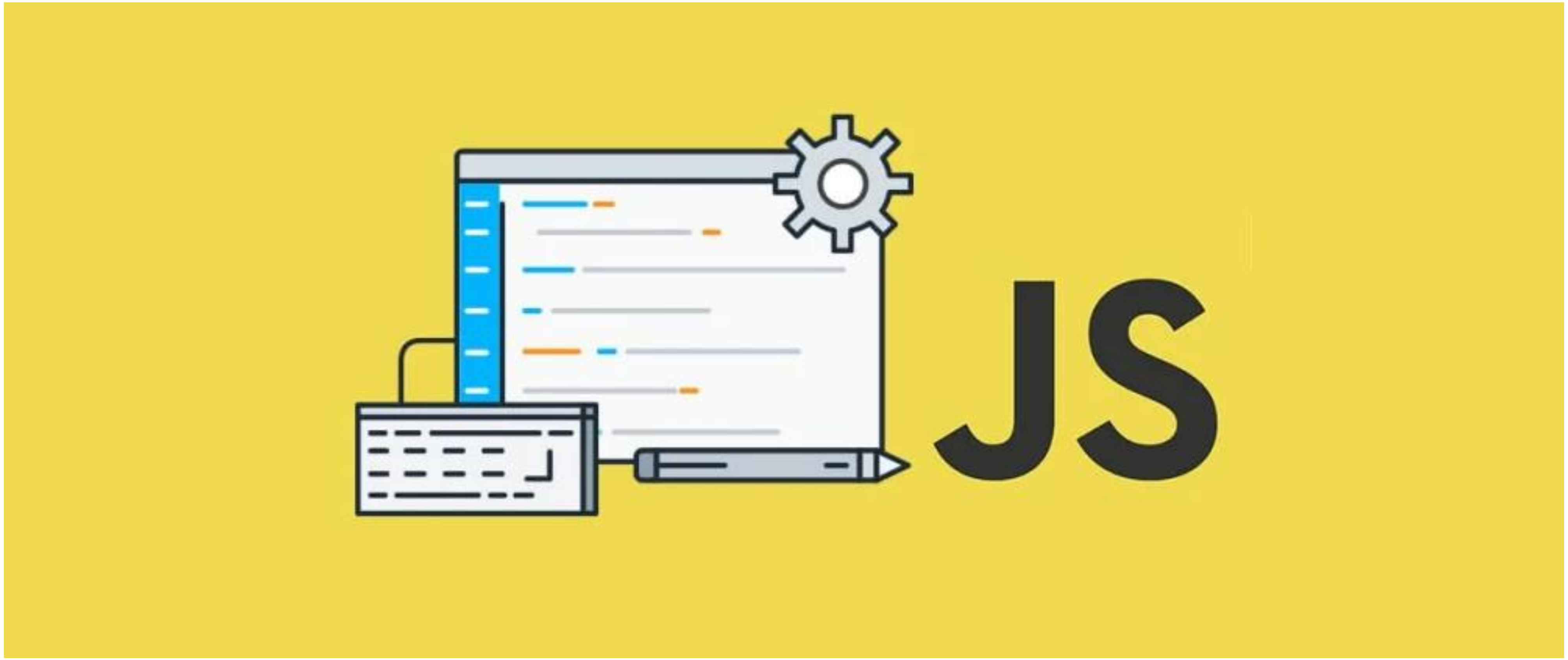
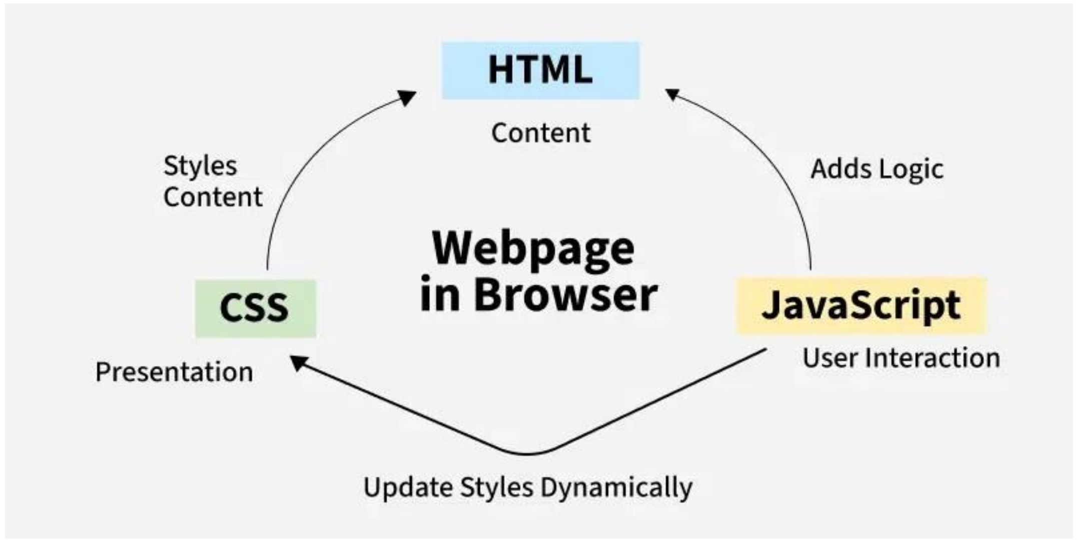
JavaScript for the React



