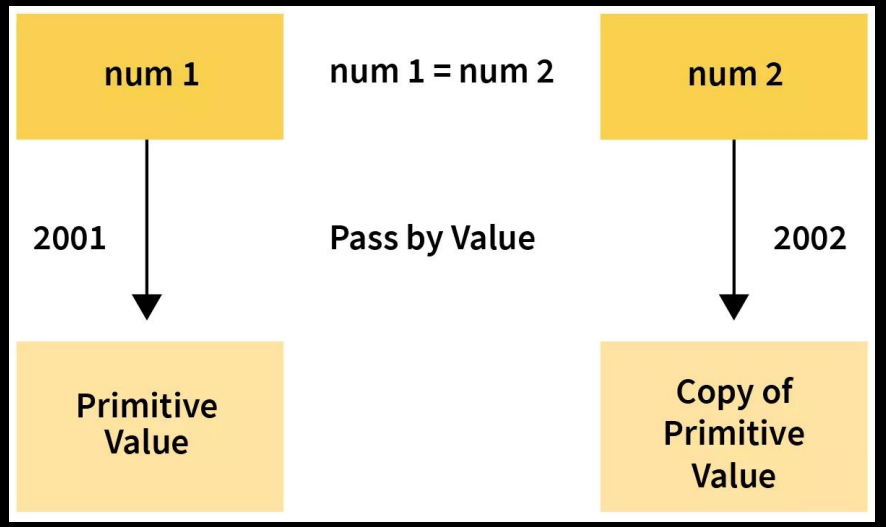
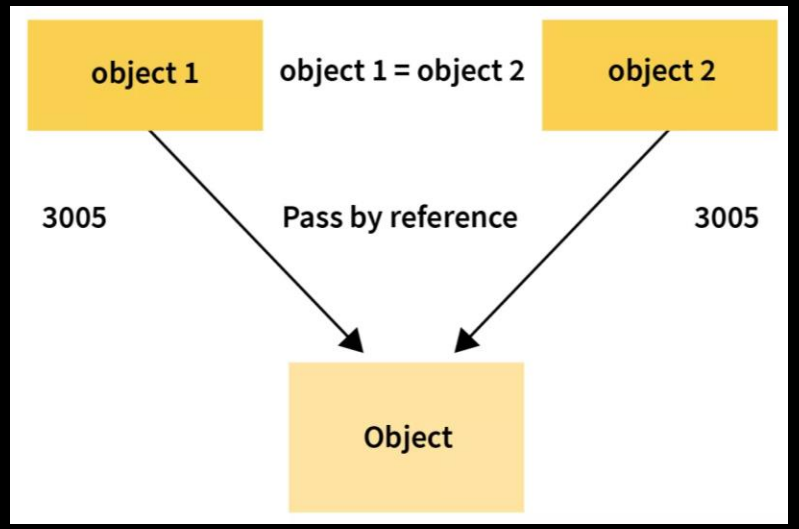
Object References





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.

