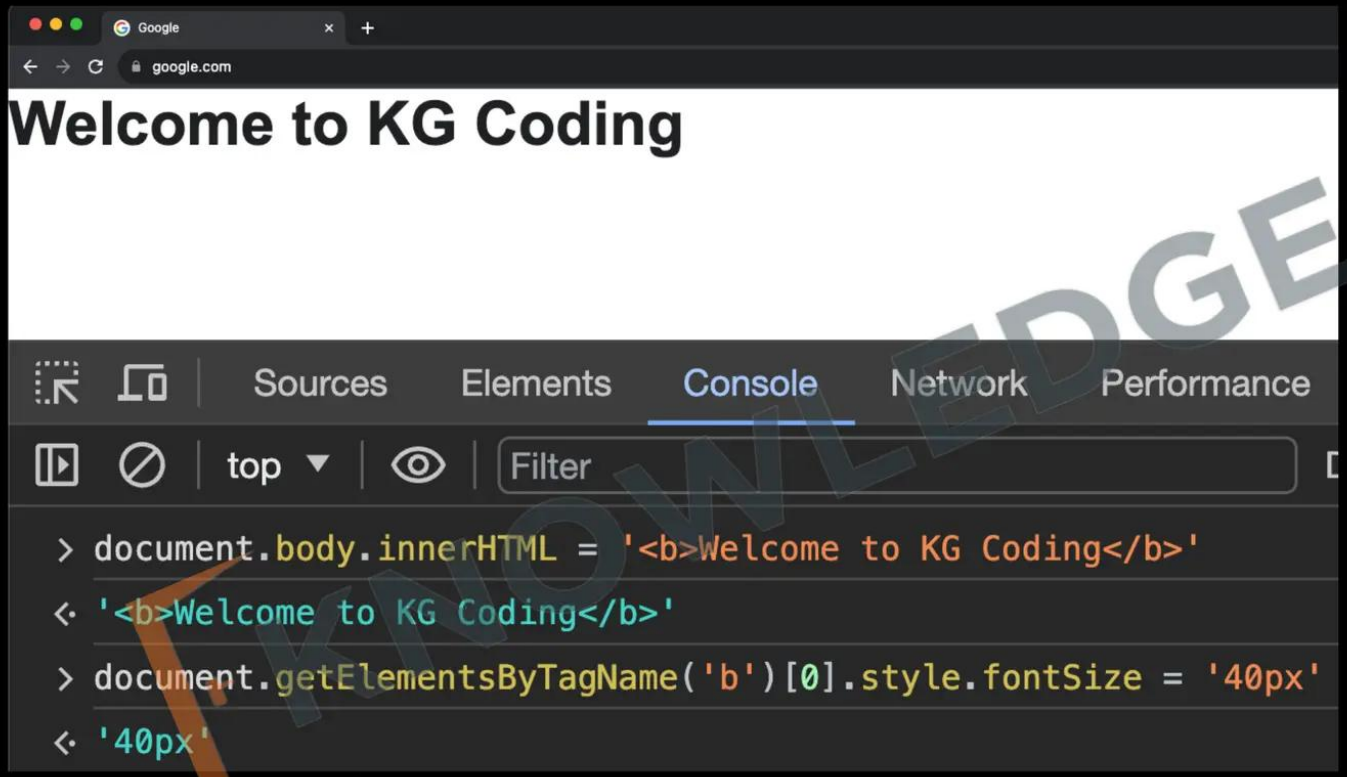
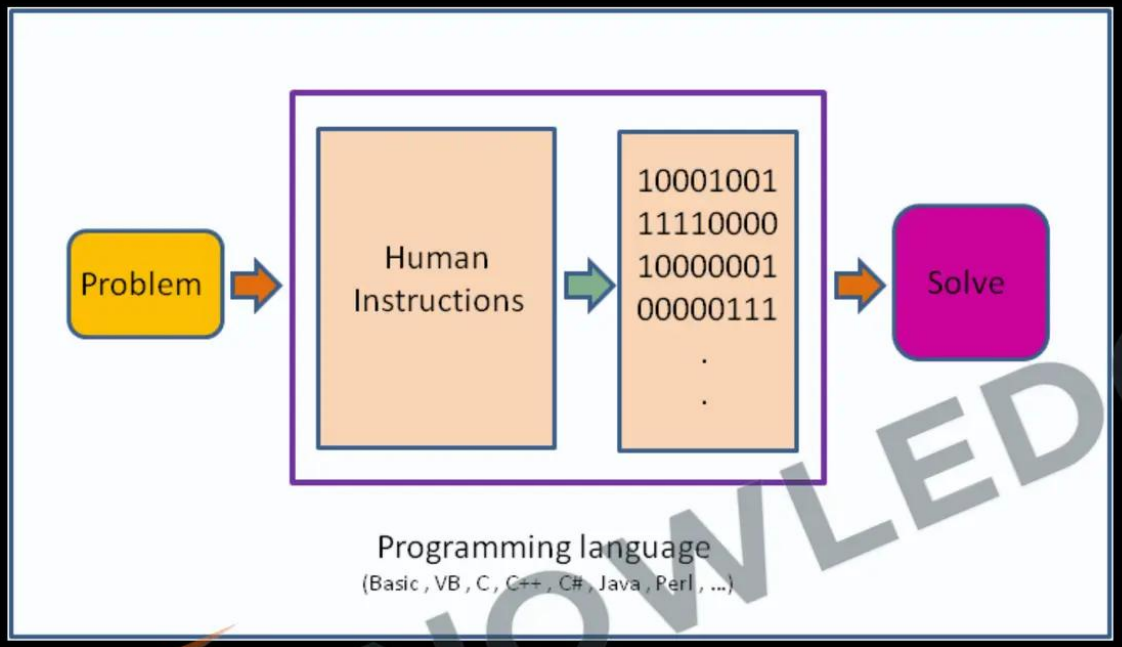
2. DOM Manipulation



1. Change HTML
2. Change CSS
3. Perform Actions

4. What is a Programming Language?



- Giving instructions to a computer
- **Instructions**: Tells computer what to do.
- These instructions are called code.

5. What is a Syntax?

- Structure of words in a sentence.
- Rules of the language.
- For programming exact syntax must be followed.

```
> alert 'hello world'
Uncaught SyntaxError: Unexpected string
>
```

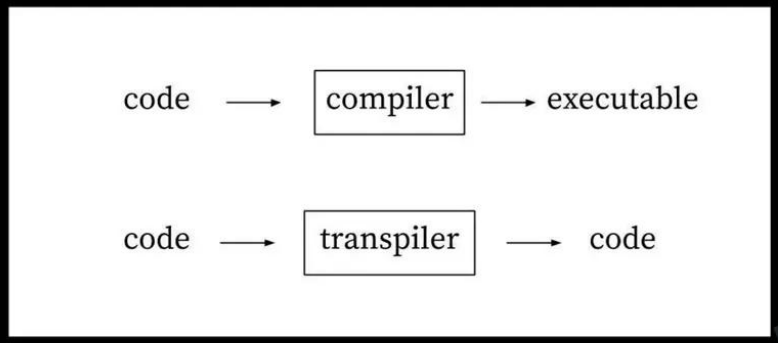
6. JS

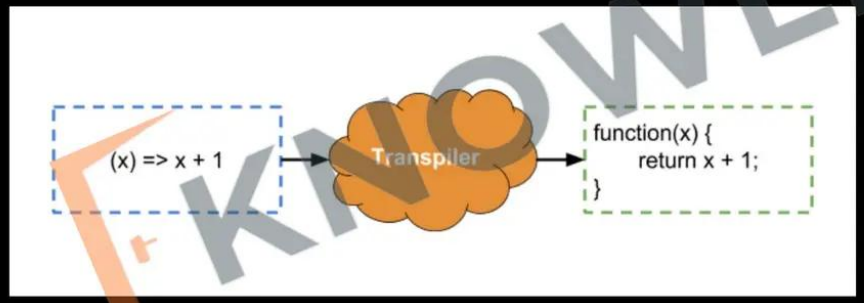
(Java Script)

1. JavaScript has nothing to do with Java
2. Actions: Enables interactivity.
3. Updates: Alters page without reloading.
4. Events: Responds to user actions.
5. Data: Fetches and sends info to server.