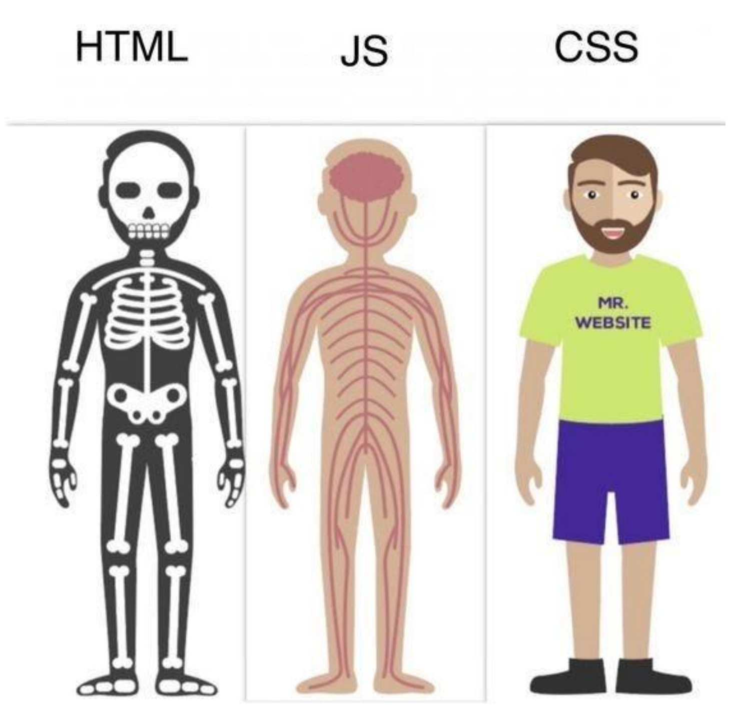
Introduction to JavaScript

- JavaScript is a **programming language** used to create dynamic content for websites. It is a **lightweight**, **cross-platform,** and **single-threaded** programming language. JavaScript is an **interpreted** language that executes code line by line providing more flexibility.



HTML v/s CSS v/s JavaScript

- HTML is used to make a website structure CSS is used to design(paint) that structure while JavaScript is used to add behavior. HTML is a markup language for creating web pages. JavaScript is used for DOM manipulation. Using JavaScript, we can handle date and time.



Console Log, Variables & Data Types

- In JavaScript, the **console.log()** method displays messages or variables in the browser's console.

