SOFT CLUB

JAVA SCRIPT

LECTURE 1

Whoa!

If you are seeing this post, it means you have been struggled a lot and finally succeeded

GETTING STARTED WITH JAVASCRIPT

JavaScript is a **popular programming** language that has a wide range of applications.

JavaScript was previously used mainly for making webpages interactive such as form validation, animation, etc. Nowadays, JavaScript is also used in many other areas such as server-side development, mobile app development and so on.

JAVASCRIPT IS ALSO USED IN MANY OTHER AREAS

SOFT CLUB

Front-end apps

Dynamic effects and web applications in the browser 😍

THIS COURSE

JS

100% based on JavaScript. They might go away, **but JavaScript won't!**

Back-end apps

Web applications on web servers

JS

node JS

Native mobile applications

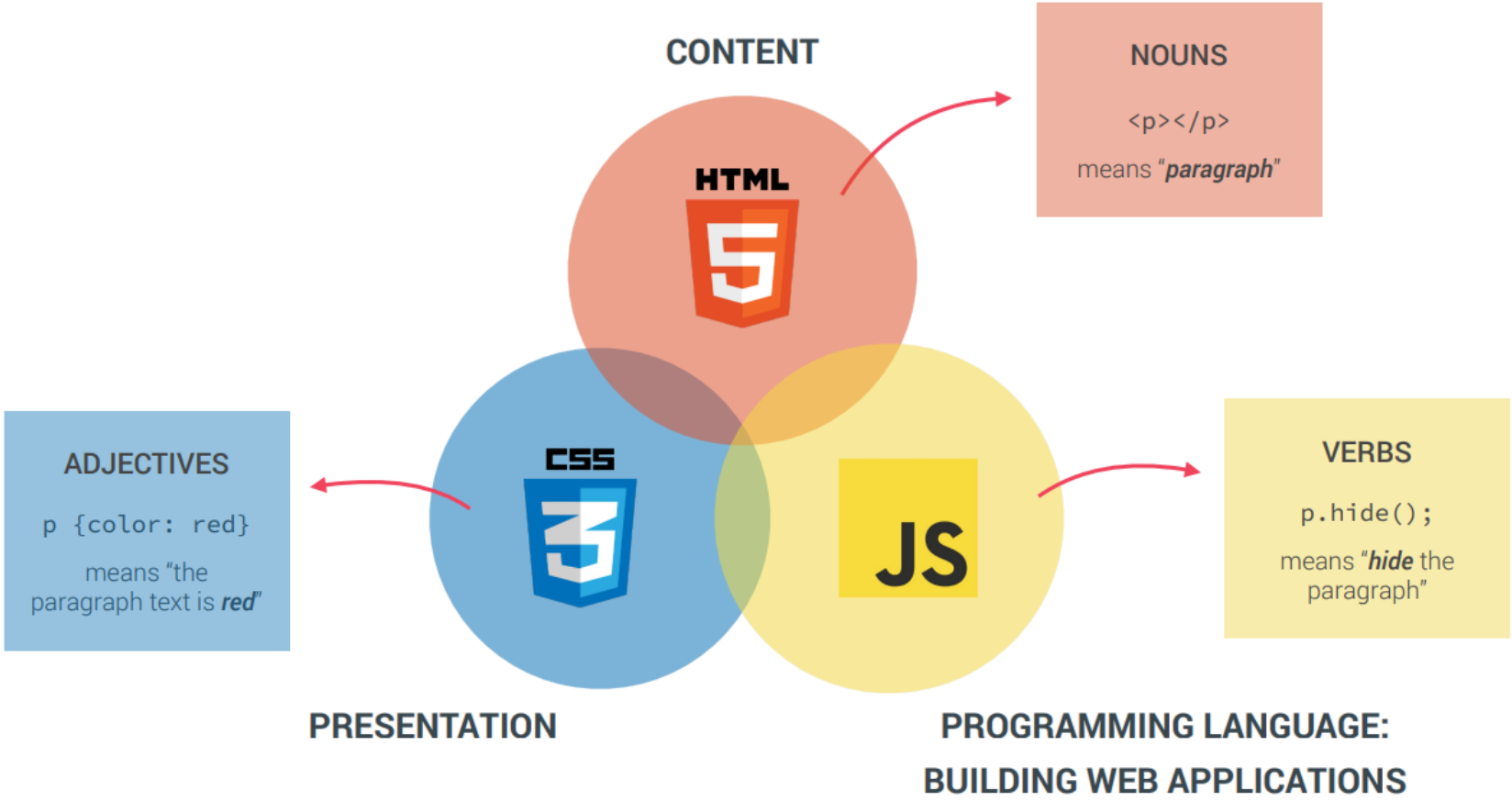
JS

IONIC

Native desktop applications

JS

THE ROLE OF JAVASCRIPT IN WEB DEVELOPMENT



A BRIEF HISTORY OF JAVASCRIPT

SOFT CLUB

1995

- 👉 Brendan Eich creates the **very first version of JavaScript in just 10 days**. It was called Mocha, but already had many fundamental features of modern JavaScript!

1996