6. JS

(Java Script)





1. JavaScript runs at the client side in the browser.
2. Coffee Script / TypeScript are transpiled to JavaScript.

Numbers & Strings

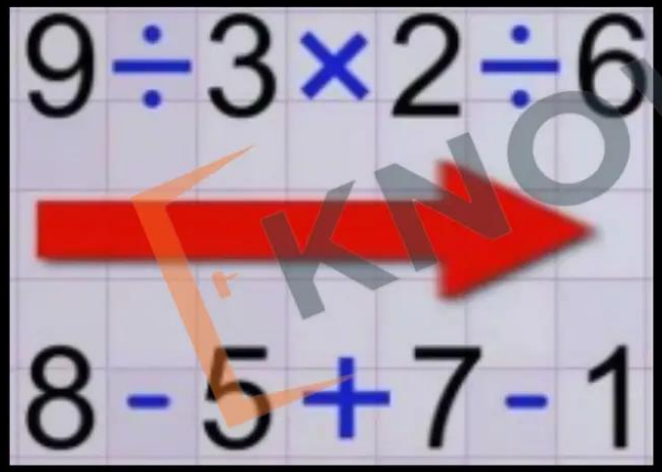
7. Arithmetic Operators
8. Order of Operations
9. Types of Numbers (Integers & Floats)
10. Don't learn syntax.
11. Strings
12. TypeOf Operator

7. Arithmetic Operators

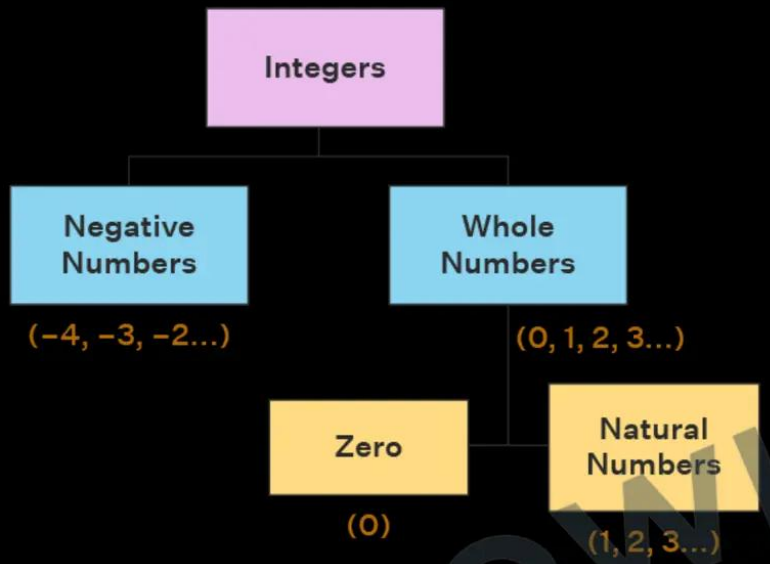
Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1

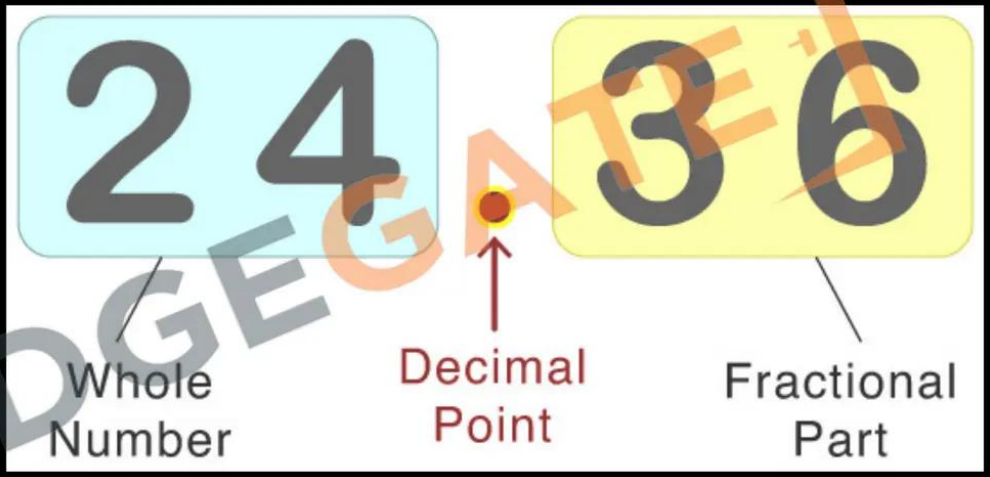
8. Order of Operations

B	O	D	M	A	S
Bracket	Order	Divide	Multiply	Add	Subtract
( )	$\sqrt{x}$ x^2	÷ or x		+ or -	
Parentheses	Exponents	Multiply	Divide	Add	Subtract
P	E	M	D	A	S

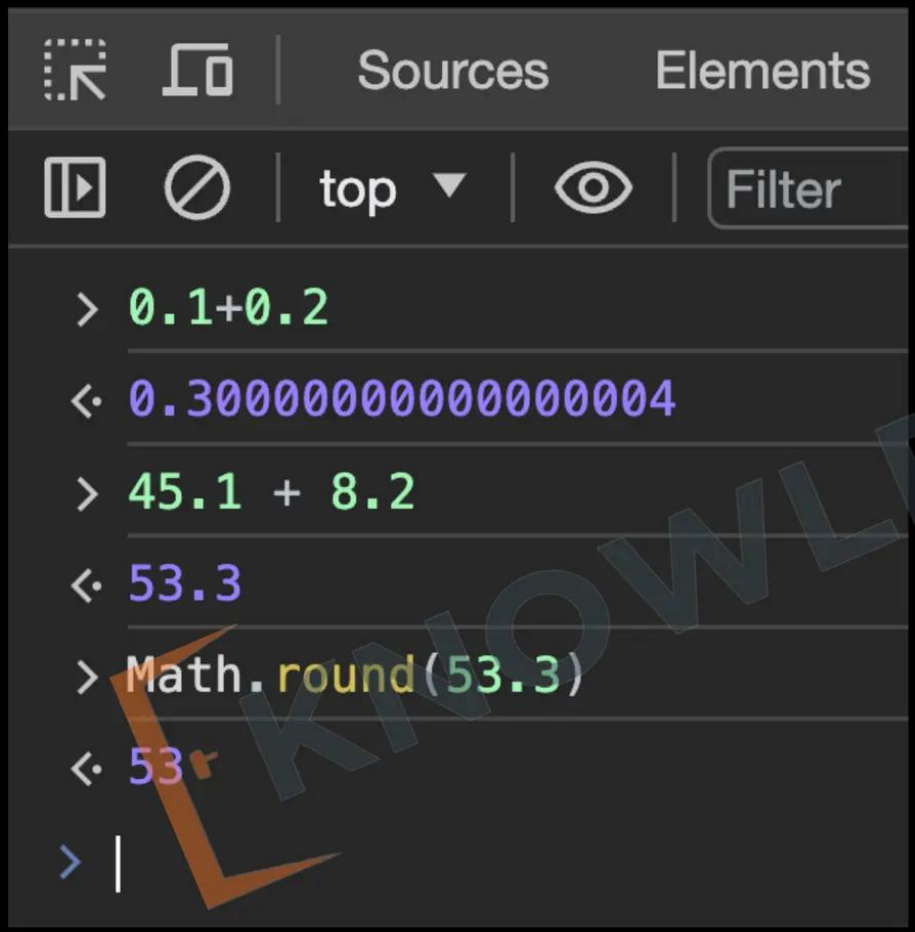


9. Types of Numbers (Integers & Floats)



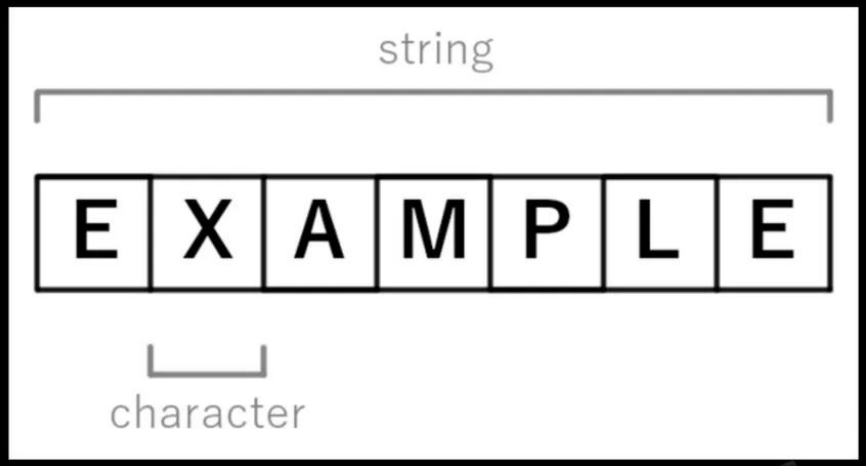


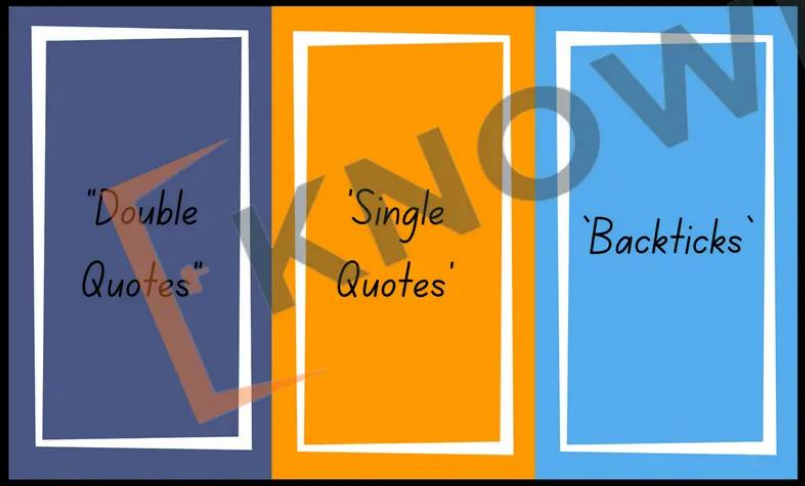
9. Types of Numbers (Float Problems)



1. JavaScript has many problems with float operations.
2. We can use Math.round() to convert floats to integers.
3. Always do money calculation in Paisa instead of Rupees

