Array

Task 1

|  |
| --- |
| let array = new Array(1,2,3,4,6);  let sum  = 0;  array.forEach(el => {       sum+=el;  });  const med = Math.round((sum/array.length) \* 10) /10;  console.log(med); |

Another variant is with for( let i=0; i<array.length; i++){ sum+=array[i];}

With random array

|  |
| --- |
| var lenghtArray = Math.random() \* 10 + 1;  let array = new Array();  for(let i=0; i<lenghtArray; i++){      array.push(Math.floor(Math.random() \* 10 + 1));  }  console.log(array);  let sum  = 0;  array.forEach(el => {       sum+=el;  });  const med = Math.round((sum/array.length) \* 10) /10;  console.log(med); |

|  |
| --- |
| var lenghtArray = Math.random() \* 10 + 1;  let array = new Array();  for(let i=0; i<lenghtArray; i++){      array.push(Math.floor(Math.random() \* 10 + 1));  }  console.log(array);  let sum = array.reduce((accumulator, currentValue) => accumulator + currentValue, 0);  const med = (sum / array.length).toFixed(1);  console.log(med); |

Task 2

|  |
| --- |
| let array = new Array(1,2,3,4,6);  let newArray = new Array();  let temp = 0;  for(let i=0; i<array.length; i++){      if(i === 0){          temp = array[i];      }else{          newArray.push(array[i]);      }  }  newArray.push(temp);  console.log(newArray); |

|  |
| --- |
| let array = [1, 2, 3, 4, 6];  let firstElement = array.shift();  array.push(firstElement);  console.log(array); |

With simple for but without a new array.

|  |
| --- |
| var lengthArray = Math.random() \* 10 + 1;  let array = new Array();  for(let i=0; i<lengthArray; i++){      array.push(Math.floor(Math.random() \* 10 + 1));  }  console.log(array);  let temp = array[0];  for (let i = 0; i < lengthArray - 1; i++) {      array[i] = array[i + 1];  }  array[array.length-1] = temp;  console.log(array); |

Task 3

|  |
| --- |
| function modify(*people*) {      var arr = [];      people.forEach( *person* =>{        arr.push(`Name: ${person.name}, age: ${person.age} years`)      });      return arr;  }  let people = [    { name: "John", age: 23 },    { name: "Ann", age: 25 },  ];  const sum = modify(people);  console.log(sum); |

With simple for

|  |
| --- |
| function modify(people) {      var arr = [];      for(let i=0; i<people.length; i++){          arr.push(`Name: ${people[i].name}, age: ${people[i].age} years`)      }      return arr;       //return people.map(person => `Name: ${person.name}, age: ${person.age} years`);  }  let people = [    { name: "John", age: 23 },    { name: "Ann", age: 25 },  ];  const sum = modify(people);  console.log(sum); |

Objects

Task 1

|  |
| --- |
| function getSum(*sal*) {    let sum = 0;      for (let person in sal) {        sum+=sal[person];      }      return sum;  }  let salaries = {    John: 100,    Ann: 160,    Pete: 130  }  const sum = getSum(salaries);  console.log(sum); |

Task 2

|  |
| --- |
| let menu = {    width: 200,    height: 300,    title: "My name"  }  function multiplyNumeric(obj) {      for(let element in obj){          if(typeof obj[element] === "number"){              obj[element] \*= 2;          }      }    }  multiplyNumeric(menu);  console.log(menu); |

Task 3

|  |
| --- |
| let ladder = {    step: 0,    up: function() {      console.log("Go up");      this.step++;    },    down: function() {      console.log("Go down");      if(this.step != 0)          this.step--;    },    showStep: function(){      console.log("Current step is " + this.step);    };  ladder.showStep();  ladder.up();  ladder.up();  ladder.showStep();  ladder.down();  ladder.showStep(); |

OOP

Task 1

|  |
| --- |
| class Worker{    constructor(name, surname, rate, days) {      this.name = name;      this.surname = surname;      this.rate = rate;      this.days = days;    }    getSalary(){      return this.rate \* this.days;    }    getFullName(){      return this.name + " " + this.surname;    }  }  class Boss extends Worker{      constructor( name, surname, rate, days, workers) {          super(name, surname, rate, days);          this.workers = workers;      }      getSalary(){          return super.getSalary() \* this.workers;    }  }  const worker = new Worker("Jimmy", "Smith", 23, 2);  const boss = new Boss("Jimmy", "Smith", 5, 2, 3);  console.log(worker.getSalary());  console.log(boss.getSalary()); |

**Regular Expressions**

**Task 1**

|  |
| --- |
| function isValidURL(url) {    const urlRegex = /^(http|https):\/\/[a-zA-Z0-9\.\/]+\.(php|html)$/;    return urlRegex.test(url);  }  console.log(isValidURL("http://site.ru/index.php"));  console.log(isValidURL("http://site.com"));  console.log(isValidURL("site.it/index.php")); |

**More precise**

|  |
| --- |
| function isValidURL(url) {    const urlRegex = /^(http|https):\/\/(([a-zA-Z0-9]+\.)?[a-zA-Z]+\/)\*[a-zA-Z0-9]+\.(php|html)$/;    return urlRegex.test(url);  }  console.log(isValidURL("http://site.ru/index.php"));  console.log(isValidURL("http://index.php"));  console.log(isValidURL("http://site.com"));  console.log(isValidURL("site.it/index.php")); |

Task 2

|  |
| --- |
| function isValidURL(url) {    const urlRegex = /^\+?(3|4)[0-9\s\(\)-]+$/; // instead of 0-9 can be \d    return urlRegex.test(url);  }  console.log(isValidURL("39991112233"));  console.log(isValidURL("3 (999) 1112233"));  console.log(isValidURL("+4 (999) 111-22-33"));  console.log(isValidURL("+4 (999) 111 22 33")); |

More specified

|  |
| --- |
| function isValidURL(url) {    const urlRegex = /^\+?(3|4)(\d{3}|(\s\(\d{3}\)\s))\d{3}(\s|\-)?\d{2}(\s|\-)?\d{2}$/;    return urlRegex.test(url);  }  console.log(isValidURL("39991112233"));  console.log(isValidURL("3 (999) 1112233"));  console.log(isValidURL("+4 (999) 111-22-33"));  console.log(isValidURL("+4 (999) 111 22 33")); |

Task 3

|  |
| --- |
| function getCredit(data, regex) {    const creditCardNumbers = data.match(regex);    return creditCardNumbers.toString();  }  function getCardSafe(data){    const regex = /\b\d{16}\b/g;    let card = getCredit(data, regex);    let safeCard = "\*\*\*\*\*\*\*\*\*\*\*\*" + card.slice(12, 16);    return data.replace(card, safeCard);  }  console.log(getCardSafe("My credit card is 1111222233334444. Please keep it safe.")); |

If the credit card is repeted more than once

|  |
| --- |
| function getCardSafe(data){    const regex = /\b\d{16}\b/g;    const parts = data.split(regex);    const cards = data.match(regex);    let safeCard = [];    for(let card in cards){      safeCard.push("\*\*\*\*\*\*\*\*\*\*\*\*" + cards[card].toString().slice(12, 16));    }      let newText = "";    for(let i=0; i<parts.length; i++){      newText += parts[i];      if(parts[i+1]){        newText += safeCard[i];      }    }    return newText;  }  console.log(getCardSafe("My credit card is 1111222233334444. Please keep it safe.")); |