5
```

if i use decrement coming before the variable it will decrease him by 1

```
var x = 5;
console.log(--x) // 4
```

if we put the decrement after x the value of the variable will change after printing to console

```
var x = 5;
console.log(x--)  = 5
console.log(x)    = 4
```

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

Assignment Operators

we using the assignment operator to assign a value.

var x = 10; // the assignment operator is =

there few more ways of Assignment operators:

- x += 5; // is equal to x = x + 5
- x -= 5; // is equal to x = x - 5
- x *= 5; // is equal to x = x * 5
- x /= 5; // is equal to x = x / 5
- x %= 5; // is equal to x = x % 5

Comparison Operators

we use there operators to compare the value of a variable with something else.
Comparison operators return boolean True or False

console.log(x > 0); = true

here we wanna check if x is bigger than 0 and because x = 1 we have get true
the result of comparison operator is boolean (mean true or false).

another comparison operator is >=.
var x = 5

here we try to check if (x >=1)
console.log(x >= 1) = true // because the value of x is 5 he bigger than 1 and it's true

its comparison operators call relational operators
more option to relational operators are:

- console.log(x < 5) // these expression return false
- console.log(x <= 5) // these expression return true

another comparison operator are Equality operators that help check if the value of the variable is equal in the expression we try to check

few equality operators:

- x === 5 /// return true. here we check if x is equal to 5
- x !== 5 // return false. here we check if x NOT equal to 5 throw using !

noted equality operator with === call strict equality he check if the value and type are the same for example:

var x = 5 // the number 5
var y = '5' // String of 5
var z = 5 // the number 5

so x === y will return false
and x === z will return true

Logical Operators

we use these operators to make decisions based on multiple conditions.
in javascript we have three types of logical operators:

- Logical AND (&&) // return TRUE if both operands are TRUE
  console.log(true && true) // return true
  console.log(true && false) // return false
- Logical OR (||) // return TRUE if one of the operands is TRUE
  console.log(true || true) // return true
  console.log(true || false) // return true
  console.log(false || false) // return false
  console.log(false || true) // return true

  if Logical OR will return value before false example:
  console.log(false || 1) // return 1
  console.log (false || 'Asaf') return Asaf

- Logical NOT(!) // return the upside of the expression boolean value
  console.log(!true) // return false
  console.log(!false) // return true

If and Else

if and else helping as to build condinal system code on our app that apply the right code block while considering of the condition

Basic structure:

```
if (condition) {    // if the condition is true we will executed the statement
 statement          // statement = block of code
}
else if (anotherCondition) {      // we can have many conditions as we want
 statement
}
else                // if none of the conditions above isn't true JS will execute the else statement
 statement
```

Example:

```
// if hour is between 6am and 12 pm: good morning
    // if it is between 12pm and 6pm: good afternoon
    // if otherwise: good evening

    var hour = 1;

    if (hour >= 6 && hour < 12){
        console.log('Good morning')
    } else if (hour >= 12 && hour <= 18){
        console.log('Good Afternoon')
    } else {
        console.log('Good evening')
    }
```

Switch and case

switch and case help smillier to if and else just using variable instead of condition

Basic structure:

```
var forExample;

switch (variable){                  // the variable working with switch is allway compare with var
 case ‘user’:                       // first case if the variable is equal with user
 console.log(‘Guest User’);         // can be any statement
 break;                             // break sperted between cases block


 case ‘admin’:                      // we can have as many statements that we want
 console.log(‘Admin User’);         // can be any statement
break;

default:                            // if none of this cases are matched
console.log(Unknown User’);           // can be any statement
                                    // default don't need break
}
```

Loops

when we want to repeat an action a number of times we using loops.
in Javascript we have few kinds of loop and all of them doing the same thing they repeat an action a number of times. the difference between the loops is when they start and end

For

the loop will run while the condition is true

Basic structure of for loop:

```
for (initialExpression; condition; incrementExpression) {
 statements

}
```

example:

```
for (var i = 0; i < 5; i++){       // the loop will run until I equal to 5 and i will raise each loop by 1
console.log('Hello world')

}
```

loop also can count down by changing the condition and the increment
for (var i = 5; i > 0; i--)
we using this case if we want reverse the the result

While loop

one key difference between a while loop and for loop is that in for loops.
in while loops you have to declare this while variable externally

Basic structure of while loop:

```
var i = 0

while (condition){
statement
increment
}
```

```
var i = 0

while (i <= 5){
if (i % 2 !== 0) console.log(i)
i++
}
```