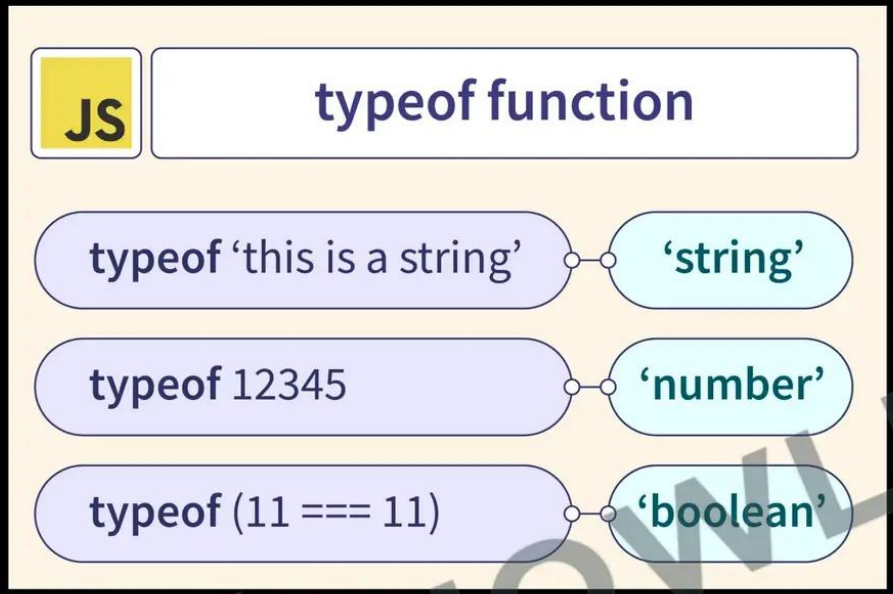
11. Strings





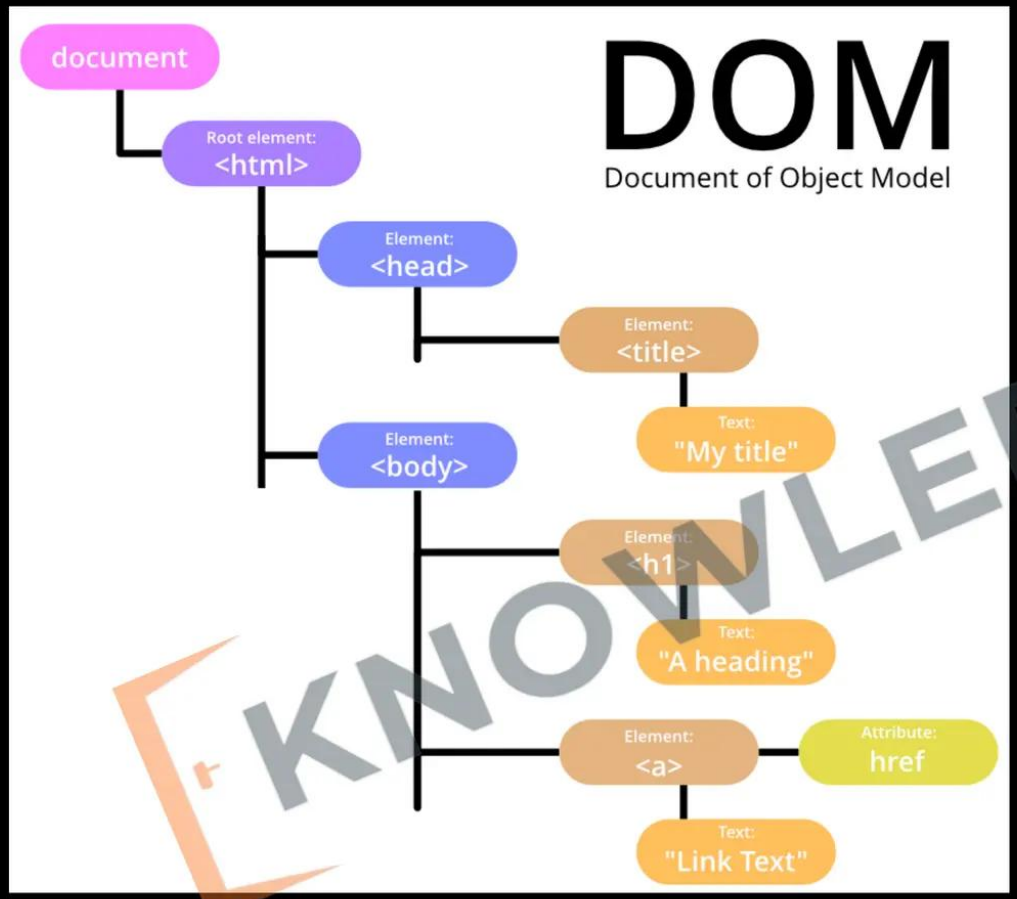
1. Strings hold textual data, anything from a single character to paragraph.
2. Strings can be defined using single quotes '', double quotes " ", or backticks ` `. Backticks allow for template literals, which can include variables.
3. You can combine (concatenate) strings using the + operator. For example, "Hello" + " World" will produce "Hello World".

12. Type Of Operator



1. Check Type: Tells you the data type of a variable.
2. Syntax: Use it like typeof variable.
3. Common Types: Returns "number," "string," "boolean," etc.

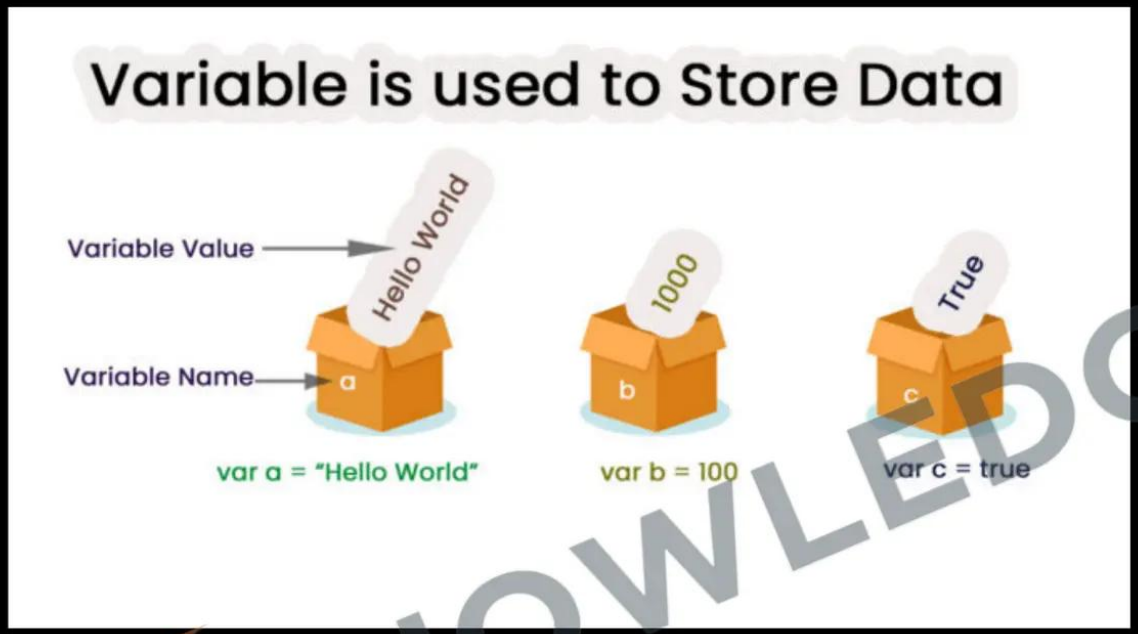
14 HTML DOM



1. Structure Understanding: Helps in understanding the **hierarchical structure** of a webpage, crucial for applying targeted CSS styles.

2. Dynamic Styling: Enables learning about dynamic styling, allowing for **real-time changes** and interactivity through CSS.

19. What are Variables?



Variables are like containers used for storing data values.

20. Syntax Rules

```
1    // Defining a number variable
2    let noOfStudents = 5;
3    // Defining a String variable
4    let welcomeMessage = "Hello Beta"
```

1. Can't use keywords or reserved words
2. Can't start with a number
3. No special characters other than $ and _
4. = is for assignment
5. ; means end of instruction

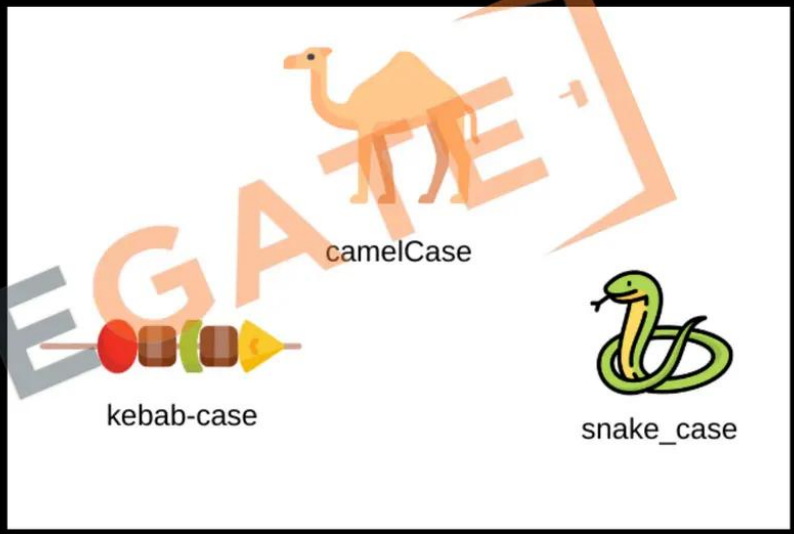
21. Updating Values

```
let noOfStudents = 5;
noOfStudents = noOfStudents + 1;

let money = 1;
money += 5; // money = 6
money -= 2; // money = 4
money *= 3; // money = 12
money /= 4; // money = 3
money++;    // money = 4
```

1. Do not need to use let again.
2. Syntax: variable = variable + 1
3. Assignment Operator is used =
4. Short Hand Assignment Operators:
   +=, -=, *=, /=, ++

