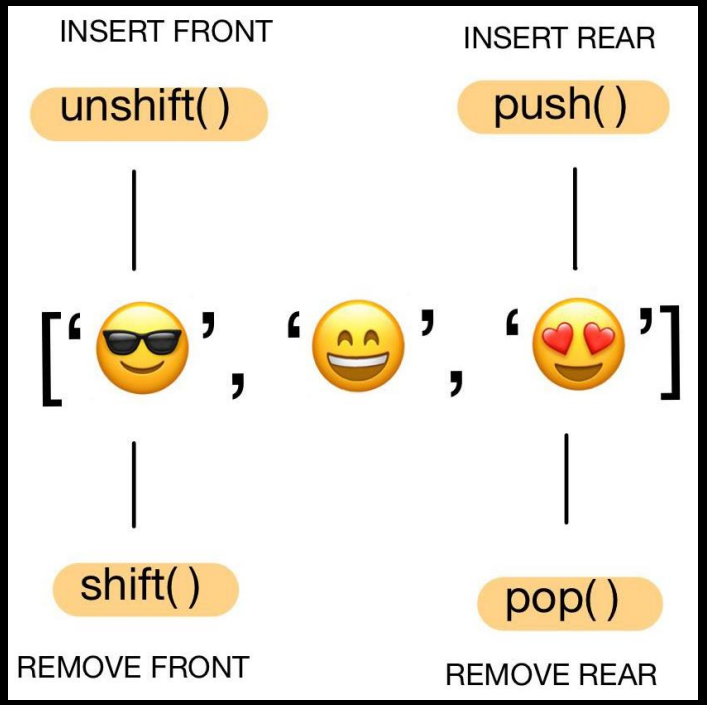
Array

```
const fruits = ["Apple", "Banana", "Cherry"];

// Using while loop
console.log("Using while loop:");
let i = 0;
while (i < fruits.length) {
  console.log(fruits[i]);
  i++;
}

// Using a for loop
console.log("Using for loop:");
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

Array 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.



Array

```
const fruits = ["Apple", "Banana", "Cherry"];

// Add elements to the end of the array using push
fruits.push("Date");
console.log("After push:", fruits); // Output: ["Apple", "Banana", "Cherry", "Date"]

// Remove the last element using pop
const lastFruit = fruits.pop();
console.log("Popped fruit:", lastFruit); // Output: Date
console.log("After pop:", fruits); // Output: ["Apple", "Banana", "Cherry"]

// Add elements to the beginning using unshift
fruits.unshift("Elderberry");
console.log("After unshift:", fruits); // Output: ["Elderberry", "Apple", "Banana", "Cherry"]

// Remove the first element using shift
const firstFruit = fruits.shift();
console.log("Shifted fruit:", firstFruit); // Output: Elderberry
console.log("After shift:", fruits); // Output: ["Apple", "Banana", "Cherry"]
```

Array

```
const numbers = [1, 2, 3, 4, 5];

// Find the first element greater than 3
const firstGreaterThanThree = numbers.find((num) => num > 3);
console.log("First number greater than 3:", firstGreaterThanThree); // Output: 4

// Find the index of the number 3
const index = numbers.indexOf(3);
console.log("Index of 3:", index); // Output: 2
```

for-each Loop

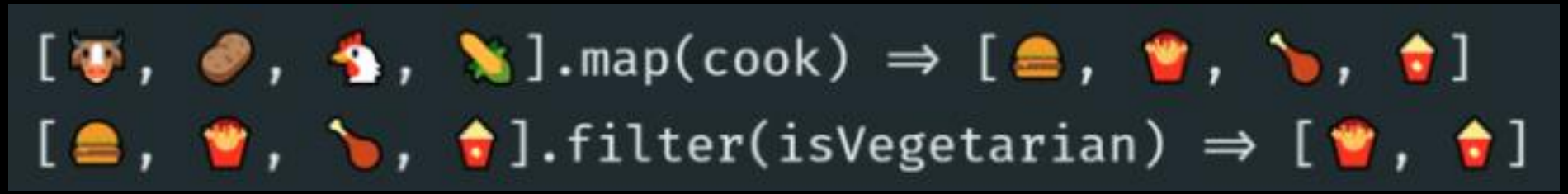
```
let foods = ['bread', 'rice', 'meat', 'pizza'];

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.

