Javascript

* Weakly typed language
  + No explicit type assignment
  + Data type can be switched dynamically
* Object oriented language
  + Data can be organized in local objects
  + Primitive and reference type
* Versatile language
  + Run in browser & directly on a PC/servers
  + Can form a board variety of task

1. **Core Javascript**a. variable: var <identifier> = value; **OR** let <identifier> = value;  
    **eg:** var name = ‘samar’; let age = 20; var isMale = true;  
     
   b. constants: const <identifier> = value;  
     
   c. functions: function <identifier> (parameters) {} **OR**  
    const <identifier> = function(parameters) {} **OR**  
    const <identifier> = (parameters) => {}  
     
    **eg:** function printName(name){console.log(name);}
2. **Reference vs Primitive Values**a. Primitives values: string, number, Boolean, undefined, null  
   Data that store in **stack** are primitive values  
     
   b. Reference values: objects (map, array)  
   Data that store in **heap** are primitive values
3. **Object**const <identifier> = {key: value}  
    **eg:** const person = {  
    name: “samar”,  
    age: 20,  
    greet: function(){  
    console.log(“hello, I am ”+ this.name);  
    }  
   }  
    **OR**  
   const person = {  
    name : “samar”,  
    age : 20,  
    greet(){  
    console.log(“hello, I am ”+ this.name);  
    }  
   }
4. **Arrays**const <identifier> = [“samar”,20, true, [], {}];
5. **Rest and Spread Operator (…)**  
   a. spread operator (…) = is used get a copy of an each element of object or array.  
     
    **eg:** const person = {  
    name: “samar”,  
    age: 20,  
    greet: function(){  
    console.log(“hello, I am ”+ this.name);  
    }  
   }  
     
   const copyPerson = {…person};  
     
   const hobbies = [“weight lifting”, “reading”, “cooking”];  
   const copyhobbies = […hobbies];  
   b. rest operator (…) = it is used to get arrays from a values.  
     
    **eg:** function toArray = (…args){  
    return args;   
    }  
    toArray(1,2,3,4,5,4,3,6,7,8,9);
6. **Destructuring**  
   We use destructuring to get data which is required from object.  
    **eg:** const person = {  
    name: “samar”,  
    age: 20,  
    greet: function(){  
    console.log(“hello, I am ”+ this.name);  
    }  
   }  
   const {name , age} = person; // key & variable name should be same   
     
   const hobbies = [“weight lifting”, “reading”, “cooking”];  
   const [hobby1, hobby2] = hobbies;  
      
   function displayPersonName({name})=> console.log(name);  
   displayPersonName(person);
7. **Async code & Promise & Callbacks**a. callbacks = in we pass a callback function to the async function  
     
    eg: const data = [1,2,3,4,5];  
   function displayData(){  
    setTimeout(()=>{  
    data.forEach(value => console.log(value));  
    }, 5000);  
   }   
   function addData(value, callback){  
    setTimeout(()=>{  
    data.push(value);  
    callback();  
    }, 5000);  
   }  
     
   addData(6, displayData);  
     
   b. Promise = Promise is like a Future in Dart  
     
    if you want to execute more then one promises; use   
    Promise.all([promise1, promise2,…..]).then(values => console.log(values)); // values is list of return value of each promise.   
     
    eg: const data = [1,2,3,4,5];  
     
   function displayData(){  
    setTimeout(()=>{  
    data.forEach(value => console.log(value));  
    }, 5000);  
   }   
   function addData(value){  
   return new Promise((resolve, reject)=>{  
    setTimeout(()=>{  
    data.push(value);  
    resolve();  
    }, 5000);  
     
    });  
   }  
     
   addData(6)  
   .then(getData)  
   .catch(error => console.log(error));  
     
   c. async await  
     
   eg: const data = [1,2,3,4,5];  
     
   function displayData(){  
    setTimeout(()=>{  
    data.forEach(value => console.log(value));  
    }, 5000);  
   }   
   function addData(value){  
   return new Promise((resolve, reject)=>{  
    setTimeout(()=>{  
    data.push(value);  
    resolve();  
    }, 5000);  
     
    });  
   }  
     
   async function init(){  
    await addData(6);  
    getData();  
   }  
     
   init();