23. Naming Conventions



camelCase

- Start with a lowercase letter. Capitalize the first letter of each subsequent word.
- Example: myVariableName

snake_case

- Start with an lowercase letter. Separate words with underscore
- Example: my_variable_name

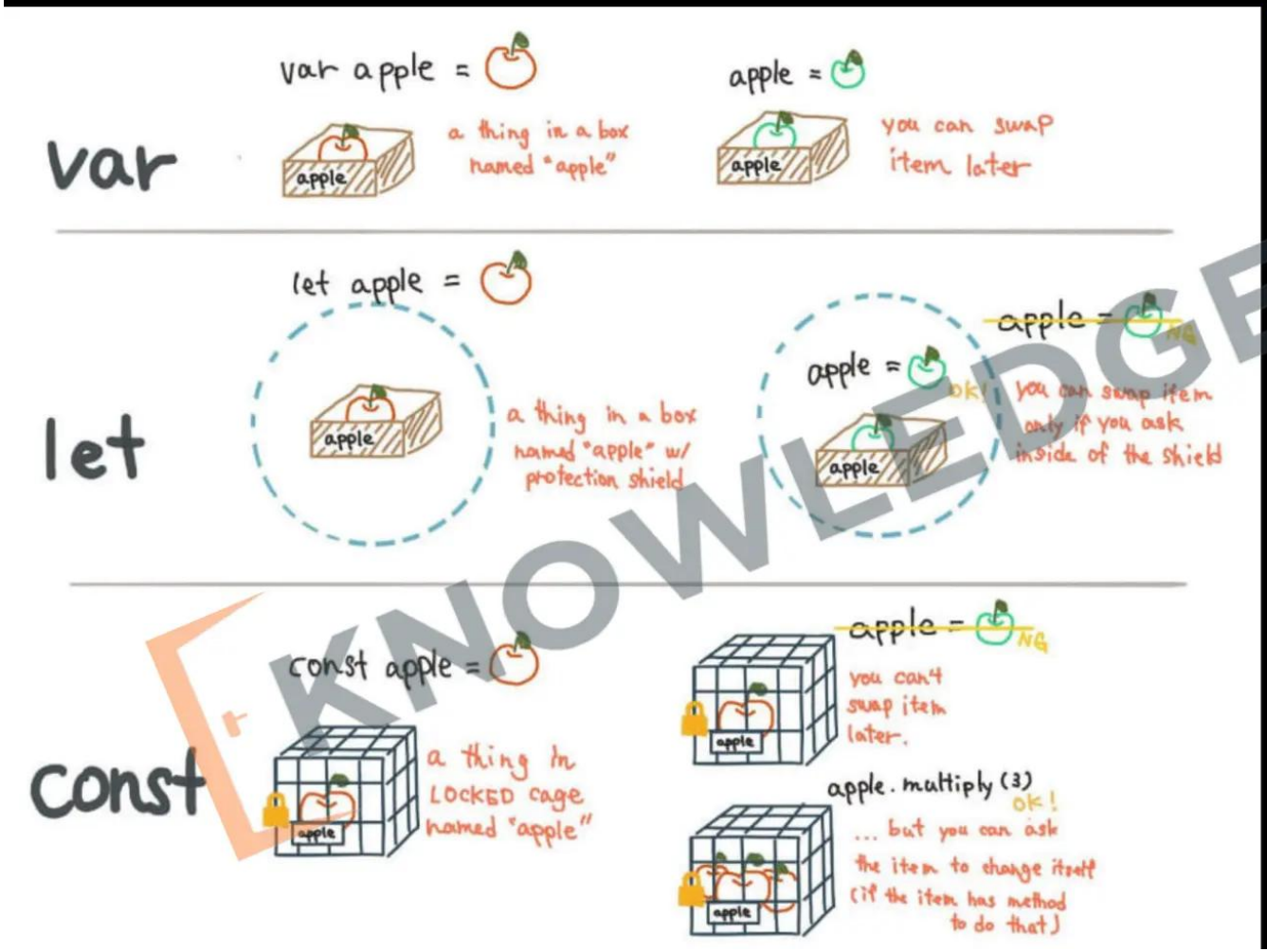
Kebab-case

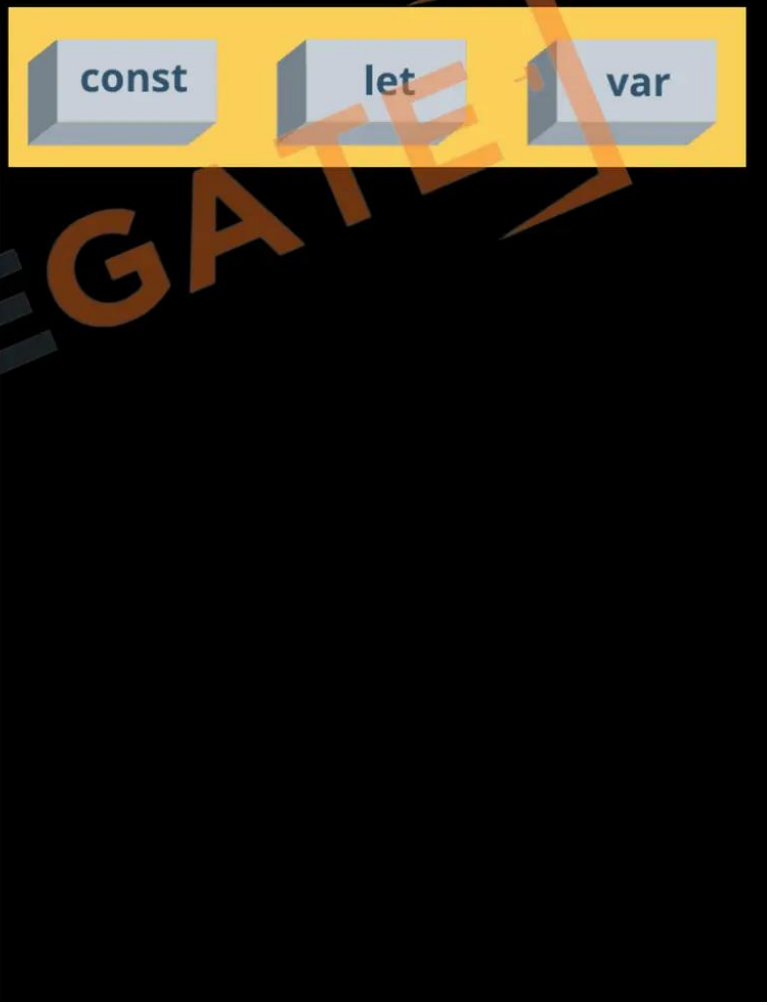
- All lowercase letters. Separate words with hyphens. Used for HTML and CSS.
- Example: my-variable-name

Keep a Good and Short Name

- Choose names that are descriptive but not too long. It should make it easy to understand the variable's purpose.
- Example: age, firstName, isMarried

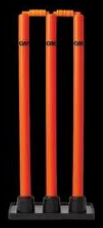
24. Ways to Create Variables



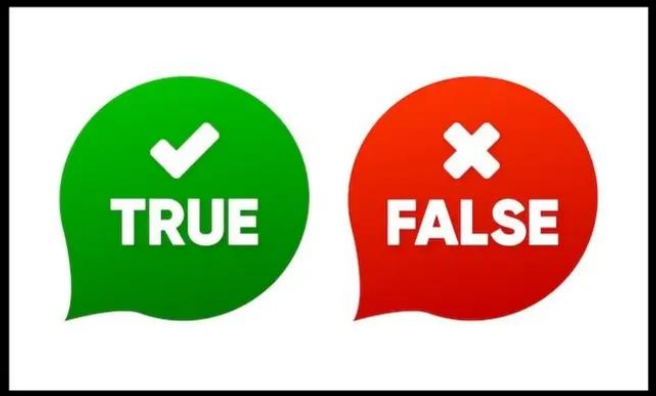


if-else & Boolean

25. What are Booleans
26. Comparison Operators
27. if-else
28. Logical Operator
29. Scope
30. Truthy and Falsy Values
31. If alternates

Create    Project

25. What are Booleans?



1. Data Type: Booleans are a basic data type in JavaScript.
2. Two Values: Can only be true or false.
3. 'true' is a String not a Boolean

26. Comparison Operators

<, >, <=, >=, ==, !=

Equality

- == Checks value equality.
- === Checks value and type equality.

Inequality

- != Checks value inequality.
- !== Checks value and type inequality.

Relational

- > Greater than.
- < Less than.
- >= Greater than or equal to.
- <= Less than or equal to.

Order of comparison operators is less than arithmetic operators

27. if-else

1. Syntax: Uses if () {} to check a condition.
2. What is if: Executes block if condition is true, skips if false.
3. What is else: Executes a block when the if condition is false.
4. Curly Braces can be omitted for single statements, but not recommended.
5. If-else Ladder: Multiple if and else if blocks; only one executes.
6. Use Variables: Can store conditions in variables for use in if statements.

