**Challenge1**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge1.js"></script>

</body>

</html>

**Challenge1.js**

/\*\*

 \* CHALLENGE 7-1

 \*

 \* Insert new prototype "Vehicle.prototype" to the existing prototype chain

 \*

 \* CURRENT CHAIN

 \* CivilPlane.prototype > Airplane.prototype

 \* > Object.protoype

 \*

 \* UPDATED CHAIN

 \* CivilPlane.prototype > Airplane.prototype

 \* > Vehicle.prototype > Object.protoype

 \*

 \* Vehicle.prototype should set two additional

 \* properties:

 \* 1. maxSpeed

 \* 2. weight

 \*

 \* It should also contain method "vehicleInfo"

 \* available for any object down the prototype chain

 \*/

function Vehicle(props){

    this.maxSpeed=props.maxSpeed;

    this.weight=props.weight;

}

Vehicle.prototype.vehicleInfo = function(){

    console.log(`Max Speed  of vehicle is ${this.maxSpeed}`);

    console.log(`Weight of the vehicle is ${this.weight}`);

}

 function Airplane(props) {

    Vehicle.call(this,props);

    this.wingspan = props.wingspan;

    this.maxRangeOfFlight = props.maxRangeOfFlight;

  }

  Airplane.prototype= Object.create(Vehicle.prototype);

  Airplane.prototype.constructor= Airplane;

  Airplane.prototype.airplaneInfo = function() {

    console.log(

      `Wingspan of the airplane is ${

        this.wingspan

      } and maximal range of flight is ${

        this.maxRangeOfFlight

      }`

    );

  };

  function CivilPlane(props) {

    Airplane.call(this, props);

    this.numberOfSeats = props.numberOfSeats;

  }

  CivilPlane.prototype = Object.create(Airplane.prototype);

  CivilPlane.prototype.constructor = CivilPlane;

  CivilPlane.prototype.seatsInfo = function() {

    console.log(this); // instance of the prototype

    console.log(

      `Number of seats in the plane is ${this.numberOfSeats}`

    );

  };

  CivilPlane.prototype.modifySeatsNumber = function(

    newSeatsQty

  ) {

    this.numberOfSeats = newSeatsQty;

  };

  const propsForSmallPlane = {

    numberOfSeats: 4,

    wingspan: 20,

    maxRangeOfFlight: 1000,

    weight : 15,

    maxSpeed : 800

  };

  const smallPlane = new CivilPlane(propsForSmallPlane);

  const propsForLargePlane = {

    numberOfSeats: 250,

    wingspan: 60,

    maxRangeOfFlight: 3500,

    weight : 40,

    maxSpeed : 900

  };

  const largePlane = new CivilPlane(propsForLargePlane);

  /\*\*

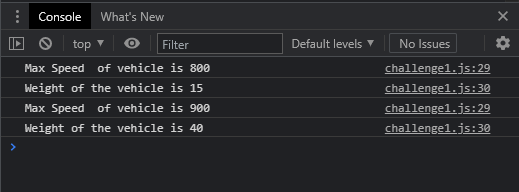
   \* VERIFICATION

   \*/

  smallPlane.vehicleInfo(); // Max speed of the vehicle is 800 and weight is 15

  largePlane.vehicleInfo(); // Max speed of the vehicle is 900 and weight is 40

**Result**

****

**Challenge2**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge2.js"></script>

</body>

</html>

**Challenge2.js**

 function Product(props) {

    this.price = props.price;

  }

  Product.prototype.priceInfo = function(){

    console.log(`Price of the product is ${this.price}`);

  }

  function ElectricDevice(props) {

    Product.call(this,props);

    this.energyEfficiencyClass = props.energyEfficiencyClass;

  }

  ElectricDevice.prototype = Object.create(Product.prototype);

  ElectricDevice.prototype.constructor=ElectricDevice;

  ElectricDevice.prototype.energyInfo = function() {

    console.log(`Energy Efficiency Class is ${this.energyEfficiencyClass}`);

  }

  function TV(props) {

    ElectricDevice.call(this, props);

    this.model = props.model;

    this.diagonal = props.diagonal;

  }

  TV.prototype = Object.create(ElectricDevice.prototype);

  TV.prototype.constructor = TV;

  const propsForMyTv = {

    model: "A1620",

    price: 1200,

    energyEfficiencyClass: "A+",

    diagonal: 42

  };

  const myTV = new TV(propsForMyTv);

  /\*\*

   \* VERIFICATION

   \*/

  console.log(myTV);

  /\* {

    model: "A1620",

    price: 1200,

    energyEfficiencyClass: "A+",

    diagonal: 42,

    \_\_proto\_\_: ...

  } \*/

  myTV.energyInfo(); // "Energy Efficiency Class is A+"

  myTV.priceInfo(); // Price of the product is 1200

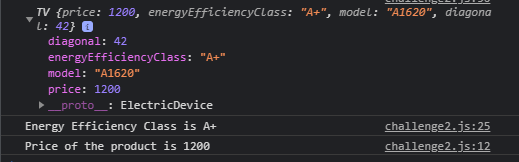
  myTV instanceof TV; // true

  myTV instanceof ElectricDevice; // true

  myTV instanceof Product; // true

  myTV instanceof Object; // true

**Result**

****

**Challenge3**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge3.js"></script>

</body>

</html>

**Challenge3.js**

 class Product {

    constructor({ price }) {

      this.price = price;

    }

    priceInfo() {

      console.log(`Price of the product is ${this.price}`);

