

Module - 1

Page:

Date: / /

① Alert ("Program to show an alert");

②

1. True
2. False
3. ~~False~~ True
4. ~~False~~ True
5. False
6. False
7. False

③ False

④ Alert popper PopUp box with text: 2

⑤

Function myFunction(Age) {

 If (Age > 18) {
 return true;
 }

 else

 else {

 return confirm ("your age is less than 18");

 }

```

ask = (question, yes, no) => {
  If (confirm(question)) {
    yes();
  }
  else {
    no();
  }
}

```

```

ask("Plz confirm", () => alert("you agreed?"),
    () => alert("you canceled the execution"))

```

Module - 2

①

```

var user = new object();
user.name = "John";
user.surname = "Smith";
user.name = "Pete";
delete user.name

```

②

Let ~~John~~
Not copy

```

Fruits = ["Apple", "Peach", "Orange", "Banana"];

```

~~Fruits.length~~
Fruits.length = 4

③

```

Let john = {name: "john", age: 25};
Let pete = {name: "pete", age: 25};
Let mary = {name: "mary", age: 25};
Let users = [john, pete, mary];
Let names = users.map((item) => {
  return item.name;
})

```

④

```

Let john = {name: "john", surname: "smith", id: 1};
Let pete = {name: "pete", surname: "hunt", id: 2};
Let mary = {name: "mary", surname: "key", id: 3};
Let users = [john, pete, mary];
Let names = users.map((item) => {
  Let temp = {};
  temp.fullname = [item.name, item.surname].join(" ");
  temp.id = item.id;
  return temp;
})

```

⑤

```

Let salaries = {
  john: 100,
  pete: 300,
  mary: 250
}

```

```

function sumSalary(salaries) {
  var sum = 0;
  for (let key in salaries) {
    sum += salaries[key];
  }
  return sum;
}

```

6

(a) `let {name} = student;`

(b) `let {year's : age} = student;`

(c) ~~`let {isAdmin} = student;`~~

(d) `let {isAdmin = false} = student;`

(e) `let {isAdmin, ...user} = student;`

7

```
let user = {  
  name: "John Smith",  
  age: 35  
}
```

```
let userString = JSON.stringify(user);  
console.log(userString);
```

```
let userObj = JSON.parse(userString);  
console.log(userObj)
```


① JSON is an open standard file format & data interchange format that uses human-readable text to store & transmit data objects consisting of attribute value pairs & array.

② JavaScript promise are easy to manage when dealing with multiple asynchronous operations where callbacks can create callback hell leading to unmanageable code. Multiple callback functions would create callback hell that leads to unmanageable code.

③

```
let promise = new Promise(function  
  const x = "geeksforgeeks";  
  const y = "geeksforgeeks";  
  if (x === y) {  
    resolve();  
  } else {  
    reject();  
  }  
});
```

promise.

```
  then(function() {  
    console.log('success, You are a GEEK');  
  })  
  .catch(function() {  
    console.log('some error has occurred');  
  });
```

4. `fetch("https://fakestoreapi.com/products")`

`.then((res) => res.json())`

`.then((response) => {`

`resolve(response);`

`})`