28. Logical Operators

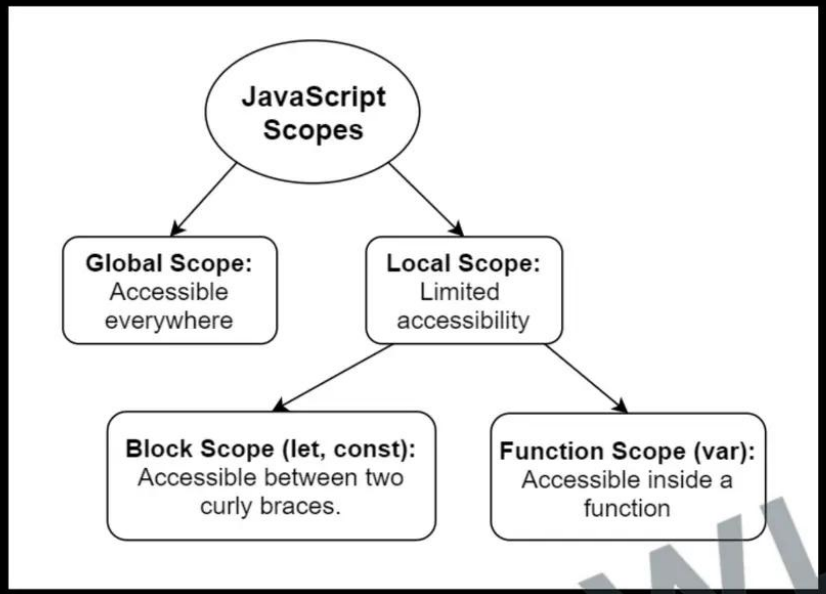

AND


Or

not

1. Types: && (AND), || (OR), ! (NOT)
2. AND (&&): All conditions must be true for the result to be true.
3. OR (||): Only one condition must be true for the result to be true.
4. NOT (!): Inverts the Boolean value of a condition.
5. Lower Priority than Math and Comparison operators

29. Scope



1. Any variable created inside {} will remain inside {}
2. Variable can be redefined inside {}
3. Var does not follow scope
4. Global Scope: Accessible everywhere in the code.
5. Block Scope: Limited to a block, mainly with let and const.
6. Declare variables in the narrowest scope possible.

30. Truthy and Falsy Values

1. Falsy Values: 0, null, undefined, false, NaN, "" (empty string)
2. Truthy Values: All values not listed as falsy.
3. Used in conditional statements like if.
4. Non-boolean values are auto-converted in logical operations.
5. Be explicit in comparisons to avoid unexpected behaviour.

31. If alternates

1. Ternary Operator: condition ? trueValue : falseValue

   Quick one-line if-else.

2. Guard Operator: value || defaultValue

   Use when a fallback value is needed.

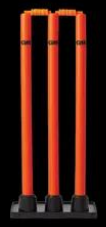
3. Default Operator: value ?? fallbackValue

   Use when you want to consider only null and undefined as falsy.

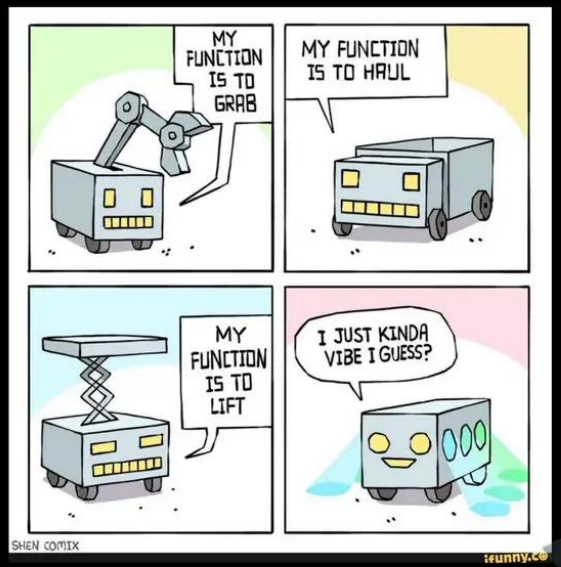
4. Simplifies conditional logic.
5. Use wisely to maintain readability.

Functions

32. What are Functions
33. Function Syntax
34. Return statement
35. Function Parameters

Improve    Project

32. What are Functions?



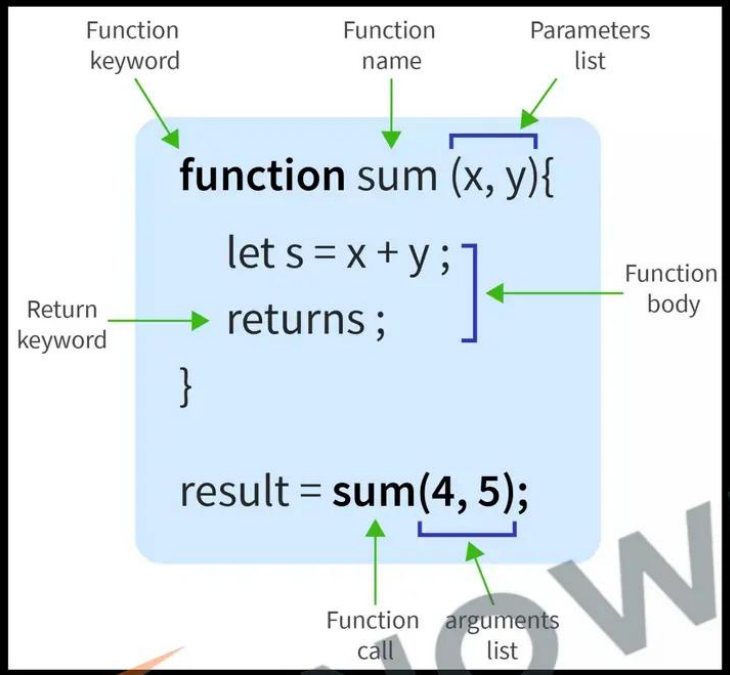

```
function greet(name) {
    // code
}

greet(name);
// code
```

function call

1. Definition: Blocks of reusable code.
2. DRY Principle: "Don't Repeat Yourself" it Encourages code reusability.
3. Usage: Organizes code and performs specific tasks.
4. Naming Rules: Same as variable names: camelCase
5. Example: ”Beta Gas band kar de"

33. Functions Syntax



1. Use function keyword to declare.
2. Follows same rules as variable names.
3. Use () to contain parameters.
4. Invoke by using the function name followed by ().
5. Fundamental for code organization and reusability.

34. Return statement

1. Sends a value back from a function.
2. Example: "Ek glass paani laao"
3. What Can Be Returned: Value, variable, calculation, etc.
4. Return ends the function immediately.
5. Function calls make code jump around.
6. Prefer returning values over using global variables.

35. Parameters

Parameter

Function

Return

1. Input values that a function takes.
2. Parameters put value into function, while return gets value out.
3. Example: "Ek packet dahi laao"
4. Naming Convention: Same as variable names.
5. Parameter vs Argument
6. Examples: alert, Math.round, console.log are functions we have already used
7. Multiple Parameters: Functions can take more than one.
8. Default Value: Can set a default value for a parameter.

Objects

36. What is an Object
37. Objects Syntax
38. Accessing Objects
39. Inside Objects
40. Autoboxing
41. Object References
42. Object Shortcuts

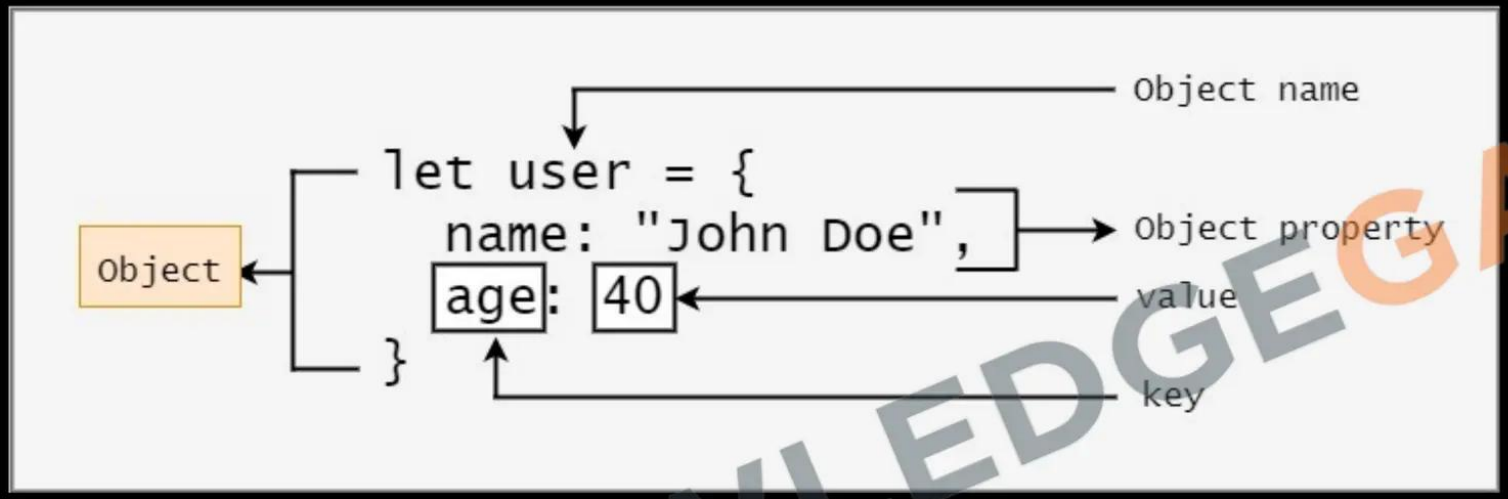
Maintain score of    Project

36. What is an Object?


```
let product = {
  company: 'Mango',
  item_name: 'Cotton striped t-shirt',
  price: 861
};
```


1. Groups multiple values together in key-value pairs.
2. How to Define: Use {} to enclose properties.
3. Example: product {name, price}
4. Dot Notation: Use . operator to access values.
5. Key Benefit: Organizes related data under a single name.