- 👉 Mocha changes to LiveScript and then to JavaScript, in order to attract Java developers. However, **JavaScript has almost nothing to do with Java** ☝️
- 👉 Microsoft launches IE, **copying JavaScript from Netscape** and calling it JScript;

1997

- 👉 With a need to standardize the language, ECMA releases ECMAScript 1 (ES1), the first **official standard for JavaScript** (ECMAScript is the standard, JavaScript the language in practice);

⋮

2009

- 👉 ES5 (ECMAScript 5) is released with lots of great new features;

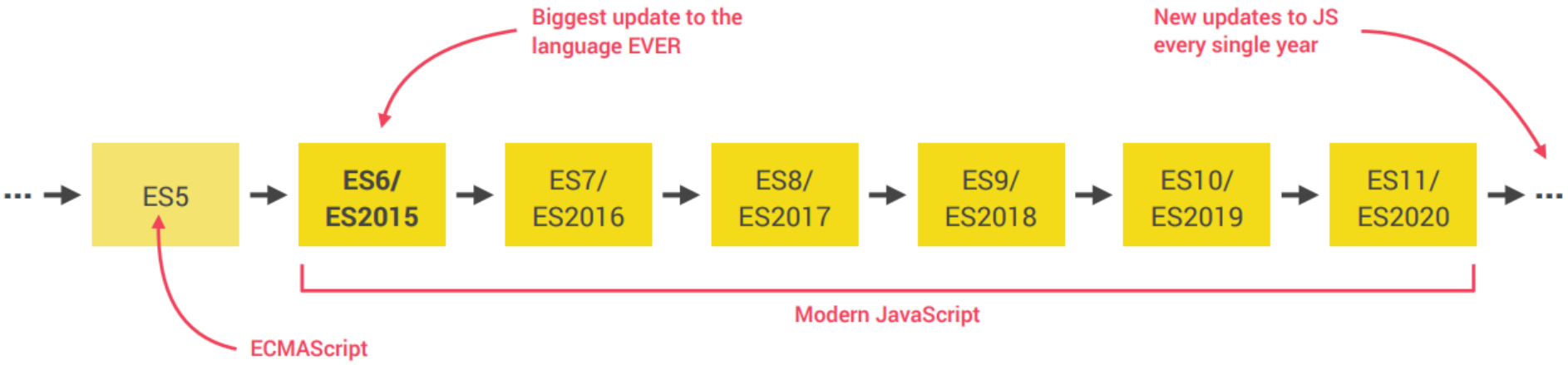
2015

- 👉 ES6/ES2015 (ECMAScript 2015) was released: **the biggest update to the language ever!**
- 👉 ECMAScript changes to an **annual release cycle** in order to ship less features per update 🙏

2016 – ∞

- 👉 Release of ES2016 / ES2017 / ES2018 / ES2019 / ES2020 / ES2021 / … / ES2089 😅

JAVASCRIPT RELEASES



The ECMAScript specification is a standardized specification of a scripting language developed by **Brendan Eich** of **Netscape** initially named **Mocha**, then **LiveScript**, and finally **JavaScript**.
Learn modern JavaScript from the beginning, but without forgetting the older parts!