Example: `console.log("Hello world!");`

- In JavaScript, a **variable is a named storage location that holds a value**, which can be any data type, such as numbers, strings, objects, etc. It can be declared using keywords like var, let, or const.
- JavaScript Variables can be declared in 4 ways: **1) Automatically, 2) Using var, 3) Using let, 4) Using const**
- **Examples:**

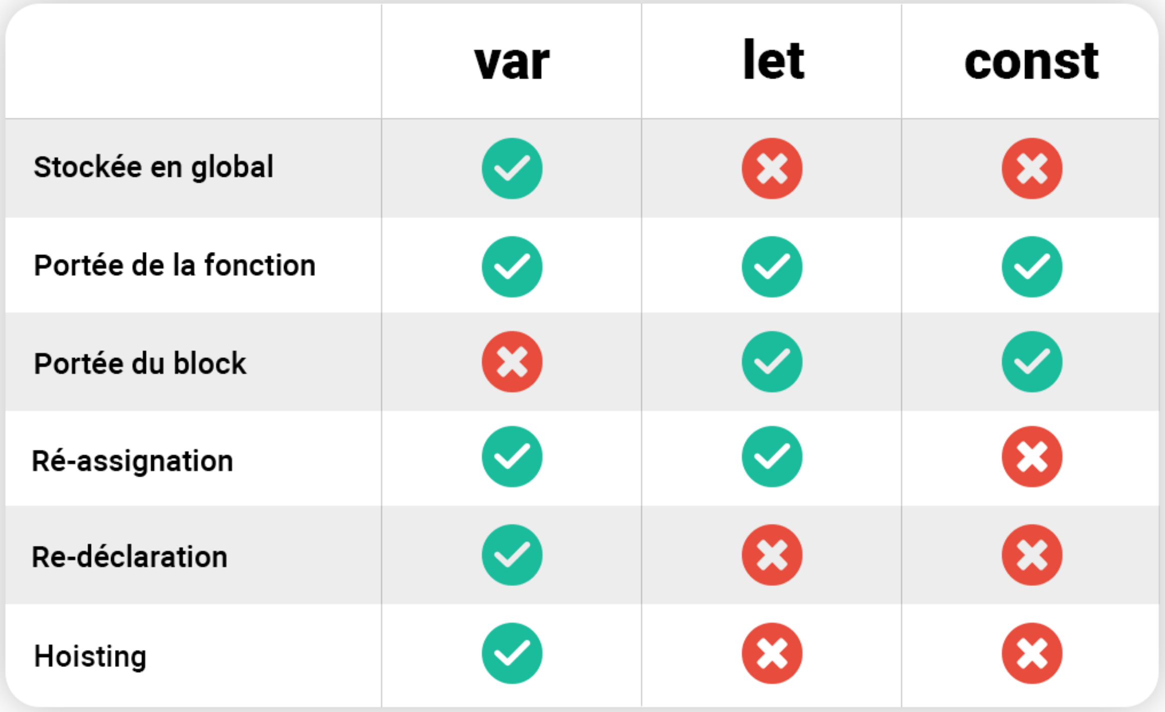
  x = 5; , y = 6; , z = x + y; (Automatically)

  var x = 5; , var y = 6; , var z = x + y; (Using var)

  let x = 5; , let y = 6; , let z = x + y; (Using let)

  const x = 5; , const y = 6; , const z = x + y; (Using const)