    }

  }

  class ElectricDevice extends Product {

    constructor(props) {

      super(props);

      this.energyEfficiencyClass =

        props.energyEfficiencyClass;

    }

    energyInfo() {

      console.log(

        `Energy Efficiency Class is ${

          this.energyEfficiencyClass

        }`

      );

    }

  }

  class TV extends ElectricDevice {

    constructor(props) {

      super(props);

      this.model = props.model;

      this.diagonal = props.diagonal;

    }

  }

  const propsForMyTv = {

    model: "A1620",

    price: 1200,

    energyEfficiencyClass: "A+",

    diagonal: 42

  };

  const myTV = new TV(propsForMyTv);

  /\*\*

   \* VERIFICATION

   \*/

  console.log(myTV);

  myTV.energyInfo(); // "Energy Efficiency Class is A+"

  myTV.priceInfo(); // Price of the product is 1200

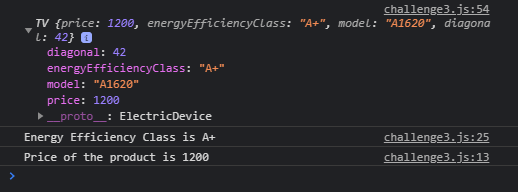
  myTV instanceof TV; // true

  myTV instanceof ElectricDevice; // true

  myTV instanceof Product; // true

  myTV instanceof Object; // true

**Result**

****

**Challenge4**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge4.js"></script>

</body>

</html>

**Challenge4.js**

/\*\*

 \* CHALLENGE 7-4: TASK

 \*

 \* Create new class "ExtendedArray" that should extend built-in "Array".

 \*

 \* Add two custom methods to the new class:

 \* 1. "sum" - it should return sum of all elements in the array

 \* 2. "onlyNumbers" - it should return new array that will contain only numbers from the source array

 \*

 \* Create several instances of the new "ExtendedArray" class and test both methods "sum" and "onlyNumbers"

 \*/

 class ExtendedArray extends Array {

    sum() {

      return this.reduce((sum, el) => sum + el);

    }

    onlyNumbers() {

      return this.filter(el => typeof el === "number");

    }

  }

  const firstInstance = new ExtendedArray(2, 3, 6, 10);

  console.log(firstInstance.sum());//21

  console.log(firstInstance.onlyNumbers()); // [2, 3, 6, 10]

**Result**

****

**Challenge5**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge5.js"></script>

</body>

</html>

**Challenge5.js**

/\*\*

 \* CHALLENGE 7-5: TASK

 \*

 \* Create new class "CustomArray" that should extend built-in "Array".

 \*

 \* Add one custom method "customPush" to the new class.

 \* This method will have one parameter "newElement".

 \*

 \* It should perform following actions:

 \* 1. Add new element to the existing array

 \* (don't use "push" method for this)

 \* 2. Modify "length" property of the array (increment it)

 \* 3. Log following line to the console:

 \* "New element <ELEMENT> was just added to the array"

 \*

 \* Create instance of the new "CustomArray" class and test new method "customPush" and compare it with "push"

 \*

 \* What will happen if name of the custom method in the "CustomArray" class will be also "push" instead of "customPush"?

 \* Try this.

 \*/

 class CustomArray extends Array {

    customPush(newElement) {

      this[this.length] = newElement;

      console.log(

        `New element ${newElement} was just added to the array`

      );

    }

  }

  const firstInstance = new CustomArray();

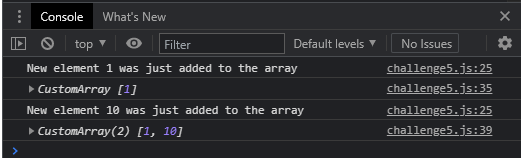
  firstInstance.customPush(1);

  console.log(firstInstance); // [1]

  firstInstance.customPush(10);

  console.log(firstInstance); // [1, 10]

**Result**

****

**Challenge6**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Classes, Prototypes and Function Constructors</title>

</head>

<body>

    <script src="challenge6.js"></script>

</body>

</html>

**Challenge6.js**

/\*\*

 \* CHALLENGE 7-6 - SOLUTION

 \*

 \* Add one more button with color "secondary".

 \* Also change styles of the existing and new buttons

 \* using property "variant".

 \*

 \* You can use one of the following values:

 \* "text", "flat", "outlined", "contained", "raised", "fab", "extendedFab"

 \*/

 class MyButton extends React.Component {

    constructor(props) {

      super(props);

      this.state = { clickedQty: 0 };

    }

    render() {

      return React.createElement(

        window["material-ui"].Button,

        {

          variant: this.props.variant,

          color: this.props.color,

          onClick: () =>

            this.setState({

              clickedQty: ++this.state.clickedQty

            })

        },

        this.state.clickedQty > 0

          ? `Button was clicked ${

              this.state.clickedQty

            } times`

          : "Click me!!!"

      );

    }

  }

  // button1

  const button1 = React.createElement(MyButton, {

    color: "primary",

    variant: "outlined"

  });

  const buttonContainer1 = document.getElementById("button1");

  ReactDOM.render(button1, buttonContainer1);

  // button2

  const button2 = React.createElement(MyButton, {

    color: "secondary",

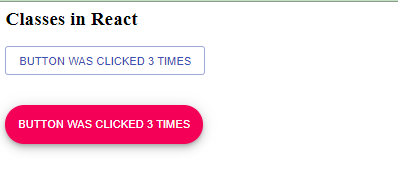
    variant: "extendedFab"

  });

  const buttonContainer2 = document.getElementById("button2");

  ReactDOM.render(button2, buttonContainer2);

**Result**

****