RUN JAVASCRIPT

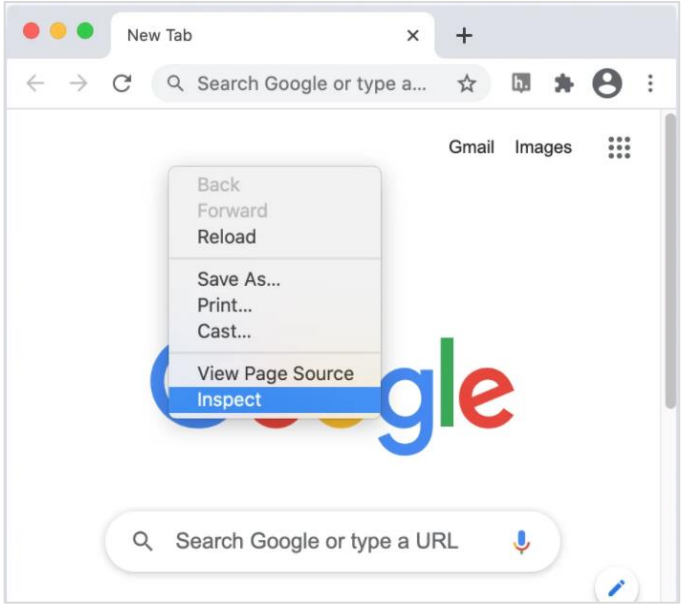
Because of its wide range of applications, you can run JavaScript in several ways:

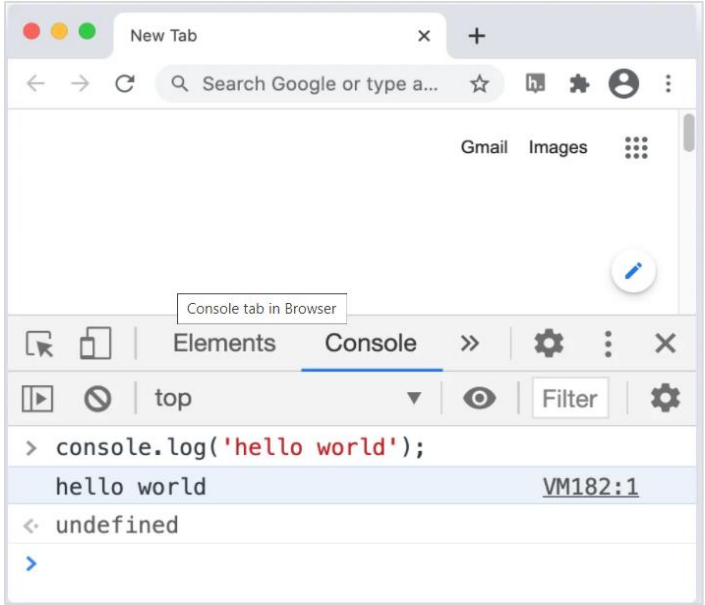
- **Using console tab of web browsers**
- **Using Node.js**
- **By creating web pages**

1. USING CONSOLE TAB OF WEB BROWSERS

1. Open your browser and right click in any empty area and select **inspect** or press **F12.**

2. Open the **developer tools** and go to **Console** tab. Write the Javascript code and press **Enter**





2. USING NODE.JS

Node is a back-end environment for executing JavaScript code. To run JS using Node.js, follow these steps:

1. Install the latest version of Node.js.
2. Open Visual studio code and create js file
3. Run node hello.js
4. See the result

hello.js U ×

hello.js

```
1   console.log("Hey Welcome");
```

TERMINAL OUTPUT PROBLEMS DEBUG CONSOLE

```
PS C:\Users\admin\Desktop\JsTest> node .\hello.js
Hey Welcome
PS C:\Users\admin\Desktop\JsTest>
```

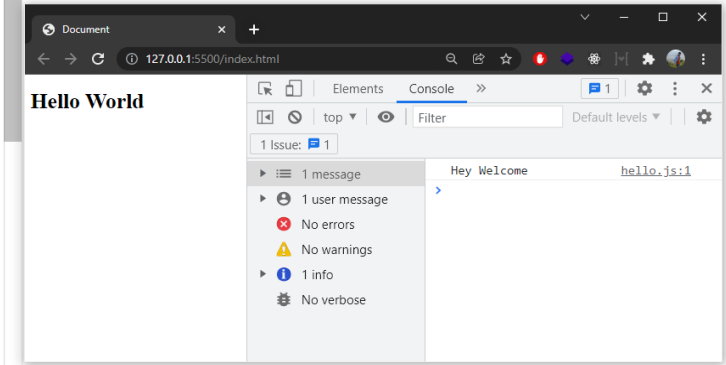
3. BY CREATING WEB PAGES

index.html U

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
     initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello World</h1>