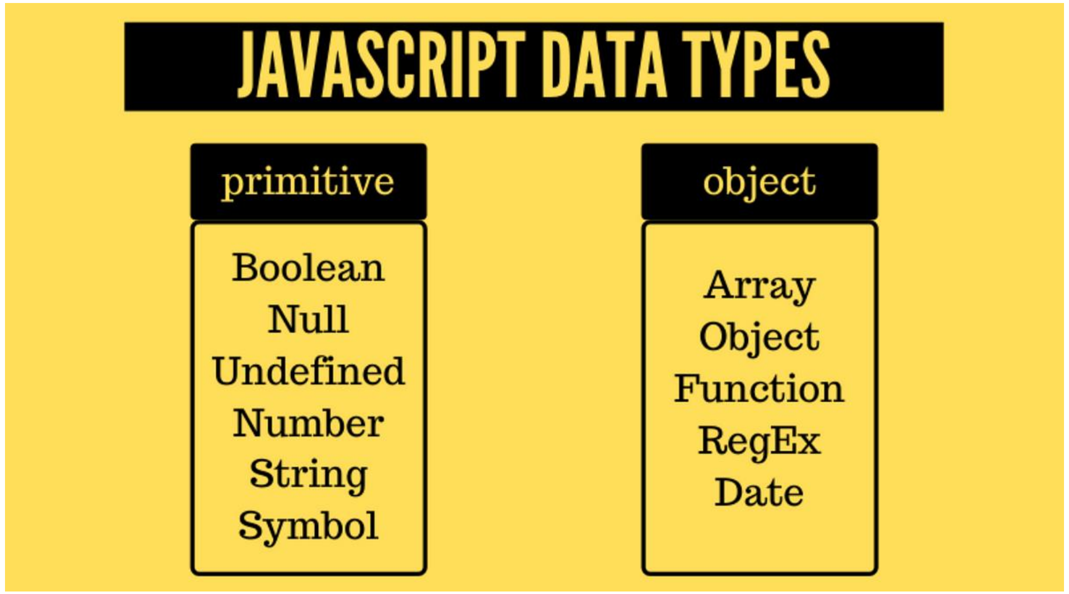
Const v/s Var v/s Let



	var	let	const
Stockée en global	✓	✗	✗
Portée de la fonction	✓	✓	✓
Portée du block	✗	✓	✓
Ré-assignation	✓	✓	✗
Re-déclaration	✓	✗	✗
Hoisting	✓	✗	✗

Data Types

- In JavaScript, data types describe the different types or kinds of data that we're gonna be working with and storing in the variables. There are some basic data types such as strings, numbers, Booleans, or undefined, and null.
- There are two types of variables in JavaScript-data type and user-defined. So, in sum, there exist five data types in JavaScript which are described in the following.



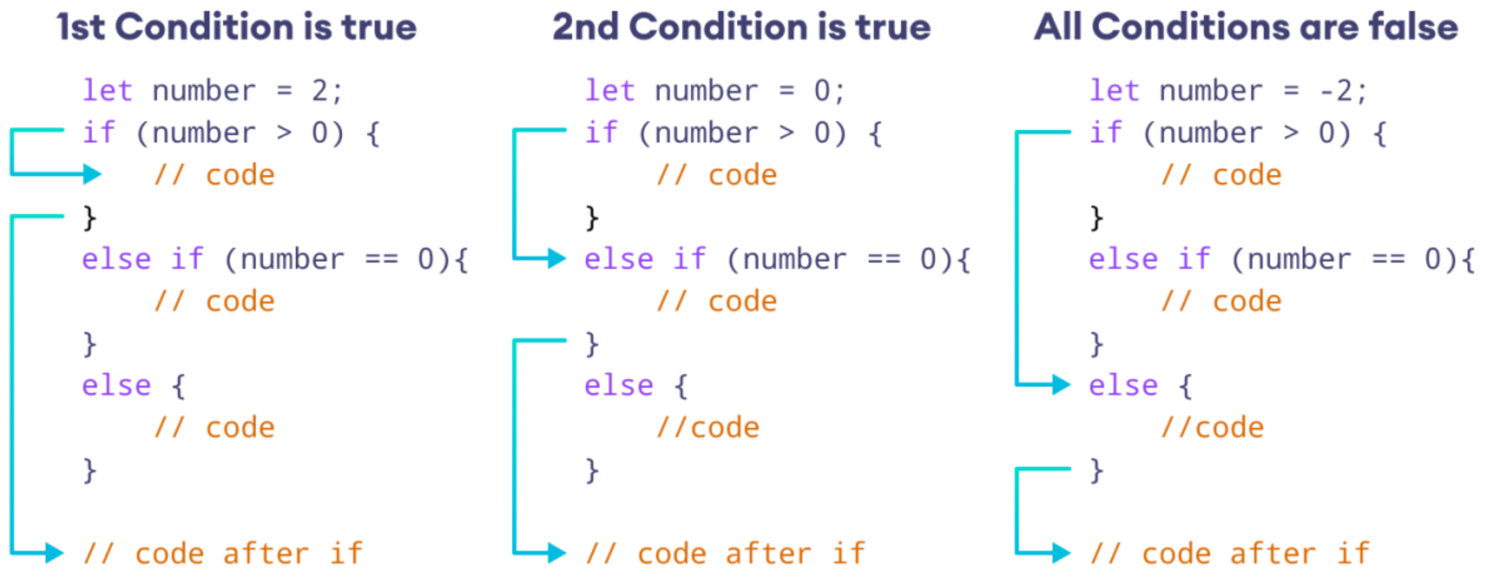
Operators in JavaScript

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical “and” evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical “or” evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical “not” evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	“A”+”BC”	ABC