Array

```
const fruits = ["Apple", "Banana", "Cherry"];

// Using forEach method
console.log("Using forEach:");
fruits.forEach((fruit) => {
  console.log(fruit);
});
```

Array Advance 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.

Array

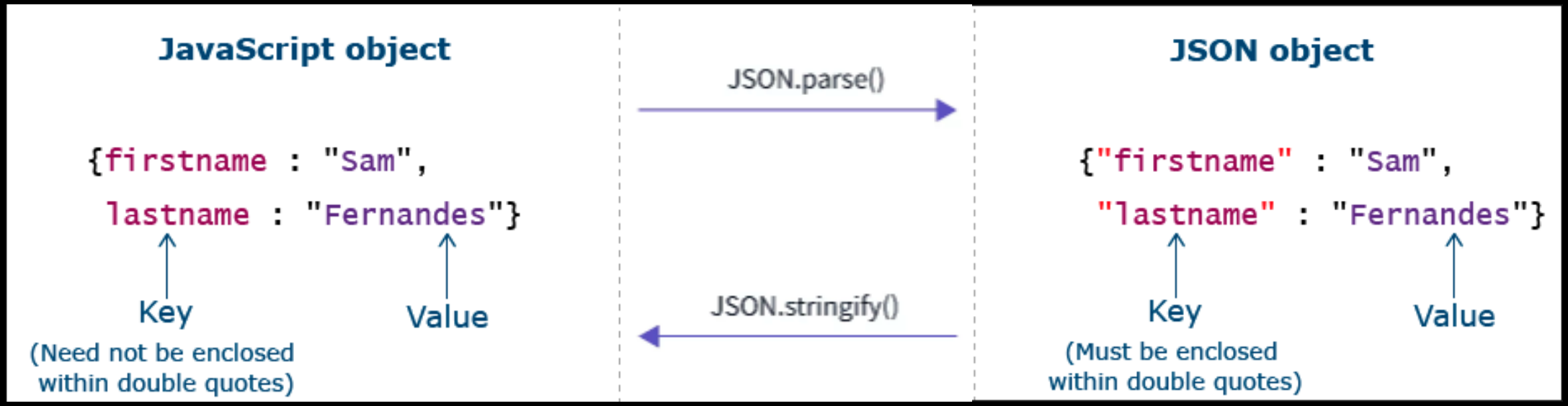
```
const numbers = [1, 2, 3, 4, 5];

// Create a new array with each number doubled
const doubled = numbers.map((num) => num * 2);
console.log("Doubled:", doubled); // Output: [2, 4, 6, 8, 10]

// Filter the array to include only even numbers
const evens = numbers.filter((num) => num % 2 === 0);
console.log("Evens:", evens); // Output: [2, 4]

// Calculate the sum of all numbers
const sum = numbers.reduce((accumulator, currentValue) => accumulator + currentValue, 0);
console.log("Sum:", sum); // Output: 15
```

Practice Exercise

Arrays

1. Create an array of numbers [5,6]. Add 4 at the beginning and 7 at the end of the array.
2. Create a method to return an element at a particular position in the array.
3. Create an array copy using slice method.
4. Using accumulator pattern concatenate all the strings in the given array [‘KG’, ‘Coding’, ‘Javascript’, ‘Course’, ‘is’, ‘Best’]

What is JSON?



