JavaScript
Objects

Naveen Pete

Introduction

- You would like to store person details

```
// Using array
var person = [‘Hari’, 24, ‘Bengaluru’];

// Retrieve city
person[2];

// The order of city is changed
var anotherPerson = [‘Krish’, ‘Mumbai’, 31];

// This does not provide the city!
anotherPerson[2];
```

Introduction

- You can create a person object instead

```
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru'
};

var anotherPerson = {
  name: 'Krish',
  city: 'Mumbai',
  age: 31
};
```

  - Curly braces {} instead of square brackets []
  - Every single item in the object is a key-value pair
  - Unlike arrays, objects have no order

Retrieving Data

- Two choices
  - Bracket notation
  - Dot notation

```
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru'
};

// similar to arrays
console.log(person["name"]);

// dot notation
console.log(person.name);
```

Retrieving Data

- Two choices
  - Bracket notation
  - Dot notation

```
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru'
};

// similar to arrays
console.log(person["name"]);

// dot notation
console.log(person.name);
```

  - You cannot use dot notation if the property name
    - starts with a number
    - contains a space within it

Updating Data

- Similar to array, access a property and reassign the value to it

```
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru'
};

// bracket notation
person["city"] = 'Mysuru';

// dot notation
person.age += 1;
```

Creating Objects

```
// all at once
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru'
};

// make an empty object and assign properties
var person = {};
person.name = 'Hari';
person.age = 24;
person.city = 'Bengaluru';

// using new operator
var person = new Object();
person.name = 'Hari';
person.age = 24;
person.city = 'Bengaluru';
```

Objects

- Can hold any type of data

```
var person = {
  name: 'Hari',
  age: 24,
  isEmployed: true,
  friends: ['Krish', 'Shiv'],
  address: {
    houseNo: 10
    area: 'Jayanagar'
    city: 'Bengaluru',
  }
};
```

Nested Objects and Arrays

```
var posts = [
  {
    title: 'p1',
    body: 'p1 body',
    author: 'p1 author',
    comments: [
      { text: 'p1 comment 1', user: 'p1 c1 user' },
      { text: 'p1 comment 2', user: 'p1 c2 user' },
    ]
  },
  {
    title: 'p2',
    body: 'p2 body',
    author: 'p2 author',
    comments: [
      { text: 'p2 comment 1', user: 'p2 c1 user' },
      { text: 'p2 comment 2', user: 'p2 c2 user' },
    ]
  }
];
```

Methods

- A method
  - Is a function that is defined as a property inside an object
  - Is an action that can be performed on an object

```
var person = {
  name: 'Hari',
  age: 24,
  city: 'Bengaluru',
  friends: ['Krish', 'Shiv'],
  greet: function() {
    console.log('Hi');
  }
};

person.greet();
```

  - Helps you keep your code organized so your can group things logically together
  - Avoids namespace collision

this keyword

- Within a function definition, ‘this’ refers to the owner of the function

```
var person = {
    firstName: "John",
    lastName : "Doe",
    id       : 5566,
    fullName : function() {
        return this.firstName + " " + this.lastName;
    }
};
```

  - In this example, this refers to the ‘person’ object, because it owns ‘fullName()’ method

this keyword

- When used alone, ‘this’ refers to the Global object
- In a browser, the Global object is the ‘Window’ object

```
var x = this;
```

- When used in a function, this refers to the Global object, that is, ‘Window’ object

```
function myFunction() {
    return this;
}
```

Q & A

- Thank you!