Conditional Statements

- In JavaScript we have the following conditional statements:

1. Use **if** to specify a block of code to be executed, if a specified condition is true
2. Use **else** to specify a block of code to be executed, if the same condition is false
3. Use **else if** to specify a new condition to test, if the first condition is false
4. Use **switch** to specify many alternative blocks of code to be executed



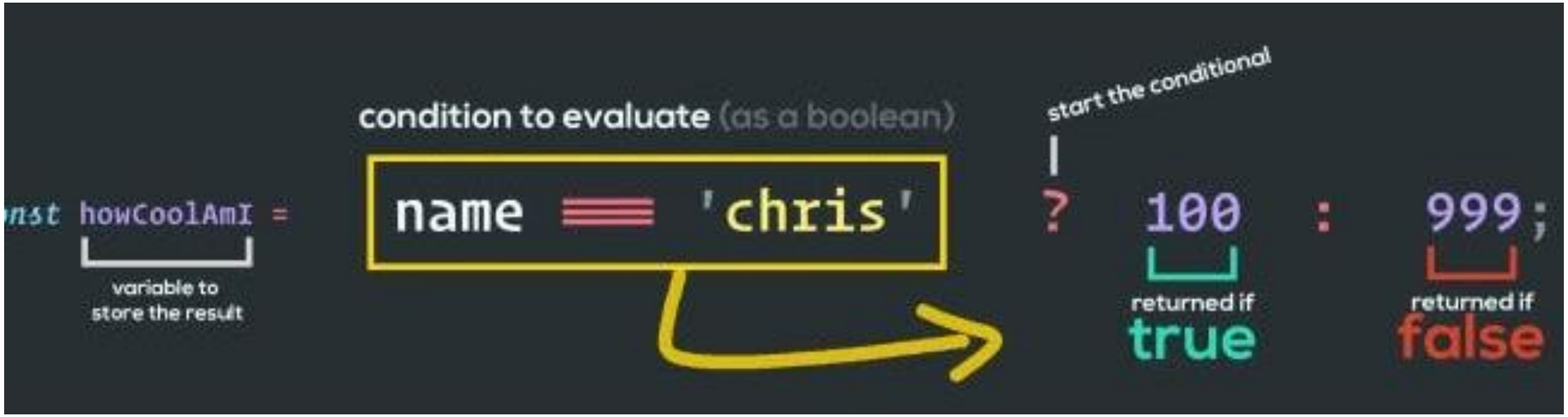
```
let number = 2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    // code
}

// code after if
```

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

// code after if
```

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

// code after if
```

Ternary Operator

- A ternary operator can be used to replace an if..else statement in certain situations. Before you learn about ternary operators, be sure to check the JavaScript if...else tutorial.
- A ternary operator evaluates a condition and executes a block of code based on the condition.
- Syntax: condition ? expression1 : expression2



```
nst howCoolAmI = name === 'chris' ? 100 : 999;
```

condition to evaluate (as a boolean)
start the conditional
variable to store the result
returned if true
returned if false

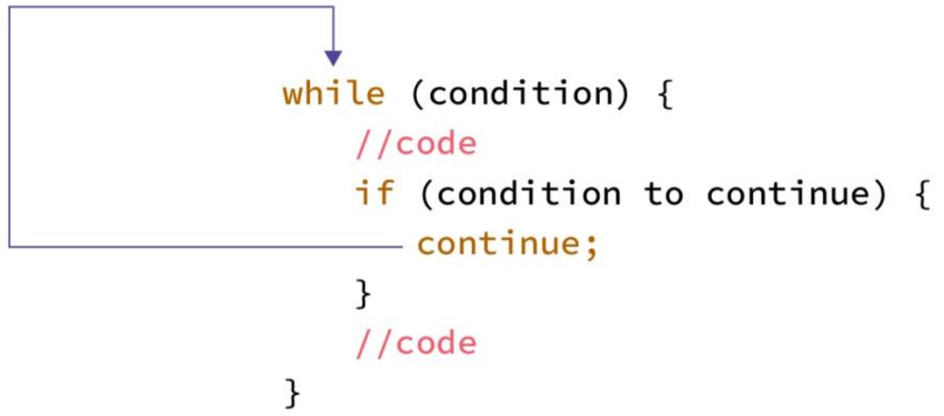
Loops in JavaScript

- In JavaScript, a loop is a programming tool that is used to repeat a set of instructions. Loops are used to reduce repetitive tasks by repeatedly executing a block of code as long as a specified condition is true.
- Loops in JavaScript make the code more concise and efficient. The loops are used to iterate the piece of code using for, while, do-while, or for-in loops.
- There are several types of loops present in JavaScript. Such as:

1. JavaScript for Loop
2. JavaScript while Loop
3. JavaScript do-while Loop
4. JavaScript for…of Loop
5. JavaScript for…in Loop

For loop & While loop

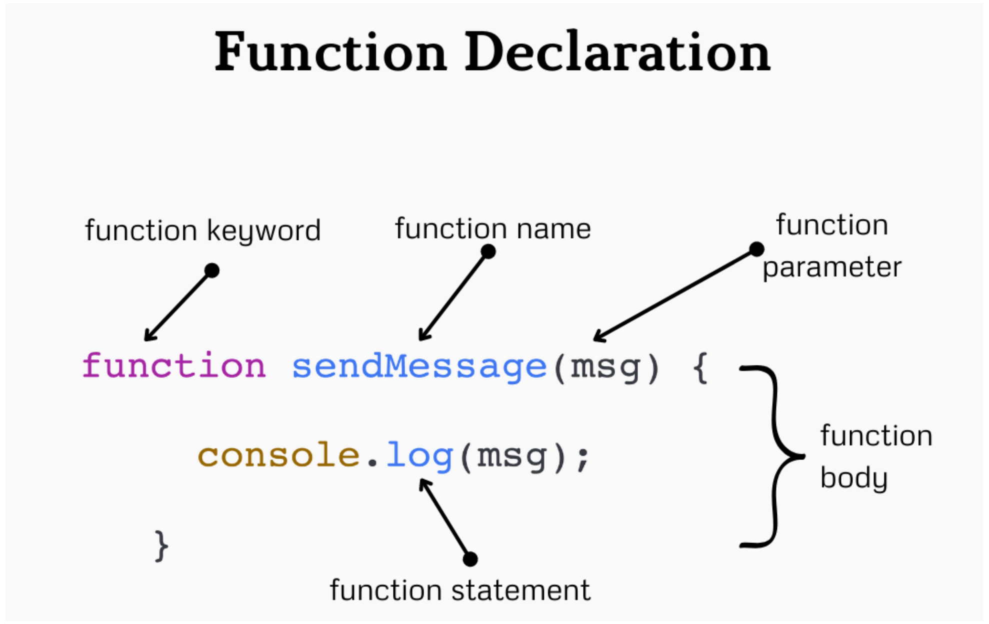
```
for (start_point; condition; step) {

  // code to execute

}
```



```
while (condition) {
    //code
    if (condition to continue) {
      continue;
    }
    //code
}
```