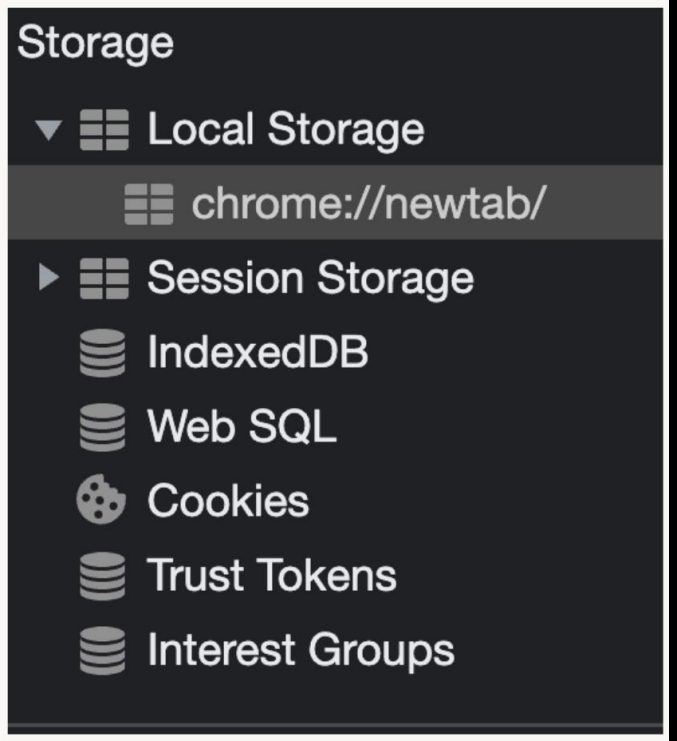
1. JavaScript Object Notation: Not the same as JS object, but similar.
2. Common in network calls and data storage.
3. JSON.stringify() and JSON.parse()
4. Strings are easy to transport over network.
5. JSON requires double quotes. Escaped as \".
6. JSON is data format, JS object is a data structure.

What is JSON?

```
// Define a JavaScript object
const person = {
  name: "Amit",
  age: 28,
  city: "New Delhi",
  skills: ["JavaScript", "Node.js", "React"]
};

// Convert the JavaScript object to a JSON string
const jsonString = JSON.stringify(person);
console.log("JSON String:", jsonString);

// Convert the JSON string back to a JavaScript object
const jsonObject = JSON.parse(jsonString);
console.log("JavaScript Object:", jsonObject);

// Access properties from the parsed JavaScript object
console.log("Name:", jsonObject.name);
console.log("Age:", jsonObject.age);
console.log("City:", jsonObject.city);
console.log("Skills:", jsonObject.skills.join(", "));
```

