Page:
Date: / /

1) Explain ES6 with history and features in details.

Ans ES6, also known as ECMAScript 2015, is a ~~significant~~ significant update to the Javascript language standard.

- it was finalized in June 2015 by the ECMA international standards organization.

- ES6 brought numerous enhancements and new features to Javascript, aimed at making the language more powerful, expressive, and easier to work with.

History!:

- The ECMAScript specification is the standardized specification of scripting language, which is developed by Brendan Eich of Netscape.

- Prior to ES6, the last major update to Javascript was ES5, which was released in 2009.
- ES6 was in development for several years before its finalization and brought many new feature and improvements to the language.

- The development of ES6 was driven by the need to address shortcomings in Javascript, make the language more expressive, and improve developer productivity.

Features!-

1) Default Parameters!- Default Parameters allow function Parameters to have default values if no argument or 'undefiend' is Provided
Ex!- Function fun(a, b=1){
        return a+b;
    }
    console.log(fun(5,2));
    console.log(fun(3)) // default value =1

2) Template literals !- Template literals allow for easy string interpolation and multiline strings using backticks (`...`).
Ex!- let name = 'Rahul';
      let message = `Hello, ${name}!`;

3) Arrow function!- Arrow function (' => ') Provide a concise syntax for writing anonymous function, enhancing code readability and reducing boilerplate.
Ex!-
      const add = (a,b) => a+b;

4) Let and Const !- The const keyword is used to declare constant variables whose values can't be changed.
  The let variables are mutable i.e. their values can be changed.

- ES6 introduced block-scoped variable using 'let' and 'const' keywords, which provided a more predictable variable scope behavior compared to 'var'.

5) Classes:- ES6 introduced classes in Javascript classes in Javascript can be used to create new objects with the help of a constructor, each class can only have one constructor inside it.

Ex:-
```
class Person {
    constructor (name) {
        this.name = name;
    }
  const P = new Person ('Rahul');
  console.log (P.name);
```

6) Rest and Spread operators:-

- The spread operator ('...') spreads the elements of an iterable (e.g., an array) into individual elements

- The rest operator ('...') collects multiple function arguments into a single array parameter.

7) Promises:- Promises provide a cleaner and more robust way to work with asynchronous code compared to callback functions.

2) define given concepts of advanced Javascript which are let, const, arrow function, ternary operator, destructuring, spread and modules.

Ans 1) let :- 'let' is keyword introduced in ES6 for declaring variables.

- Variables declared with 'let' are block-scoped, meaning they are only accessible within the block in which the are defined.
- The let variables are mutable i.e. their values can be changed. it works similar to the var keyword with some key differences like scoping which makes it a better option when compared to var

Ex:-

```
let x = 10;
if (true) {
    let y = 20;
    console.log(x);
}
console.log(y); // Reference error: y is not defi
```

2) const :- constants are block-scoped and cannot be reassigned after declaration.

- However, if a constant holds a reference to an object or array, the properties or elements of that object or array can still be mutated.

Ex:-

```
const PI = 3.14
PI = 3.14159; // TypeError
const name = "Rahul"
consol.log (name);       //output
```

3) arrow function:- Arrow functions provide a concise syntax for writing anonymous functions.
- They are denoted by the '=>' syntax.

- Arrow function are useful for writing shorter and cleaner code, especially for callback functions.
- you don't need the function keyword, the return keyword, and the curly brackets.

Ex:-
```
let age = 5;
let welcome = (age < 18) ?
    () => console.log ('Baby!') :
    () => console.log ('Adult');

welcome();
```

4) ternary operator:- The ternary operator, also known as the conditional operator, is a JavaScript operator that provides a compact way to write simple conditional statements.
- it ~~takes the~~ consists of a condition followed by a question mark ('?'), then an expression to execute if the condition is true. followed by a colon (':'), and finally an expression to execute if the condition is falsy.

Ex:-
```
const x = 10;

const message = x > 5 ? 'greater than 5' :
                 'less than or equal to 5';
```

Page:
Date: / /

5) Destructuring:- Destructuring allows you to extract values from arrays or objects and assign them to variables in a more concise way.

- Destructuring in Javascript basically means the breaking down of a complex structure into simpler parts.

Ex:-
```
const person = { name: 'Rahul', age = 21 };
cons { name , age } = person ;
console.log ( name , age);
```

6) Spread:- The spread operator ('...') allows an iterable (like an array or string) to be expanded into individual elements.

- it is commonly used for array manipulation, function arguments, and object literals.

Ex:-
```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const mergeArray = [...arr1, ...arr2];
```

7) modules:- modules are a way to organize and encapsulate Javascript code into reusable units of functionality.

- Javascript modules allow you to break up your code into separate files.

- modules can export functions, objects, or other values for use in other modules and can also import functionality from other modules.

Page:
Date: / /

Ex:- //module.js
```
export const add = (a,b) => a+b;

//main.js
import {add} from './module.js';
console.log(add(2,3));
```

5marks:-

1) differentiate types of variable in js with a truth table.

Ans

1) var:- ~~is~~ var is function-scoped. It means that variables declard with 'var' are accessible within the function they are declared in, regardless of block scope.

- variables declared with 'var' can be redeclared and reassigned within the same scope.

Syntax:- var variableName = value;

Ex:-
```
var x=10;
function examle () {
    var x =20;
    console.log(x); // output 20
}
example ();
console.log (x); // output: 10
```

2) let :- 'let' is block-scoped. It means that variables declared with 'let' are accessible only within the block they are declared in (e.g, within curly braces '{}').

- variables declared with 'let' can be reassigned but cannot be redeclared within the same scope.

syntax :- let variableName = value;

ex :-
```
let x = 10;
if (true) {
    let x = 20;
    console.log(x);   // output 20
}
console.log(x);   // output : 10
```

3) const :- const is also block-Scoped like 'let'

→ variables declared with 'const' cannot be reassigned once they are initialized. However, for objects and arrays, the properties or elements of the object or array can be mutated

- variables declared with 'const' cannot be redeclared within the same scope.

syntax :- const variableName = value;

ex :-
```
const PI = 3.14;
PI = 3.14159   // cannot be reassigned
const person = { name: 'Rahul' };
person.age = 20;   // properties of object
               can be modified   is valid
```

Truth Table :-