    <script src="hello.js"></script>
</body>
</html>
```

hello.js U

```js
console.log("Hey Welcome");
```



JavaScript was initially created to make web pages interactive, that's why JavaScript and HTML go hand in hand. To run JS from a webpage, follow these steps:

1. Create new folder
2. Create index.html, script.js files
3. Write shown line of codes

JAVASCRIPT VARIABLES AND CONSTANTS

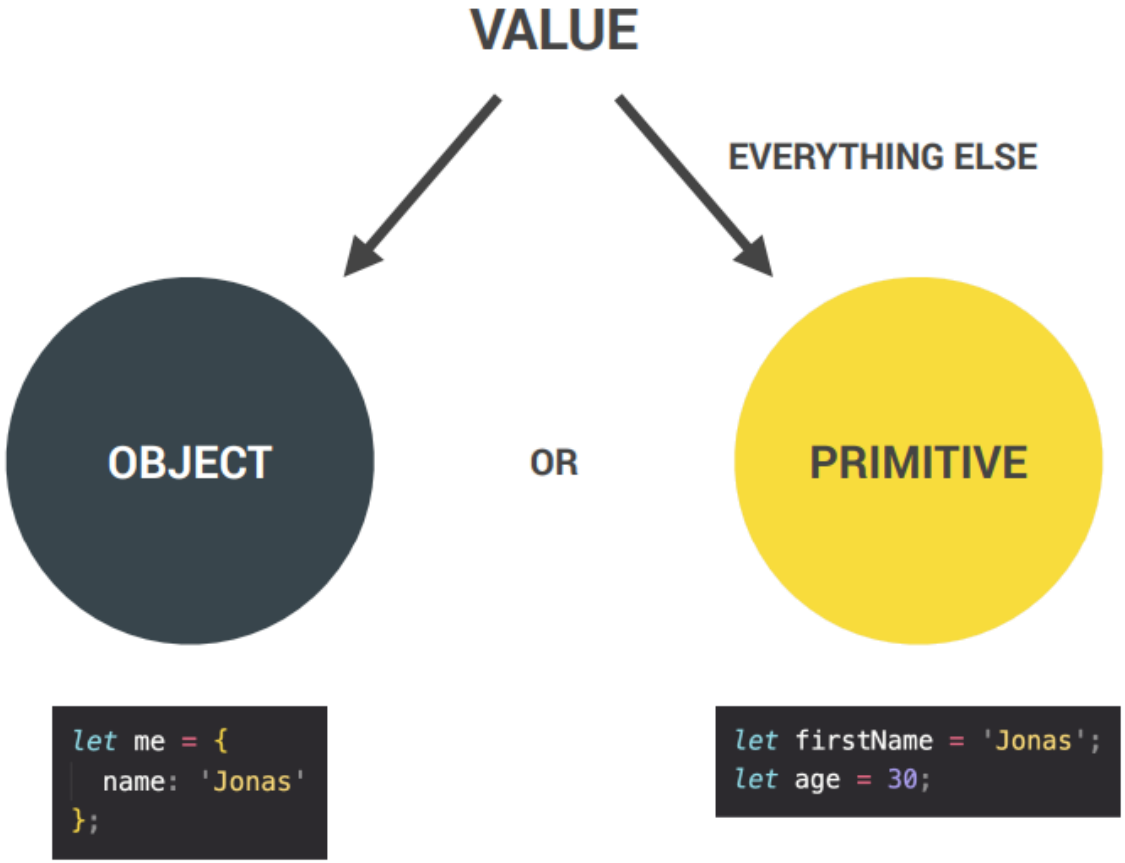
SOFT CLUB

In programming, a variable is a container *(storage area)* to hold data.

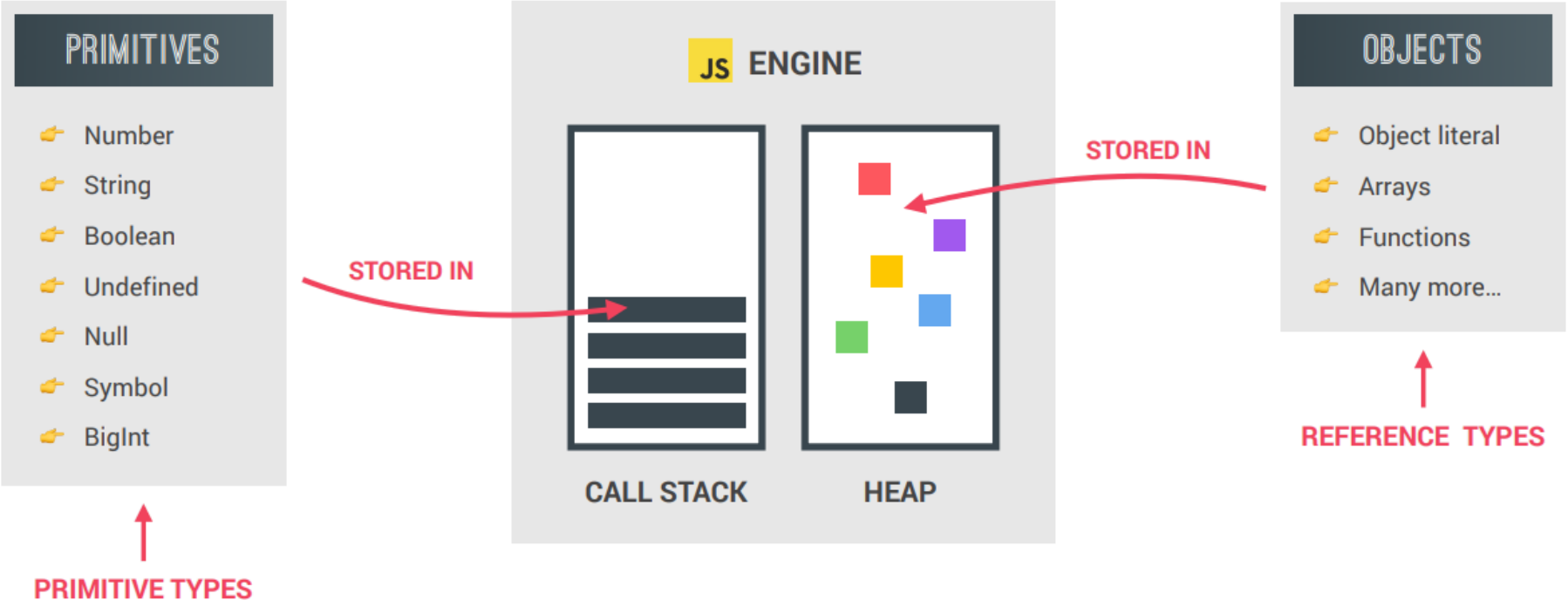
In Javascript there is two types of intializing variables, **var** and **let.** You can use both of them. However, there are some differences between them.
If you are sure that the value of a variable won't change throughout the program, it's recommended to use **const** .

var	let	const
`var` is used in the older versions of JavaScript	`let` is the new way of declaring variables starting **ES6 (ES2015).**	`const` keyword indicates that the value of a variable is a constant **ES6(ES2015).**
`var` is function scoped (will be discussed in later tutorials).	`let` is block scoped (will be discussed in later tutorials).	`const` scope is defined as '**block scoped**'
