Advanced Constructs: Advanced Function Concepts

Relevel
by Unacademy

Topics Covered

- Basics of JSON
- Basic problems on JSON and functions
- Creation of JSON objects
- Accessing json objects using key
- Iterating through an JSON object
- NESTED objects
- Comparing different JSON Objects
- Higher Order Functions
- Composability
- Recursion
- Arrow Function
- Closures

What is a JSON?

What is a JSON??

JavaScript Object Notation is a format to store and transfer data between server and applications.

It has a specific structure and is easy to create.

Writing convention-

{Key:value}

Key is written in double quotes(“ ”) followed by semicolon (‘:’) ans a value.

```
{"firstName":"John", "lastName":"Doe"}
```

What is a JSON?

For the given JSON object print the first name of the student.

```
var student = {
  "Roll_No": 12,
  "First_Name": "John",
  "Last_Name": "Doe",
  "Class": 12,
  "Contact No": 9999999999,
  "Email": "john.doe@gmail.com",
  "Address": "A-123,New Street,LA"

}
```

```
var student = {
  "Roll_No": 12,
  "First_Name": "John",
  "Last_Name": "Doe",
  "Class": 12,
  "Contact No": 9999999999,
  "Email": "john.doe@gmail.com",
  "Address": "A-123,New Street,LA"

}
console.log(student.First_Name);
```

Programming Example

Create a JSON object and print all of its objects using properties

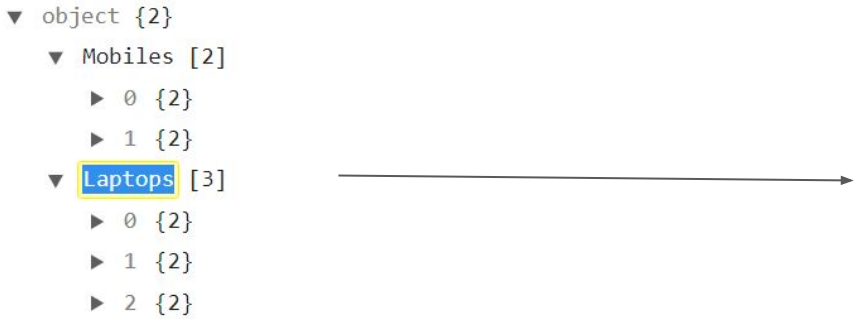
```
var Cars = [{
    brand: "BMW",
    model: "X3"
  }, {
    brand: "BMW",
    model: "X5"
  },
  {
    brand: "Audi",
    model: "A6"
  },
  {
    brand: "Audi",
    model: "R8"
  }
];

for (var car of Cars) {
  document.write(car.brand + " - " + car.model + "<br />");
}
```

Programming Example

Find student with max marks in a provided JSON object

```
var Students = [{
    "RollNo": 2,
    "Name": "Jon Doe",
    "Class": 12,
    "Marks": 93
  },
  {
    "Roll No": 1,
    "Name": "Annie",
    "Class": 12,
    "Marks": 89
  },
  {
    "Roll No": 3,
    "Name": "Susane",
    "Class": 12,
    "Marks": 93
  }
];
let max = 0;
let topper;
for (let i in Students) {
  if (Students[i].Marks > max) {
    max = Students[i].Marks;
    topper = i;
  }
}
console.log(Students[topper]);
```

Nested JSON object from a Given structure




```
let devices = {
  "Mobiles": [{
      "model": "iphone 13",
      "price": "$ 799"
    },
    {
      "model": "iphone 12",
      "price": "$ 699"
    }
  ],
  "Laptops": [{
      "model": "Mac Air M1",
      "price": "$ 899"
    },
    {
      "model": "Macbook pro m1",
      "price": "$ 1299"
    },
    {
      "model": "Macbook pro 14",
      "price": "$ 1849"
    }
  ]
};
for (let product in devices.Laptops) {
  console.log(devices.Laptops[product]);
}
```

Compare two different JSON objects

```
var isEqual = (p1, p2) => {
  keys1 = Object.keys(p1);
  keys2 = Object.keys(p2);
  return keys1.length === keys2.length && Objec
t.keys(p1).every(key => p1[key] == p2[key]);
}

var p1 = {
  name: "Aman",
  age: 23,
  country: "India"
};
var p2 = {
  age: 23,
  name: "Aman",
  country: "India"
};
console.log(isEqual(p1, p2));
```

Advanced Functions

Higher Order Functions: Functions which take another function as an argument or return another function are called higher order functions.