Browser 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")
);
```



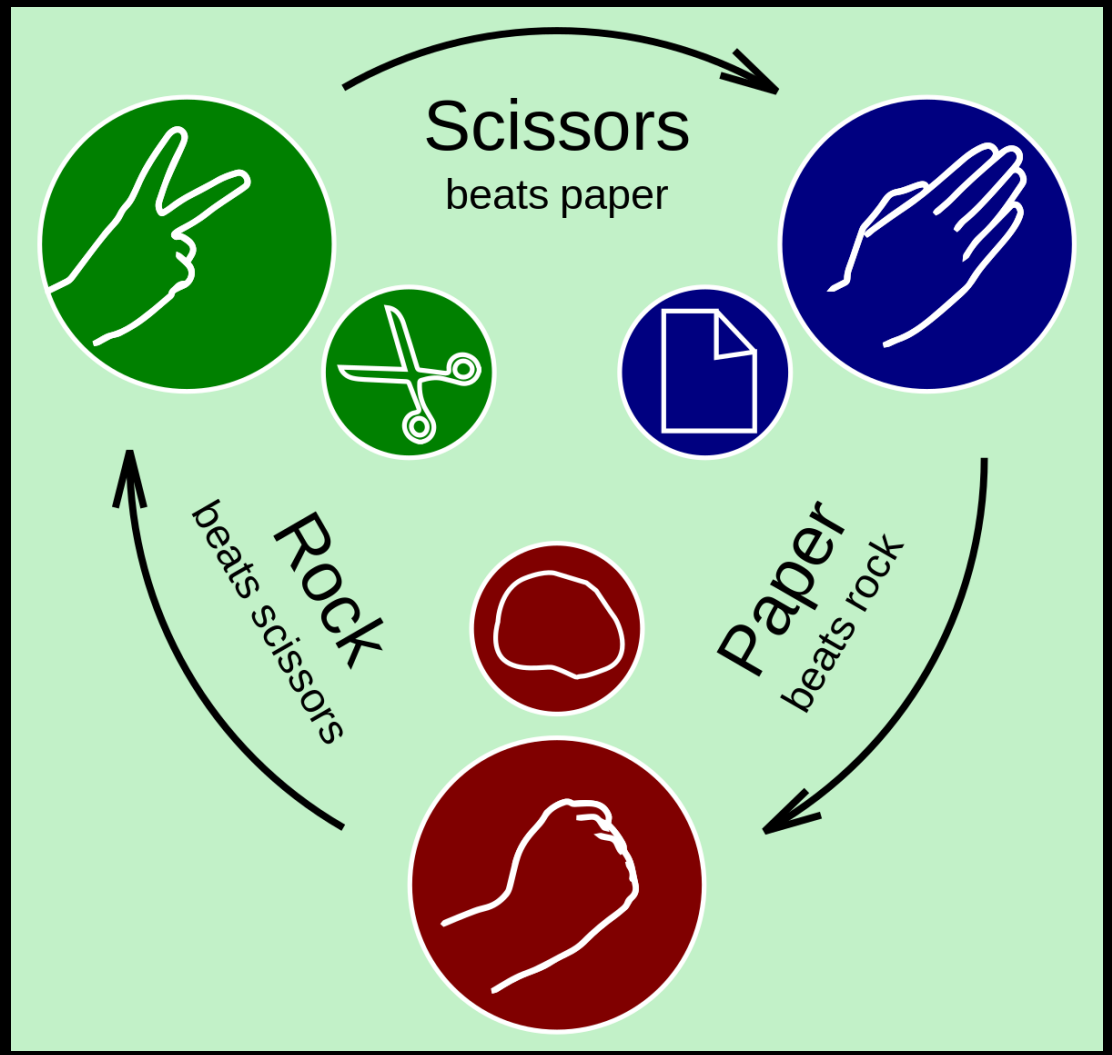
1. Persistent data storage in the browser.
2. setItem: Stores data as key-value pairs.
3. Only strings can be stored.
4. getItem: Retrieves data based on key.
5. Other Methods: localStorage.clear(), removeItem().
6. Do not store sensitive information. Viewable in storage console.

Browser Local Storage

```
// Store data in localStorage using setItem
localStorage.setItem("name", "Amit");
localStorage.setItem("age", "28");
localStorage.setItem("city", "New Delhi");

// Retrieve data from localStorage using getItem
const name = localStorage.getItem("name");
const age = localStorage.getItem("age");
const city = localStorage.getItem("city");

console.log("Name:", name);  // Output: Amit
console.log("Age:", age);    // Output: 28
console.log("City:", city);  // Output: New Delhi
```

```
// Remove a specific item from localStorage
localStorage.removeItem("age");

// Attempt to retrieve the removed item
const removedAge = localStorage.getItem("age");
console.log("Removed Age:", removedAge); // Output: null

// Clear all items from localStorage
localStorage.clear();

// Attempt to retrieve data after clearing localStorage
const clearedName = localStorage.getItem("name");
console.log("Cleared Name:", clearedName); // Output: null
```

Project Rock-Paper-Scissor Game



Rock Paper Scissors Game

Click on one of the following to play the game:

👊 Rock | ✋ Paper | ✌ Scissors

1. Score will survive browser refresh.
2. Add Reset Button To clear or reset stored data.