Date : 24 - 06 - 2020
Morning Session : 9am – 11 PM
By ~ Sundeep Charan Ramkumar Today

# Topics: ES6 Day 3

```
function House(type, address) {
  this.type = type;
  this.address = address;
}

House.prototype.paint = function () {
  return `The house of type ${this.type} is being painted`;
};

const bungalow = new House("villa", {
  city: "Coimbatore",
  state: "Tamilnadu",
  country: "India"
});

console.log(bungalow);
```

The class contains the **Constructors** and **Functions**. The Constructors take responsibility for allocating memory for the objects of the class. The function takes the responsibility of the action of the objects. Combining these two **Constructor** and **Function** to make the **Class**.

```
// ES6
class House {
  constructor(type, address) {
    this.type = type;
    this.address = address;
  }

  paint() {
    return `The house of type ${this.type} is being painted`;
  }
}

const bungalow = new House("villa", {
  city: "Coimbatore",
  state: "Tamilnadu",
  country: "India"
});

console.log(bungalow);
```

**Class inheritance:** **Inheritance has the courage to create entities from existing entities.**

**parent class/super class:** The class extended to create a new class is known as a parent class or superclass.

**child/subclasses:** The classes are newly created are known as child or subclass. Subclass inherit all the properties from parent class except constructor

**Super Keyword:** **This keyword helps child class to invoke the parent class data.**

```
class House {
  constructor(type, address) {
    this.type = type;
    this.address = address;
  }
  paint() {
    return `The house of type ${this.type} is being painted`;
  }
}
class Bungalow extends House {
  constructor(
    isGardenAvailable,
    isSolarPanelAvailable,
    isSwimminPoolAvailable,
    type,
    address
  ) {
    super(type, address);
    this.isGardenAvailable = isGardenAvailable;
    this.isSolarPanelAvailable = isSolarPanelAvailable;
    this.isSwimminPoolAvailable = isSwimminPoolAvailable;
  }
}
const bungalow = new Bungalow(true, true, true, "Bungalow", {
  city: "Coimbatore",
  state: "Tamilnadu",
  country: "India"
});

console.log(bungalow);
```

**getters and setters.** It is smart to use **getters** and **setters** for the properties, especially if you want to do something special with the value before returning them, or before you set them. To add **getters** and **setters** in the class, use the **get** and **set** keywords.

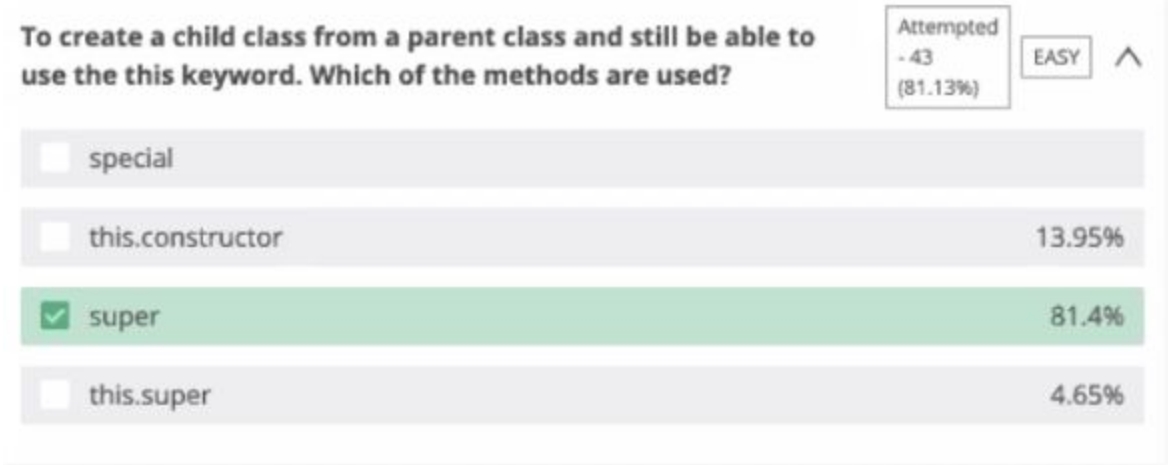
```
  get getIsGardenAvailable() {
    return this.isGardenAvailable;
  }

  set setIsGardenAvaialable(value) {
    this.isGardenAvailable = value;
  }

  toggleLights() {
    return `Switching off the lights`;
  }
}
```

**MCQ1:**

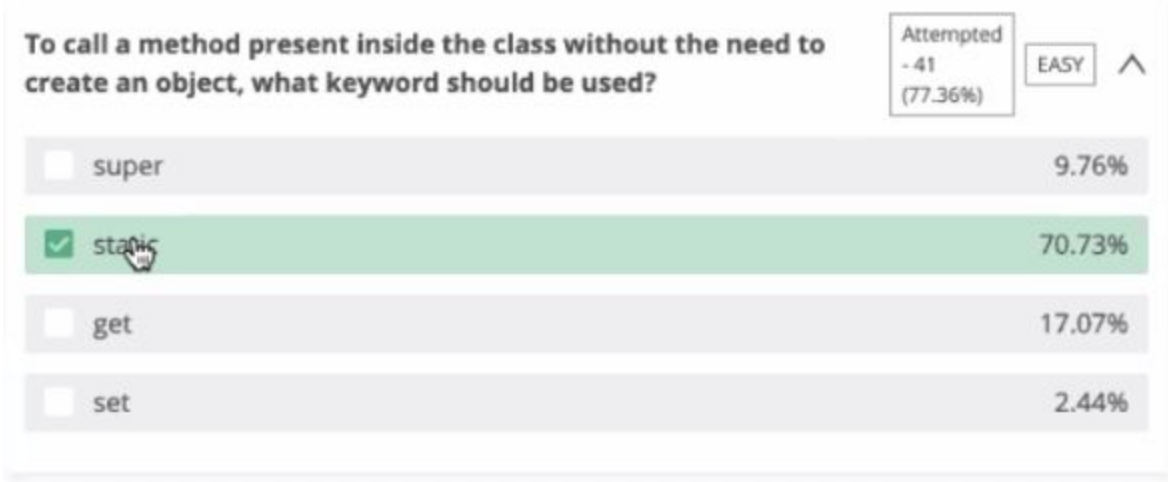

**MCQ2:**



**MCQ3:**


Getters and setters are invoked as functions. True or False?
Attempted - 42 (79.25%)
EASY
True
47.62%
False
52.38%

**MCQ4:**


To call a method present inside the class without the need to create an object, what keyword should be used?
Attempted - 41 (77.36%)
EASY
super
9.76%
static
70.73%
get
17.07%
set
2.44%