For example, `var x;`	For example, `let y;`	For example, `const n`

OBJECTS AND PRIMITIVES



OBJECTS AND PRIMITIVES



THE 7 PRIMITIVE DATA TYPES

SOFT CLUB

1. **Number:** Floating point numbers 👉 Used for decimals and integers `let age = 23;`
2. **String:** Sequence of characters 👉 Used for text `let firstName = 'Jonas';`
3. **Boolean:** Logical type that can only be `true` or `false` 👉 Used for taking decisions `let fullAge = true;`
4. **Undefined:** Value taken by a variable that is not yet defined ('empty value') `let children;`
5. **Null:** Also means 'empty value'
6. **Symbol (ES2015):** Value that is unique and cannot be changed *[Not useful for now]*
7. **BigInt (ES2020):** Larger integers than the Number type can hold

JavaScript has dynamic typing: We do ***not*** have to manually define the data type of the value stored in a variable. Instead, data types are determined **automatically**.

Value has type, NOT variable!

RULES FOR NAMING JAVASCRIPT VARIABLES

SOFT CLUB

1. Variable names must start with either a letter, an underscore _, or the dollar sign $.
2. Variable names cannot start with numbers. For example:

```
let 1simpleText = 'Javascript is really simple';
console.log(1simpleText);
```

JS hello.js U ×

JS hello.js > ...

