JSON.stringify():

In JavaScript, the JSON.stringify() method is used to convert JavaScript objects or values into JSON strings. This method is particularly useful when you need to send data to a server or store it in a file, as JSON is a common format for data interchange.

Syntax:

```
JSON.stringify(value [, replacer [, space]])
```

value:

The JavaScript value to be converted into a JSON string.

replacer (optional):

A function or array that can be used to filter and modify the properties of the JavaScript object before they are included in the JSON string.

space (optional):

A string or number that specifies the indentation for nested levels in the resulting JSON string. This parameter is used for pretty-printing the JSON output.

Example 1: Converting JavaScript Object to JSON String:

```
const person = { name: 'John', age: 30, city: 'New York' };
const jsonString = JSON.stringify(person);

console.log(jsonString); // Output:
{"name":"John","age":30,"city":"New York"}
```

In this example, we have a JavaScript object person. We use JSON.stringify() to convert this object into a JSON string jsonString.

Example 2: Pretty-Printing JSON String:

```
const person = { name: 'John', age: 30, city: 'New York' };
const jsonString = JSON.stringify(person, null, 2);

console.log(jsonString);
/* Output:
{
  "name": "John",
  "age": 30,
  "city": "New York"
}
*/
```

Here, we pass 2 as the third argument to JSON.stringify(), indicating that we want to use a two-space indentation for pretty-printing the JSON string.

Example 3: Excluding Properties with Replacer Function:

```
const person = { name: 'John', age: 30, city: 'New York' };
const jsonString = JSON.stringify(person, (key, value) => {
    if (key === 'age') {
        return undefined; // Exclude age property from JSON string
    }
    return value;
});

console.log(jsonString); // Output: {"name":"John","city":"New
York"}
```

In this example, we use a replacer function to exclude the age property from the resulting JSON string.

Example 4: Converting JavaScript Array to JSON String:

```
const numbers = [1, 2, 3, 4, 5];
const jsonString = JSON.stringify(numbers);

console.log(jsonString); // Output: [1,2,3,4,5]
```

In this example, we have a JavaScript array numbers. We use JSON.stringify() to convert this array into a JSON string jsonString.

Example 5: Handling Circular References:

```
const obj = { a: 1 };
obj.b = obj; // Create circular reference

try {
  JSON.stringify(obj);
} catch (error) {
  console.error("Circular reference detected:", error.message);
}
```

If there are circular references within the object graph being stringified, JSON.stringify() will throw a TypeError with the message "Converting circular structure to JSON".

These examples demonstrate different use cases of JSON.stringify() method, including converting JavaScript objects and arrays into JSON strings, pretty-printing JSON, excluding properties, and handling circular references.