Functions in JavaScript

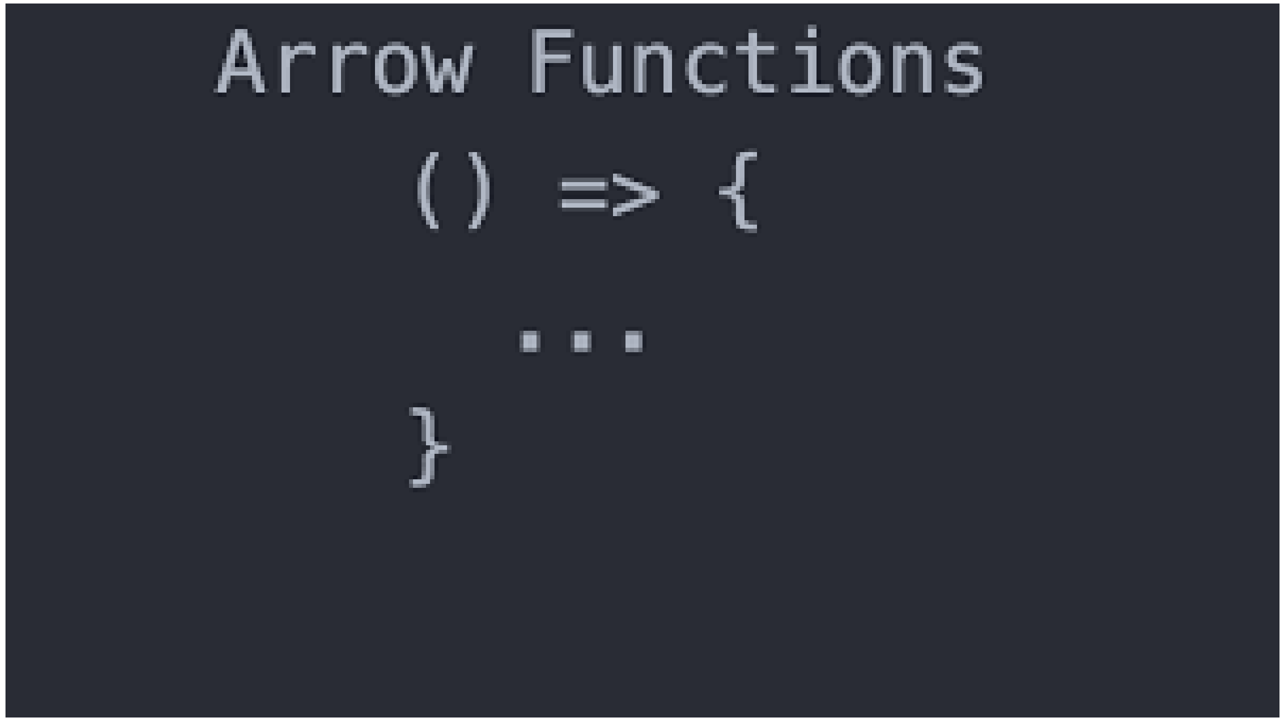
- In JavaScript, a **function is a reusable chunk of code created to carry out a certain operation**. After processing some optional input, it produces an optional output. Large programs can be divided into smaller, more manageable components with the use of functions.
- A function is defined using the **function** keyword, followed by **the function name, parentheses for parameters, and curly braces**, which contain the code to be executed.



```
function sendMessage(msg) {
    console.log(msg);
  }
```

- In JavaScript, both regular functions and arrow functions are used to define reusable blocks of code, but arrow functions offer a concise syntax and differ in this binding and how they handle arguments.

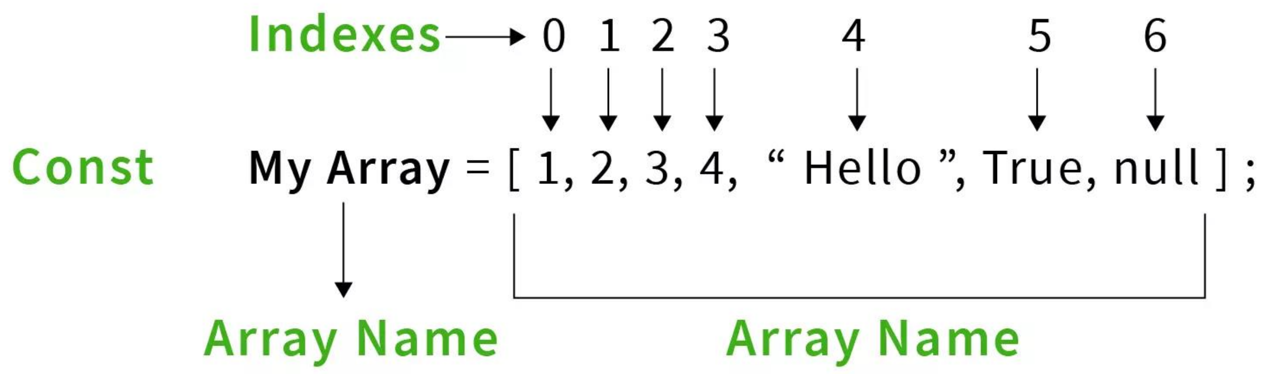
```
// an arrow function to add two numbers
const addNumbers = (a, b) => a + b;

// call the function with two numbers
const result = addNumbers(5, 3);
console.log(result);

// Output: 8
```



```
Arrow Functions
    () => {
        ...
    }
```

Arrays in JavaScript

- An array in JavaScript is a type of global object that is used to store data. Arrays consist of an ordered collection or list containing zero or more data types, and use numbered indices starting from 0 to access specific items.
- Arrays are very useful as they store multiple values in a single variable, which can condense and organize our code, making it more readable and maintainable. Arrays can contain any data type, including numbers, strings, and objects.