```
1  //valid
2  let text = 'Javascript is really simple';
3  let _text = 'No pain no gain';
4  let $text = 'you can do it';
5  console.log(text);
6  console.log(_text);
7  console.log($text)
```

TERMINAL OUTPUT PROBLEMS DEBUG CONSOLE

```
PS C:\Users\admin\Desktop\JsTest> node .\hello.js
Javascript is really simple
No pain no gain
you can do it
PS C:\Users\admin\Desktop\JsTest>
```

OPERATORS IN JAVASCRIPT

SOFT CLUB

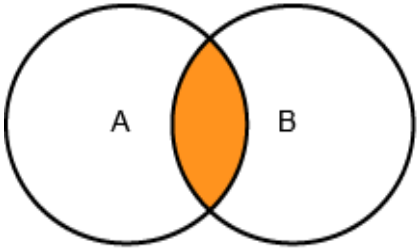
Name	Operators
Arithmetic	+ , - , * , /
Comparison	==, ===, >=, <= ,!= ,!==
Logical	|| , && , !
Type Conversions	Number("3.14")
Assignment	= , +=, -=, *=, /=,^= ,%=

JAVASCRIPT ARITHMETIC OPERATORS

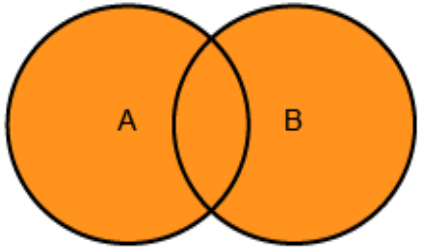
Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1
+ +	Increment	A = 10; A+ +	11
− −	Decrement	A = 10; A− −	9

LOGICAL OPERATORS : AND(&&), OR(| |), NOT(!)

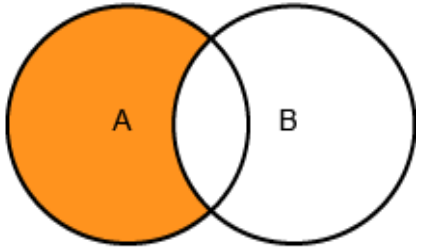
Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
||	Logical or	(5<2)||(5>3)	True
!	Logical not	!(5<2)	True



A AND B



A OR B



A NOT B

COMPARISON OPERATORS

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
! =	Not equal to	5! =2	True
===	Equal value and same type	5 === 5	True
		5 === “5”	False
! ==	Not Equal value or Not same type	5 ! == 5	False
		5 ! == “5”	True

ASSIGNMENT OPERATORS

Operator	Example	Equivalent Expression
$=$	$m = 10$	$m = 10$
$+=$	$m += 10$	$m = m + 10$
$-=$	$m -= 10$	$m = m - 10$
$*=$	$m *= 10$	$m = m * 10$
$/=$	$m /=$	$m = m/10$
$\% =$	$m \% = 10$	$m = m\%10$

JAVASCRIPT TYPE CONVERSIONS

SOFT CLUB

There are two types of type conversion in JavaScript

1. Implicit Type Conversion
2. Explicit Type Conversion

hello.js U ×

hello.js > ...

```javascript
// numeric string used with + gives string type

//Example 1: Implicit Conversion to String

var result;
result = '3' + 2;
console.log(result) // "32"
result = '3' + true;
console.log(result); // "3true"
result = '3' + undefined;
console.log(result); // "3undefined"
result = '3' + null;
console.log(result); // "3null
```

TERMINAL OUTPUT PROBLEMS DEBUG CONSOLE