Benefits of using HOF:

- Re usability of the code
- Easier to understand

Programming Example

```
function updating(arr, operation) {
    const updated = []
    for (let element of arr) {
        updated.push(operation(element))
    }
    return updated
}

function double(num) {
    return num*2
}

console.log(updating([1, 2, 3], double));
```

Output: [2, 4, 6]

Programming Example

Find cube of a number using high order function

```
const callback = (n) => {
  return n ** 2
}

function cube(callback, n) {
  return callback(n) * n
}
console.log(cube(callback, 4))
```

Programming Example

Convert values of an array to uppercase

```
1  const names = ['apple', 'google', 'microsoft', 'meta']
2  const namesToUpperCase = names.map((name) => name.toUpperCas
   e())
3  console.log(namesToUpperCase)
4
```

Composability

Creating a complex function by combining multiple simple functions is called composability.

It is similar to mathematical functional f(g(x)).Here the output of g(x) is passed to f.

```
compose = (fn1, fn2) => value =>
fn2(fn1(value))
```

```
1  const multiply20 = (price) => price * 20;  //200
2  const divide5 = (price) => price / 5;     //40
3  console.log(divide5(multiply20(10)));
4
```

Programming Example

Find square of even numbers in an array

```
let numbers = [2, 3, 6, 8, 7];
console.log(findSquare(isEven(numbers)));

function isEven(arr) {
  let evenarr = [];
  for (num in arr) {
    if (arr[num] % 2 == 0) {
      evenarr.push(arr[num]);
    }
  }
  return evenarr;
}

function findSquare(arr) {
  for (num in arr) {
    arr[num] *= arr[num];
  }
  return arr;
}
```

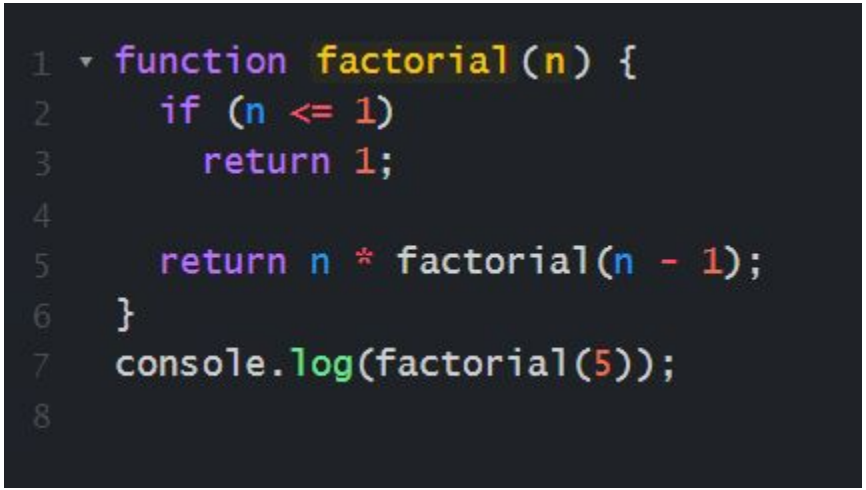
Recursion

What is recursion?
When a function calls itself multiple times, it is known as recursion.

```
function countDown(fromNumber) {
    console.log(fromNumber);

    let nextNumber = fromNumber - 1;

    if (nextNumber > 0) {
        countDown(nextNumber);
    }
}
countDown(3);
```

Programming Example

Sum of first n natural numbers



Arrow Function

What is an arrow function?

An arrow function allows us to create functions in a cleaner way compared to regular functions by using =>.

The difference here is that we store the entire function declaration in a variable.

Eg:

```
let add = (x, y) => x + y;

console.log(add(10, 20)); // 30;
```

Regular Function to Arrow Function

```
function ask(question, yes, no) {
  if (confirm(question)) yes();
  else no();
}

ask(
  "Confirm?",
  function() {
    alert("You confirmed.");
  },
  function() {
    alert("You canceled.");
  }
);
```

```
function ask(question, yes, no) {
  if (confirm(question)) yes();
  else no();
}

ask(
  "Confirm?",
  () => alert("You confirmed."),
  () => alert("You canceled.")
);
```

Closure

What are closures?

A closure is the combination of a function enclosed with references to its surrounding state .

A closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

Counter using Closures

```
function Counter() {
    var counter = 0;

    function IncreaseCounter() {
        return counter += 1;
    }

    return IncreaseCounter;
}

var counter = Counter();
console.log(counter());
console.log(counter());
console.log(counter());
```

Practice Problems

Qus 1: Create a JSON object for a showroom of cars, who sells cars of different brands.

JSON shall contain various details such as: car brand name, car models, price and few basic details.

Also write a program to print details of most expensive car

Qus 2: Write a program to sort a given data using recursive approach.

Qus 3: Create an JSON object for grades of a class having structure

{"name": 'John', "grade": 8, "sex": 'M'},

Also write a program to find classroomAverage ,boysAverage ,girlsAverage,highestGrade and lowestGrade

Practice Problems

Qus 4: Write a program using recursion to print a triangle of length n.

Input: 5

Output:

```
*
* *
* * *
* * * *
* * * * *
```

Thank You