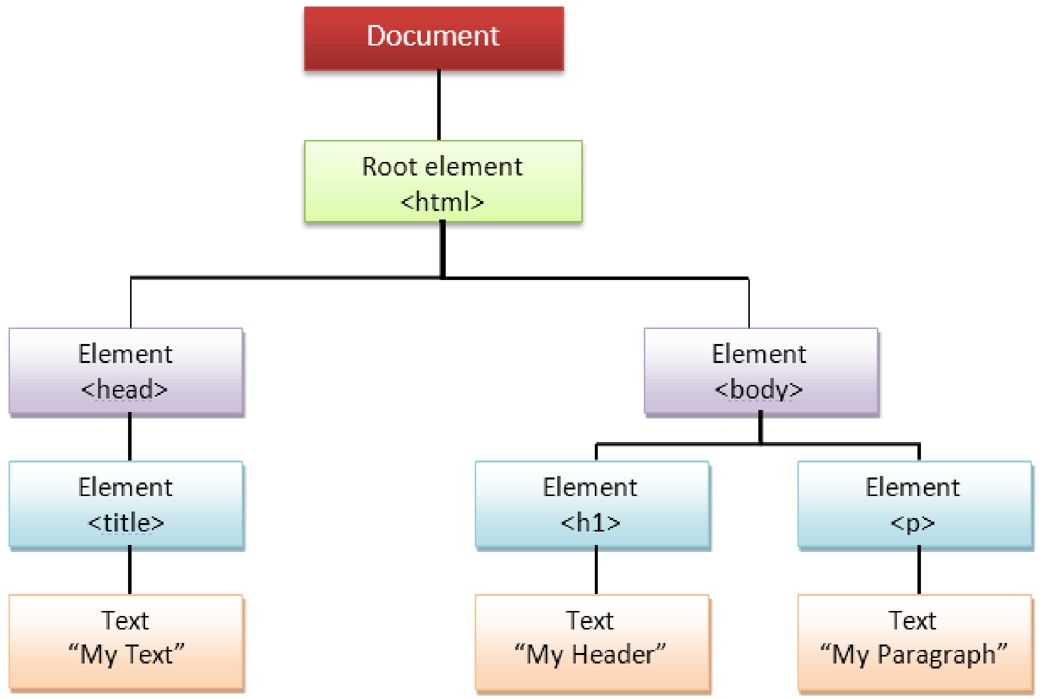
.map() & .filter()

- The map() method is used for creating a new array from an existing one, applying a function to each one of the elements of the first array.
- const output=arr.map((x)=>{ return x*2; });
- The filter() method takes each element in an array and it applies a conditional statement against it. If this conditional returns true, the element gets pushed to the output array.
- Example:

```
const arr=[1,2,3,4,5]
function isOdd(x)
{
   return x%2;
}
const output=arr.filter(isOdd);
```

DOM & Event Listener

- DOM is known as Document Object model.
- The Document Object Model (DOM) connects web pages to scripts or programming languages by representing the structure of a document—such as the HTML representing a web page—in memory. Usually it refers to JavaScript, even though modeling HTML, SVG, or XML documents as objects are not part of the core JavaScript language.



- The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javaScript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.
- For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Destructuring

- To illustrate destructuring, we'll make a sandwich. Do you take everything out of the refrigerator to make your sandwich? No, you only take out the items you would like to use on your sandwich.
- Destructuring is exactly the same. We may have an array or object that we are working with, but we only need some of the items contained in these.
- Example:

```
let a, b, rest;
[a, b] = [10, 20];
console.log(a);     // Expected output: 10
console.log(b);    // Expected output: 20
[a, b, ...rest] = [10, 20, 30, 40, 50]
;console.log(rest);   // Expected output: Array [30, 40, 50]
```