```
PS C:\Users\admin\Desktop\JsTest> node .\hello.js
32
3true
3undefined
3null
PS C:\Users\admin\Desktop\JsTest>
```

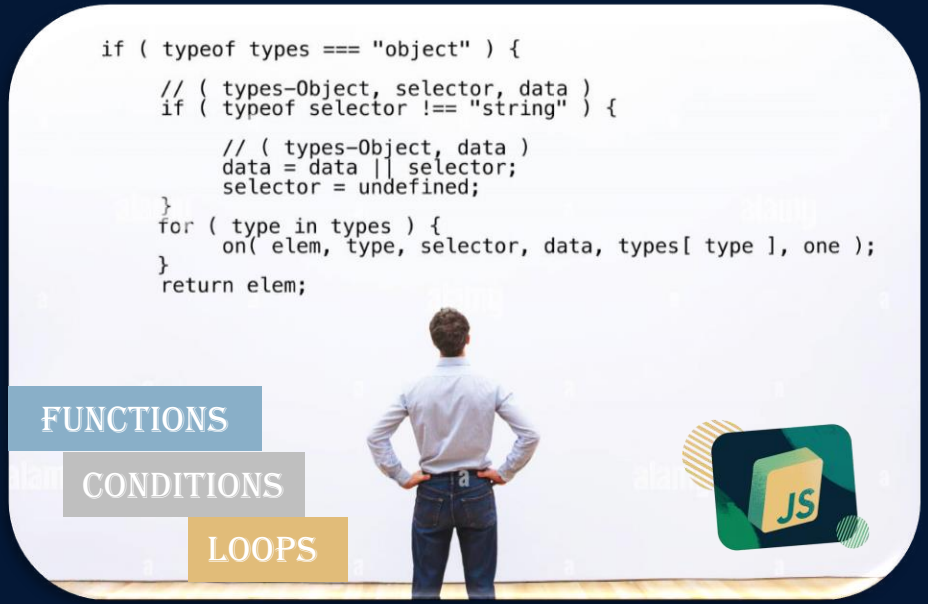
THREE IMPORTANT TOPICS

SOFT CLUB

01 CONDITIONS

02 LOOPS

03 FUNCTIONS



CONDITION IF/ELSE STATEMENT

SOFT CLUB

1st Condition is true

```
let number = 2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

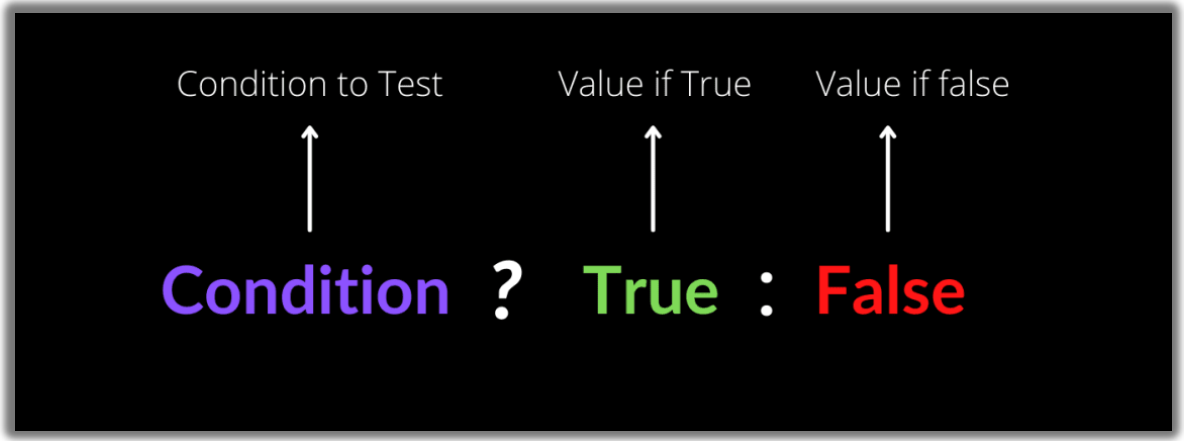
2nd Condition is true

```
let number = 0;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

All Conditions are false

```
let number = -2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

CONDITION TERNARY OPERATOR



CONDITION SWITCH STATMENT

The switch statement evaluates an expression, matching the expression's value against a series of case clauses, and executes statements after the first case clause with a matching value, until a break statement is encountered. The default clause of a switch statement will be jumped to if no case matches the expression's value.

```
const expr = 'Papayas';

switch (expr) {
  case 'Oranges':
     console.log('Oranges are $0.59 a pound.');
      break;
  case 'Papayas':
     console.log('Mangoes and papayas are $2.79 a pound.');
     // Expected output: "Mangoes and papayas are $2.79 a pound."
      break;
  default:
    console.log(`Sorry, we are out of ${expr}.`);
}
```

LOOP FOR

The for statement creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement (usually a block statement) to be executed in the loop.
The following for statement starts by declaring the variable i and initializing it to 0. It checks that i is less than nine, performs the two succeeding statements, and increments i by 1 after each pass through the loop.

```
let str = '';
let cnt=0

for (let i = 0; i < 9; i++) {
  str = str + i;
  cnt+=i
}

console.log(str);
// Expected output: "012345678"
console.log(cnt);
// Expected output: 36
```

LOOP WHILE

The while statement creates a loop that executes a specified statement as long as the test condition evaluates to true. The condition is evaluated before executing the statement.