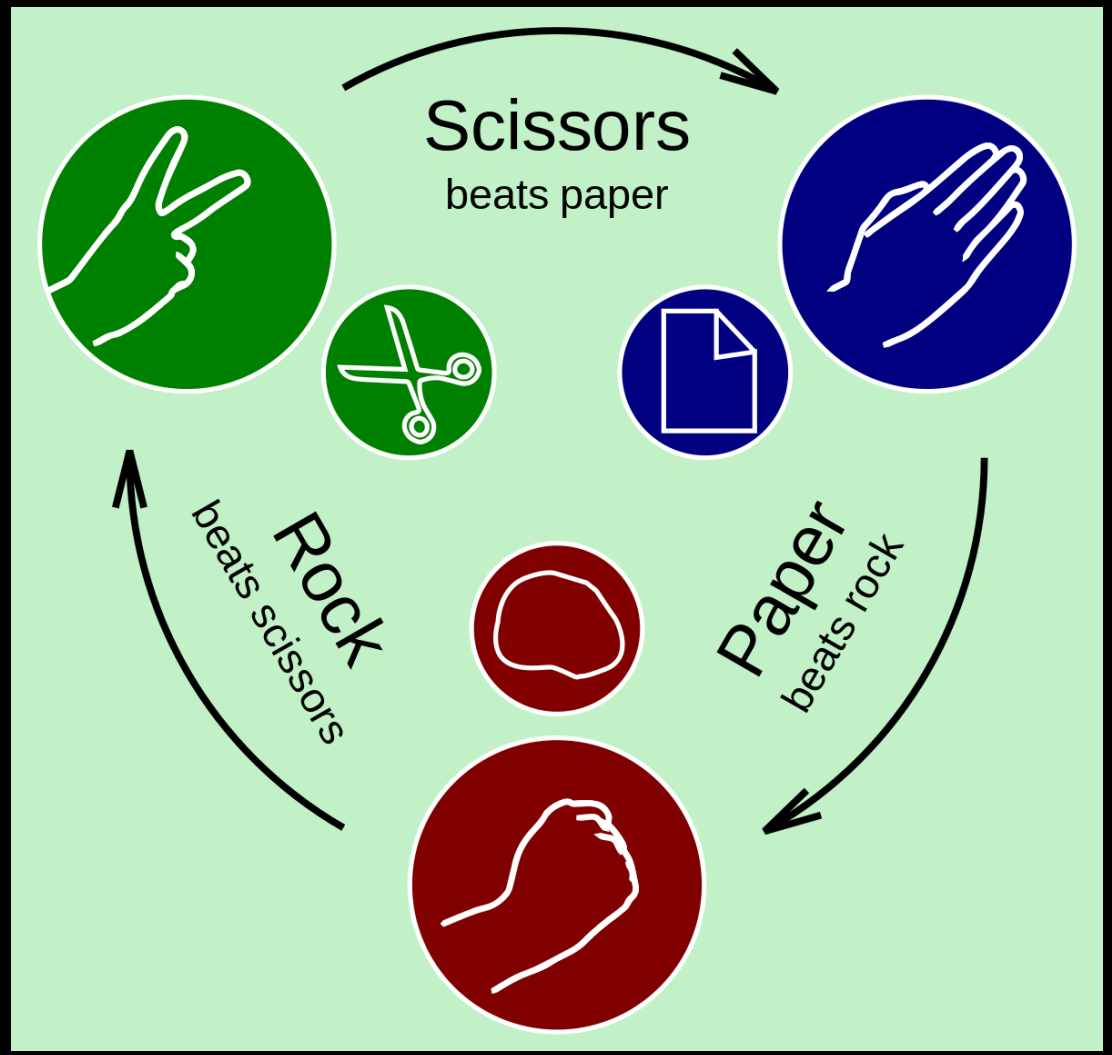
Call by Reference

```javascript
// Function to swap two numbers using call by reference
function trySwap(obj) {
  let temp = obj.a;
  obj.a = obj.b;
  obj.b = temp;
  console.log(`Inside trySwap - a: ${obj.a}, b: ${obj.b}`);
}

function main() {
  let obj = {a: 10, b: 20};
  console.log(`Before trySwap - a: ${obj.a}, b: ${obj.b}`);
  trySwap(obj); // Swap x and y using reference
  // The original values are changed
  console.log(`After trySwap - a: ${obj.a}, b: ${obj.b}`);
}

main();
```

```
Before trySwap - a: 10, b: 20
Inside trySwap - a: 20, b: 10
After trySwap - a: 20, b: 10
```

1. Direct Access: In JavaScript, call by reference is achieved by passing objects or arrays to functions, allowing the function to directly modify the original object or array.
2. Objects and Arrays: Functions receive a reference to the original object or array, enabling them to alter the properties or elements within it.
3. Memory Efficiency: This method avoids creating copies of the entire object or array, thus saving memory and improving performance.
4. Persistent Changes: Any modifications made to the object or array within the function persist outside the function, reflecting the changes in the original object or array.