37. Object Syntax?



1. Basic Structure: Uses {} to enclose data.
2. Rules: Property and value separated by a colon(:)
3. Comma: Separates different property-value pairs.
4. Example: { name: "Laptop", price: 1000 }

38. Accessing Objects

1. Dot Notation: Access properties using . Operator like product.price
2. Bracket Notation: Useful for properties with special characters product[”nick-name"]. Variables can be used to access properties
3. typeof returns object.
4. Values can be added or removed to an object
5. Delete Values using delete

39. Inside Object

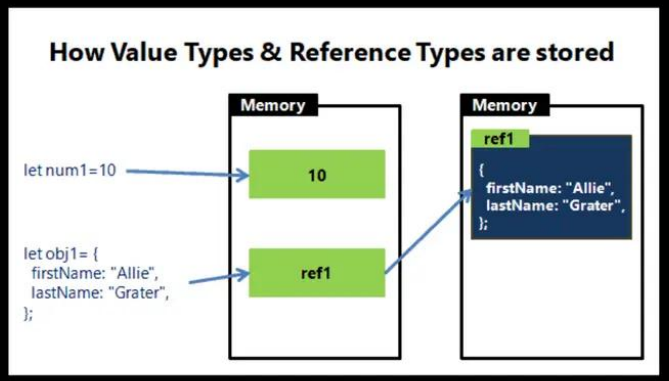
```
let product = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  price: 861,
  rating: {
    stars: 4.5,
    noOfReviews: 87
  },
  displayPrice: function() {
    return `$${this.price.toFixed(2)}`;
  }
};
```

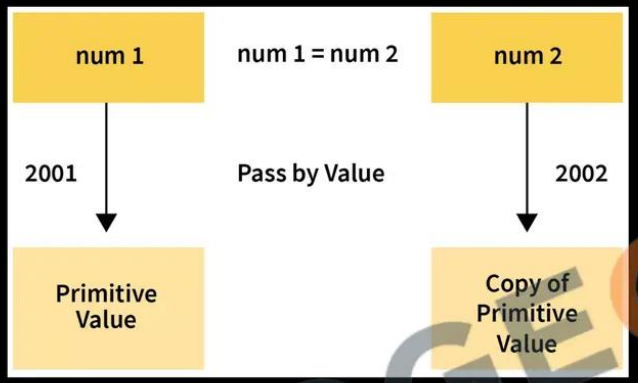
1. Objects can contain Primitives like numbers and strings.
2. Objects can contain other objects and are called Nested Objects.
3. Functions can be object properties.
4. Functions inside an object are called methods.
5. Null Value: Intentionally leaving a property empty.

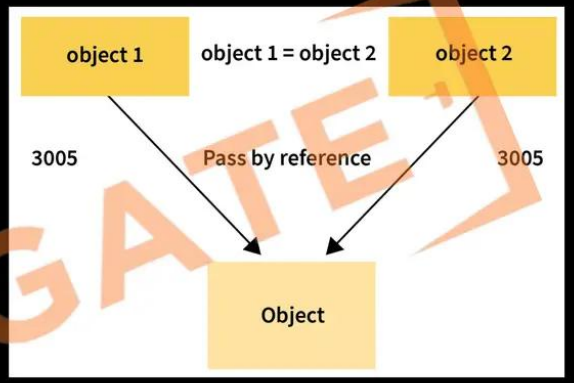
40. Autoboxing

1. Automatic conversion of primitives to objects.
2. Allows properties and methods to be used on primitives.
3. Example: Strings have properties and methods like length, toUpperCase, etc.

41. Object References







1. Objects work based on references, not actual data.
2. Copying an object copies the reference, not the actual object.
3. When comparing with ==, you're comparing references, not content.
4. Changes to one reference affects all copies.

42. Object Shortcuts

```
let product = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  price: 861
};
// Destructuring
let company = product.company
// is same as
let { company } = product;
```

```
// Property shorthand
let price = 861;
let product = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  price: price
};
// is same as
let product1 = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  price
};
```

```
// Method shorthand
let product = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  displayPrice: function() {
    return `$${this.price.toFixed(2)}`;
  }
};
// is same as
let product1 = {
  company: 'Mango',
  itemName: 'Cotton striped t-shirt',
  displayPrice() {
    return `$${this.price.toFixed(2)}`;
  }
};
```

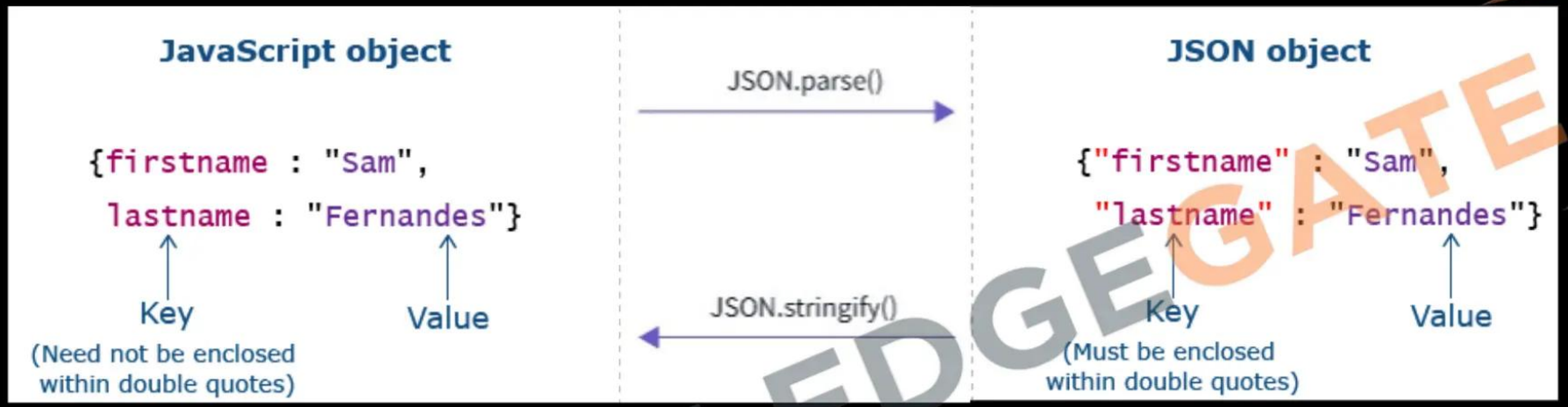
1. De-structuring: Extract properties from objects easily.
2. We can extract more than one property at once.
3. Shorthand Property: {message: message} simplifies to just message.
4. Shorthand Method: Define methods directly inside the object without the function keyword.

JSON, Local Storage, Date & DOM

43. What is JSON?
44. Local Storage
45. Date
46. DOM Properties & Methods

Finish  Project

43. What is JSON?

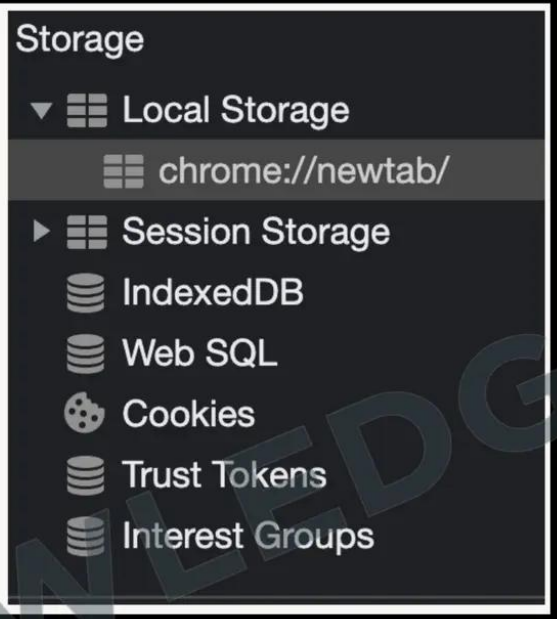


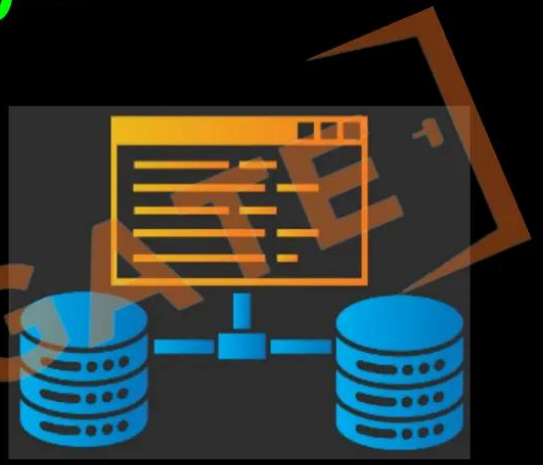
1. JavaScript Object Notation: Not the same as JS object, but similar.
2. Common in network calls and data storage.
3. JSON.stringify() and JSON.parse()
4. Strings are easy to transport over network.
5. JSON requires double quotes. Escaped as \".
6. JSON is data format, JS object is a data structure.

44. Local Storage

```
// store an object in Local Storage
localStorage.setItem(
      "user",
      JSON.stringify({
            name: "Gopi Gorantala"
            age: 32
       });
);

// retrieve an object in Local Storage
const user = JSON.parse(
     localStorage.getItem("user")
);
```





1. Persistent data storage in the browser.
2. setItem: Stores data as key-value pairs.
3. Only strings can be stored.
4. getItem: Retrieves data based on key.
5. Other Methods: localStorage.clear(), removeItem().
6. Do not store sensitive information. Viewable in storage console.

45. Date

1. new Date() Creates a new Date object with the current date and time.
2. Key Methods:
   - getTime(): Milliseconds since Epoch.
   - getFullYear(): 4-digit year
   - getDay(): Day of the week
   - getMinutes(): Current minute
   - getHours(): Current hour.
3. Crucial for timestamps, scheduling, etc.