The following while loop iterates as long as n is less than three.

Note: Use the break statement to stop a loop before condition evaluates to true.

```
let n = 0;
let x = 0;

while (n < 3) {
  n++;
  x += n;
}

console.log(n);
// Expected output: 3
console.log(x);
// Expected output: 6
```

LOOP DO/WHILE

The do...while statement creates a loop that executes a specified statement until the test condition evaluates to false. The condition is evaluated after executing the statement, resulting in the specified statement executing at least once.

In the following example, the do...while loop iterates at least once and reiterates until i is no longer less than 5.

```
let result = '';
let i = 0;

do {
  i = i + 1;
  result = result + i;
} while (i < 5);

console.log(result);
// Expected output: "12345"
```

FUNCTIONS

SOFT CLUB

There are 3 ways of writing a function in JavaScript

1. Function declaration
2. Function Expression
   - Arrow function
   - Anonymous function
3. Immediately Invoked Function Expression (IIFE)

FUNCTION DECLARATION

The **function declaration** defines a function with the specified parameters. **A function is declared using the function keyword.**
The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.

```
function calcRectArea(width, height) {
    return width * height;
  }

  console.log(calcRectArea(5, 6));
   // Expected output: 30

  console.log(calcRectArea(10, 20));
  // Expected output: 200
```

FUNCTION EXPRESSION

A **function expression** is very similar to and has almost the same syntax as a function declaration.
The main difference between a function expression and a function declaration is the *function name*, which can be omitted in function expressions to create **anonymous** *and* **arrow** functions.

```
// anonymous function
let anonymous=function(parametr){
    return parametr
}
anonymous("hi")
// arrow function
let arrow=(parametr)=>{
    return parametr
}
console.log(arrow("hi")
)
```

FUNCTION IIFE

An **IIFE** (Immediately Invoked Function Expression) is a function that runs the moment it is invoked or called in the JavaScript event loop.
Having a function that behaves that way can be useful in certain situations. IIFEs prevent pollution of the global JS scope.

```
var firstName="John";

(function(a,b){
    var firstName="Doe"
    console.log(firstName) // Doe
})()
console.log(firstName)     // John
```

FUNCTION REVIEW: ANATOMY OF A FUNCTION

SOFT CLUB

Function name

Parameters: placeholders to receive **input** values. Like local variables of a function

Function body: block of code that we want to reuse. Processes the function's input data

```
function calcAge(birthYear, firstName) {
  const age = 2037 - birthYear;
  console.log(`${firstName} is ${age} years old`);
  return age;
}

const age = calcAge(1991, 'Jonas')
```

return statement to **output** a value from the function and terminate execution

Calling, running or invoking the function, using ( )

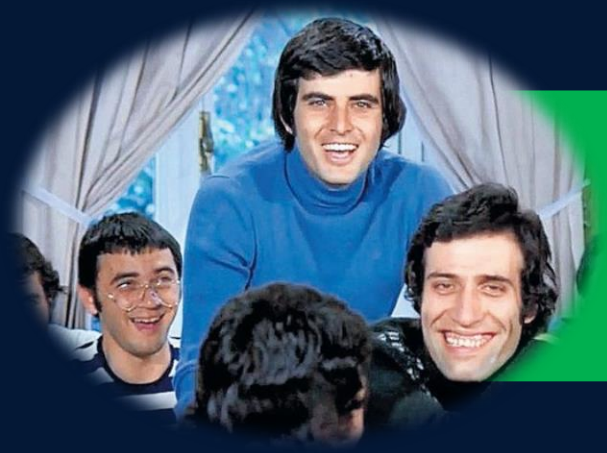
Variable to save returned value (function output)

Arguments: actual values of function parameters, to **input** data

SOFT CLUB

Thanks!

Be happy and Smile



THE END

LECTURE 1