Project Rock-Paper-Scissor Game



Rock Paper Scissors Game

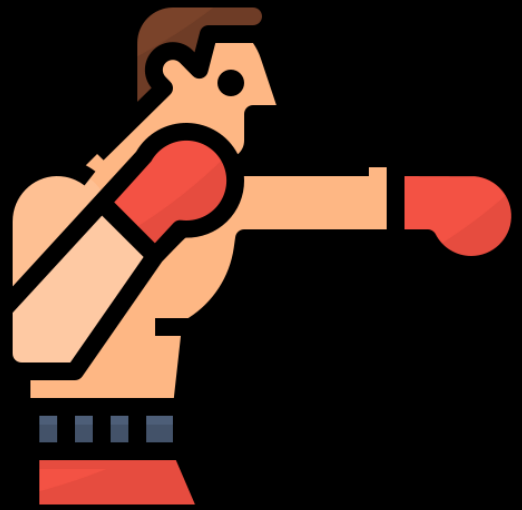
Click on one of the following to play the game:

👊 Rock | ✋ Paper | ✌ Scissors

Create method for showing result into score Object.

Autoboxing

1. Automatic Conversion: JavaScript converts primitives to their corresponding object wrappers when methods or properties are accessed.
2. Temporary Objects: The conversion creates a temporary object that is discarded after the method or property access.
3. Method Access: Enables the use of methods like .length on strings, .toFixed() on numbers, and .toString() on booleans directly on primitive values.
4. Allows properties and methods to be used on primitives.
5. Example: Strings have properties and methods like length, toUpperCase, etc.



Autoboxing

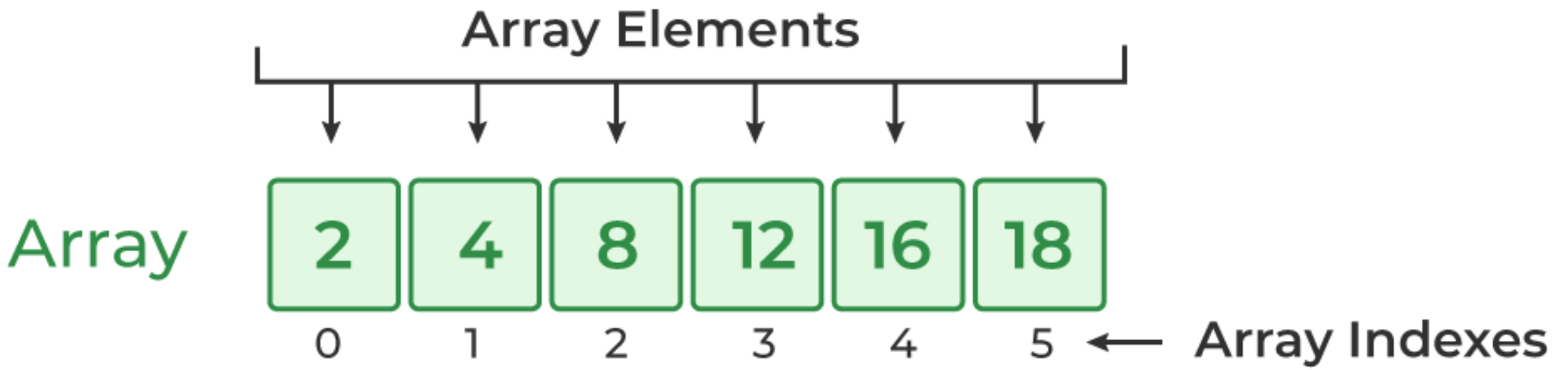
```
// Primitive string
let primitiveString = "Hello, World!";
// JavaScript automatically converts the primitive string to a String object
let length = primitiveString.length;
console.log(`Length of string: ${length}`); // Output: Length of string: 13


// Primitive number
let primitiveNumber = 42;
// JavaScript automatically converts the primitive number to a Number object
let fixedNumber = primitiveNumber.toFixed(2);
console.log(`Fixed number: ${fixedNumber}`); // Output: Fixed number: 42.00


// Primitive boolean
let primitiveBoolean = true;
// JavaScript automatically converts the primitive boolean to a Boolean object
let booleanString = primitiveBoolean.toString();
console.log(`Boolean as string: ${booleanString}`); // Output: Boolean as string: true
```

Practice Exercise

Objects

1. Create object to represent a product from Myntra
2. Create an Object with two references and log changes to one object by changing the other one.
3. Use bracket notation to display delivery-time.
4. Given an object {message: 'good job', status: 'complete'}, use de-structuring to create two variables message and status.
5. Add function isIdenticalProduct to compare two product objects.

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.

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.