46. DOM Properties & Methods

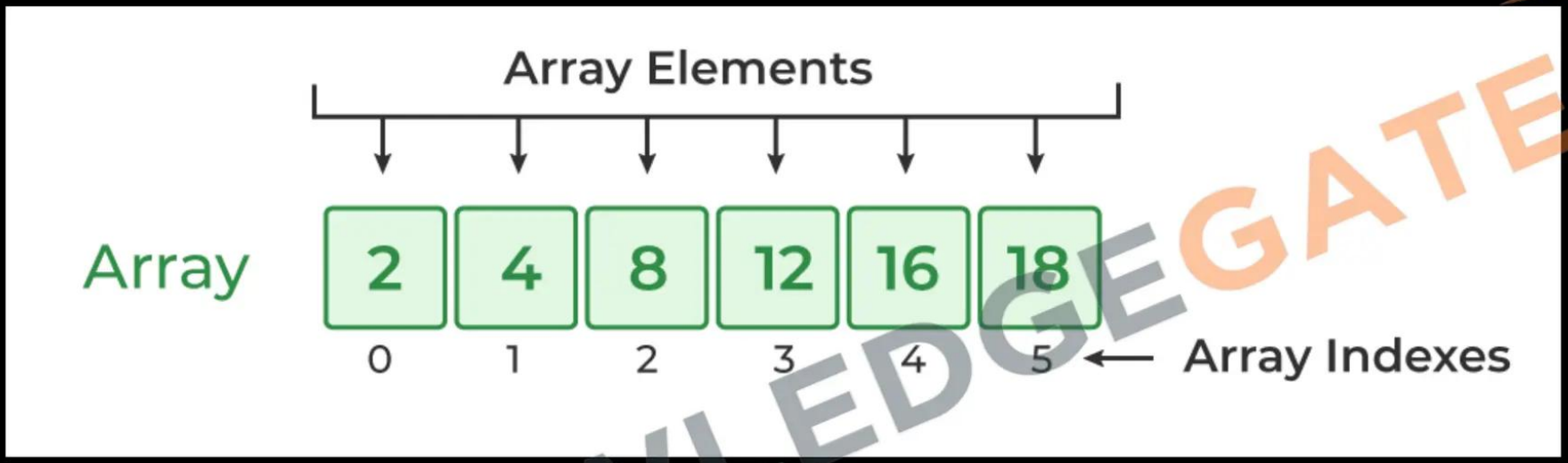
DOM and Element Properties

1. location
2. title
3. href
4. domain
5. innerHTML
6. innerText
7. classList

DOM and Element Methods

1. getElementById()
2. querySelector()
3. classList: add(), remove()
4. createElement()
5. appendChild()
6. removeChild()
7. replaceChild()

47. What is an Array?



1. An Array is just a list of values.
2. Index: Starts with 0.
3. Arrays are used for storing multiple values in a single variable.

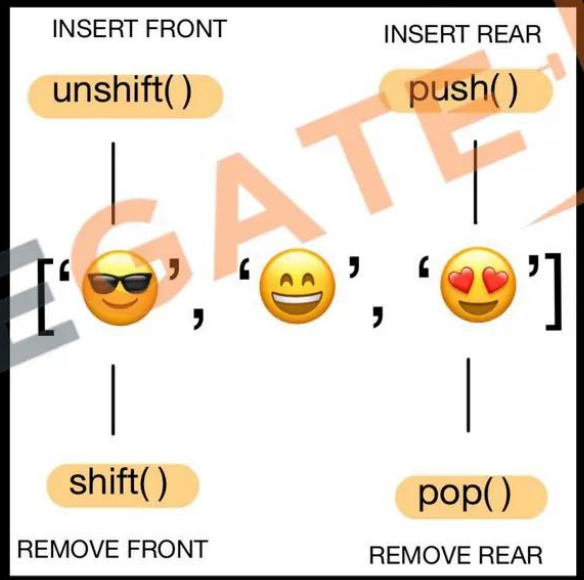
48. Array Syntax & Values

```
let myArray = [1, 'KG Coding', null, true,
  {likes: '1 Million'}];
```

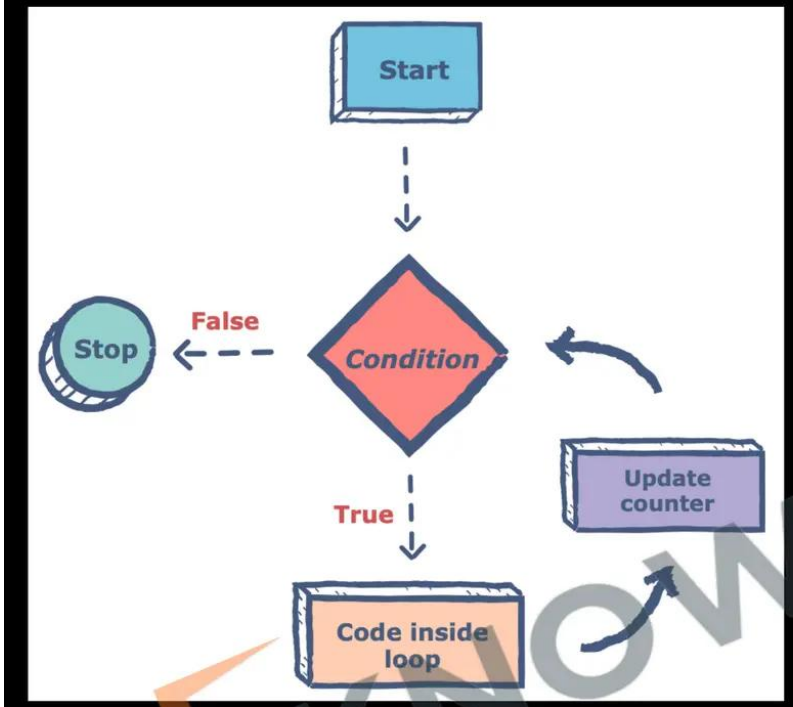
1. Use [] to create a new array, [] brackets enclose list of values
2. Arrays can be saved to a variable.
3. Accessing Values: Use [] with index.
4. Syntax Rules:
   - Brackets start and end the array.
   - Values separated by commas.
   - Can span multiple lines.
5. Arrays can hold any value, including arrays.
6. typeof operator on Array Returns Object.

49. Array Properties & Methods

1. Array.isArray() checks if a variable is an array.
2. Length property holds the size of the array.
3. Common Methods:
   - push/pop: Add or remove to end.
   - shift/unshift: Add or remove from front.
   - splice: Add or remove elements.
   - toString: Convert to string.
   - sort: Sort elements.
   - valueOf: Get array itself.
4. Arrays also use reference like objects.
5. De-structuring also works for Arrays.

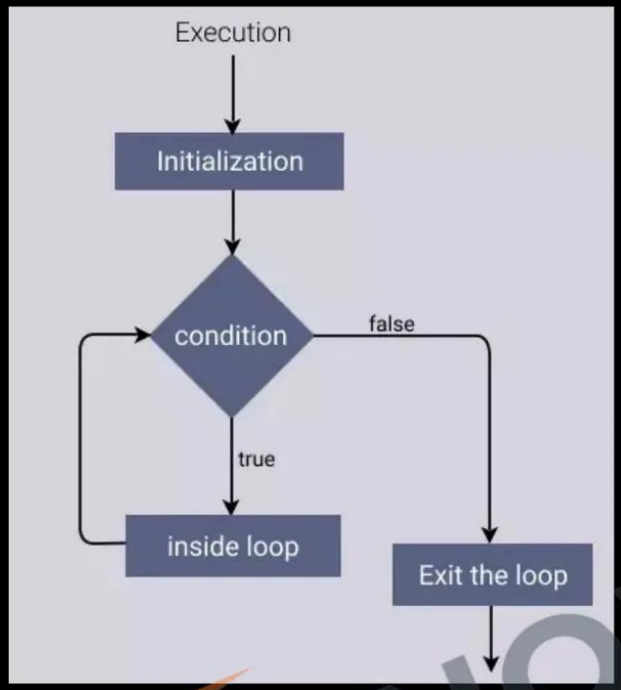


50. What is a Loop?



1. Code that runs multiple times based on a condition.
2. Loops also alter the flow of execution, similar to functions.
   - Functions: Reusable blocks of code.
   - Loops: Repeated execution of code.
3. Loops automate repetitive tasks.
4. Types of Loops: for, while, do-while.
5. Iterations: Number of times the loop runs.

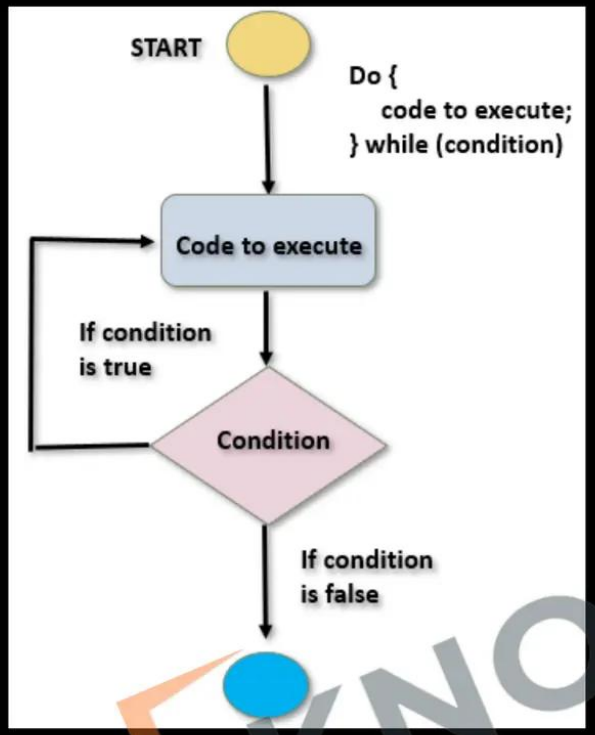
51. While Loop




```
while (condition) {
    // Body of the loop
}
```


1. Iterations: Number of times the loop runs.
2. Used for non-standard conditions.
3. Repeating a block of code while a condition is true.
4. Remember: Always include an update to avoid infinite loops.

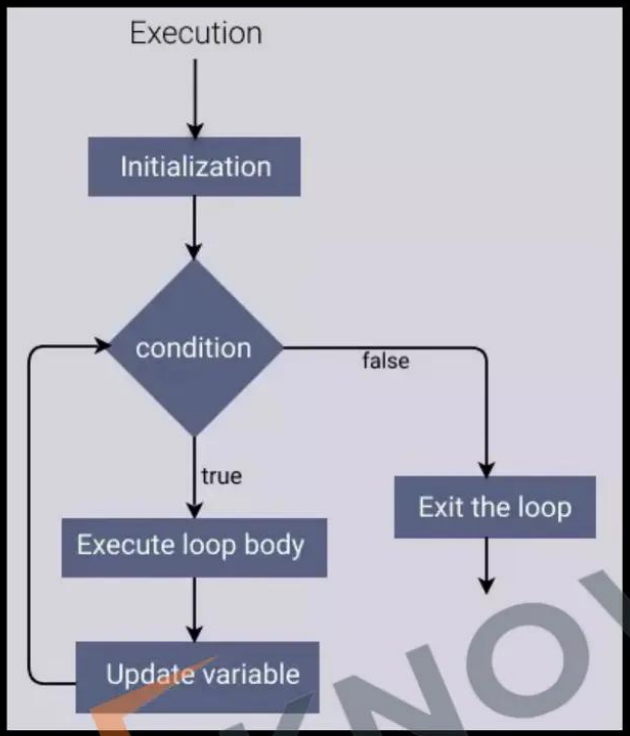
52.Do While Loop




```
do {
  // Body of the loop
}
while (condition);
```


1. Executes block first, then checks condition.
2. Guaranteed to run at least one iteration.
3. Unlike while, first iteration is unconditional.
4. Don't forget to update condition to avoid infinite loops.

53.For Loop



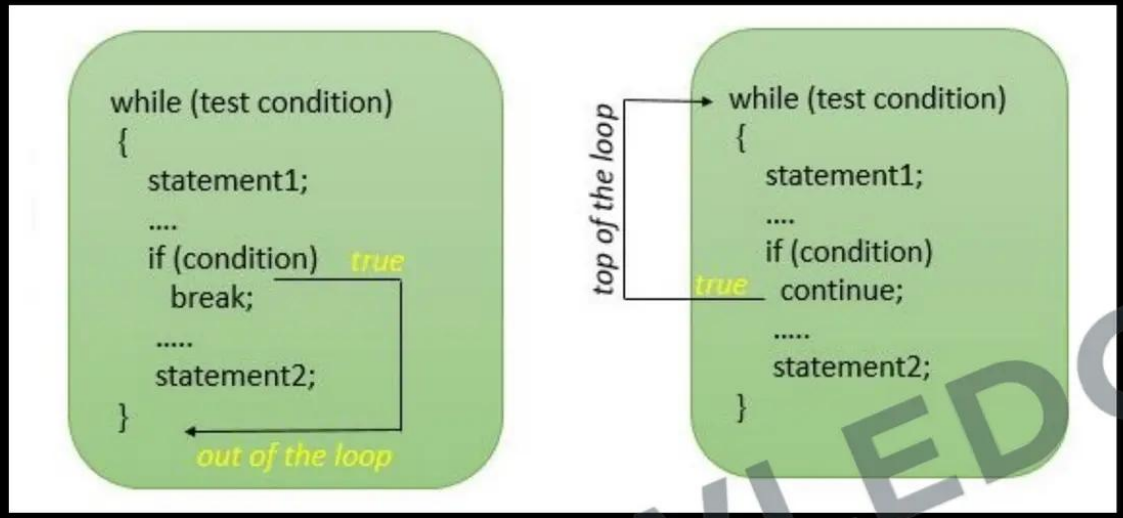

```
for (initialisation; condition; update) {
  // Body of the loop
}
```


1. Standard loop for running code multiple times.
2. Generally preferred for counting iterations.

54. Accumulator Pattern

1. A pattern to accumulate values through looping.
2. Common Scenarios:
   - Sum all the numbers in an array.
   - Create a modified copy of an array.

55.Break & Continue



1. Break lets you stop a loop early, or break out of a loop
2. Continue is used to skip one iteration or the current iteration
3. In while loop remember to do the increment manually before using continue

56. Anonymous Functions As Values

```
let sum = function(num1, num2) {
  return num1 + num2;
}
```

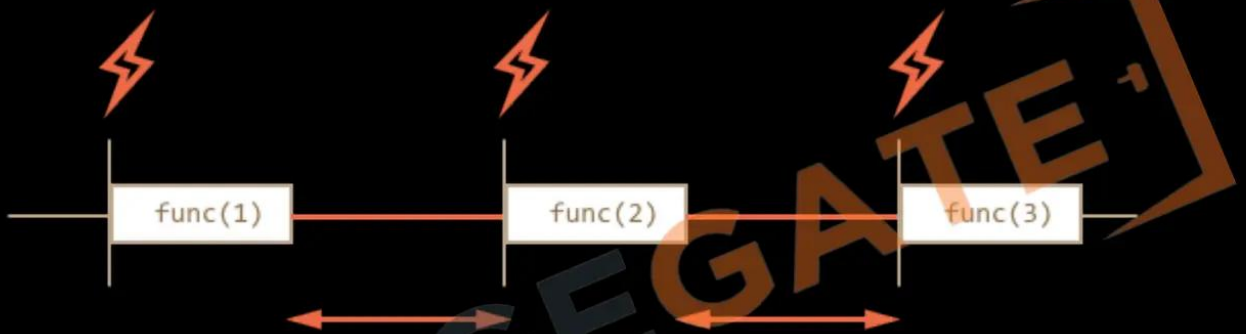

1. Functions in JavaScript are first-class citizens; they can be assigned to variables.
2. Functions defined without a name, often assigned to a variable.
3. Anonymous functions can be properties in objects
4. Can be passed as arguments to other functions.
5. Invoked using () after the function name or variable.
6. console.log(myFunction); and typeof myFunction will both indicate it's a function.

57. Arrow Functions

```
let sum = function(num1, num2) {
  return num1 + num2;
}
let Sum1 = (num1, num2) => {
  return num1 + num2;
}
let Sum2 = (num1, num2) => num1 + num2;
let square = num => num * num;
```

1. A concise way to write anonymous functions.
2. For Single Argument: Round brackets optional.
3. For Single Line: Curly brackets and return optional.
4. Often used when passing functions as arguments.

58. setTimeout & setInterval



1. Functions for executing code asynchronously after a delay.
2. setTimeout runs once; setInterval runs repeatedly
3. setTimeout:
   - Syntax: setTimeout(function, time)
   - Cancel: clearTimeout(timerID)
4. setInterval:
   - Syntax: setInterval(function, time)
   - Cancel: clearInterval(intervalID)

59. Event Listener

```
const button = document.querySelector(".btn")

button.addEventListener("click", event => {
  console.log("Hello!");
})
```

1. What Is an Event: Occurrences like clicks, mouse movement, keyboard input (e.g., birthday, functions).
2. Using querySelector to attach listeners.
3. Multiple Listeners: You can add more than one.
4. removeEventListener('event', functionVariable);

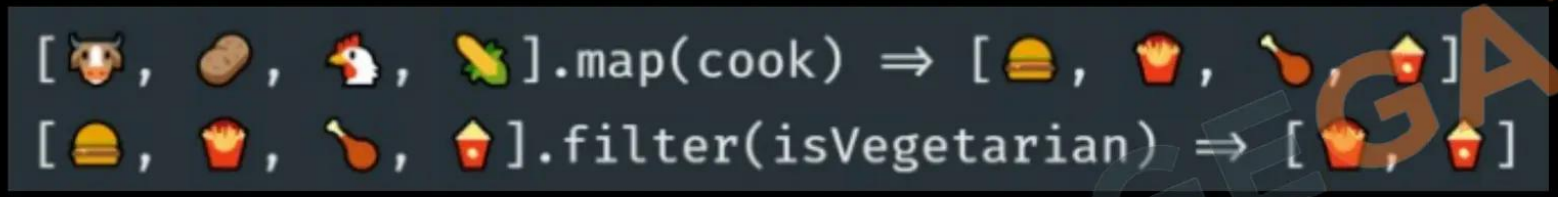
60. For Each Loop

```
let foods = ['bread', 'rice', 'meat', 'pizza'];

foods.forEach(function(food) {
    console.log(food);
})
```

1. A method for array iteration, often preferred for readability.
2. Parameters: One for item, optional second for index.
3. Using return is similar to continue in traditional loops.
4. Not straightforward to break out of a forEach loop.
5. When you need to perform an action on each array element and don't need to break early.

61. Array Methods



1. Filter Method:
   - Syntax: array.filter((value, index) => return true/false)
   - Use: Filters elements based on condition.
2. Map Method:
   - Syntax: array.map((value) => return newValue)
   - Use: Transforms each element.