Started on	Tuesday, 5 September 2023, 7:40 PM
State	Finished
Completed on	Tuesday, 5 September 2023, 7:44 PM
Time taken	4 mins 51 secs
Marks	14.00/15.00
Grade	**9.33** out of 10.00 (**93.33**%)

QUESTION 1

Correct

Mark 1.00 out of 1.00

What principle does the given code correspond to?

```
function Book(getTitle, getAuthor) {
   let title = getTitle;
   let author = getAuthor;
   this.giveTitle = function() {
      return title;
   }

   const summary = function() {
      return `${title} written by ${author}.`
   }

   this.giveSummary = function() {
      return summary()
   }
}
const book1 = new Book('JavaScript Ninja', 'John Resig');
book1.giveTitle();      // "JavaScript Ninja"
book1.summary();        // Uncaught TypeError: book1.summary is not a function
book1.giveSummary();    // "JavaScript Ninja written by John Resig."
```

Select one:

- a. abstraction ✔
- b. composition
- c. none of the listed
- d. polymorphism

QUESTION 2

Incorrect

Mark 0.00 out of 1.00

What is the type of relationship between the objects in these classes?

```
class Salary {
    constructor(pay, bonus) {
        this.pay = pay;
        this.bonus = bonus;
    }
    annual_salary() {
        return (this.pay * 12) + this.bonus;
    }
}
class Employee {
    constructor(name, age, salary) {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    total_salary() {
        if (this.salary) {
            return this.salary.annual_salary();
        }
    }
}
const salary = new Salary(15000, 10000);
const emp = new Employee('Max', 25, salary);
console.log(emp.total_salary()); // 190000
```

Select one:

- a. inheritance
- b. aggregation
- c. none of the listed
- d. association
- e. composition ✖

QUESTION 3

Correct

Mark 1.00 out of 1.00

What is the result of executing the following program?

```
class Adder {
    c = 30;
    constructor(a, b) {
        this.a = a;
        this.b = b;
    }
    getSum() {
        return this.a + this.b + c;
    }
};
const sum = new Adder(10,20);
const result = sum.getSum();
console.log(result);
```

Select one:

- a. 60
- b. null
- c. 30
- d. ReferenceError ✔
- e. undefined

QUESTION 4

Correct

Mark 1.00 out of 1.00

Which of the following examples of working with fields and methods of this class are incorrect?

```
class Employee {
    salary = 1200;
    static bonus = 300;
    constructor(position) {
        this.position = position;
    }
    getSalary() {
        return this.salary;
    }
    static getBonus() {
        return this.bonus;
    }
};
const employee = new Employee("developer");
```

Select one or more:

- [x] a. Employee.salary ✔
- [x] b. employee.getBonus() ✔
- [ ] c. Employee.bonus
- [x] d. Employee. position ✔
- [x] e. employee.bonus ✔
- [ ] f. employee.salary
- [ ] g. employee.position
- [ ] h. employee.getSalary()

QUESTION 5

Correct

Mark 1.00 out of 1.00

What is the type of relationship between the objects in these classes?

```
class Salary {
    constructor(pay, bonus) {
        this.pay = pay;
        this.bonus = bonus;
    }
    annual_salary() {
        return (this.pay * 12) + this.bonus;
    }
}

class Employee {
    constructor(name, age, pay, pay2) {
        this.name = name;
        this.age = age;
        this.salary = new Salary(pay, pay2);
    }
    total_salary() {
        return this.salary.annual_salary();
    }
}
const emp = new Employee('Max', 25, 15000, 10000);
console.log(emp.total_salary()); // 190000
```

Select one:

- a. none of the listed
- b. association
- c. inheritance
- d. composition ✔
- e. aggregation

QUESTION 6

Correct

Mark 1.00 out of 1.00

Which of the following methods calls a function with a given context this and an array of arguments?

Select one:

- a. setContext()
- b. create()
- c. apply() ✔
- d. call()
- e. bind()

QUESTION 7

Correct

Mark 1.00 out of 1.00

Which of the following methods creates a new function that at the time of the call has a specific assigned value of this, as well as a given sequence of arguments?

Select one or more:

- a. call()
- b. apply()
- c. bind() ✔
- d. setContext()
- e. create()

QUESTION 8

Correct

Mark 1.00 out of 1.00

Which keyword(s) is (are) required for ES6 Class definition?

Select one or more:

- a. super
- b. set
- c. class ✔
- d. constructor
- e. private
- f. static
- g. base

QUESTION 9

Correct

Mark 1.00 out of 1.00

Which of the above concepts are best practices in software development?

Select one or more:

- a. High Coupling
- b. Low Coupling ✔
- c. none of the listed
- d. Low Cohesion
- e. High Cohesion ✔

QUESTION 10

Correct

Mark 1.00 out of 1.00

Which of the following statements about polymorphism are correct?

Select one or more:

- a. polymorphism often uses inheritance ✔
- b. polymorphism does not promote code reuse
- c. all of the listed
- d. polymorphism is achieved through abstraction
- e. it provides an ability to call the same method on different JavaScript objects ✔

QUESTION 11

Correct

Mark 1.00 out of 1.00

How can you natively implement private data in JavaScript? (please, consider the latest proposals in this direction)

Select one:

- a. using _
- b. there is no such possibility yet
- c. using private keyword
- d. using # ✔
- e. using closures

QUESTION 12

Correct

Mark 1.00 out of 1.00

For the given source code, you need to implement the IT_specialist constructor function, which takes 3 parameters: fullName, position, salary and prototypically inherits from Employee. Indicate which of the prototypal inheritance implementations is correct.

```
function Employee(fullName, position) {
    this.fullName = fullName;
    this.position = position;
}

Employee.prototype.getPosition = function() {
    return this.position;
};

function IT_specialist() {
    // function-constructor implementation
}
const emp1 = new IT_specialist("John Johnson", "devops", 900);
console.log(emp1.fullName);        //  John Johnson
console.log(emp1.salary);          //  900
console.log(emp1.getPosition());   // devops
```

Select one:

a.
```
function IT_specialist(fullName, position, salary) {
  Employee.call(this, fullName, position);
  this.salary = salary;
}
IT_specialist.prototype =  Employee.prototype;
```

b. none of the listed

c. ✔
```
function IT_specialist(fullName, position, salary) {
  Employee.call(this, fullName, position);
  this.salary = salary;
}
IT_specialist.prototype = Object.create(Employee.prototype);
```

d.
```
function IT_specialist(fullName, position, salary) {
  Employee.call(this, fullName, position);
  this.salary = salary;
}
IT_specialist.prototype = Object.create(Employee);
```

e.
```
function IT_specialist(fullName, position, salary) {
this.fullName = fullName;
this.position = position;
  this.salary = salary;
}
IT_specialist.prototype =  Employee;
```

QUESTION 13

Correct

Mark 1.00 out of 1.00

The given Shape class has two private properties, _width and _height. Can we access them, modify them outside the Shape class?

```
class Shape {
    constructor(width, height) {
      this._width = width;      // private property
      this._height = height;    // private property
    }
    get area() {
      return this._width * this._height;
    }
  }

  const square = new Shape(10, 10);
  console.log(square.area);    // 100
```

Select one:

- a. no
- b. yes ✔

QUESTION 14

Correct

Mark 1.00 out of 1.00

Which of the following implementations of the IT_specialist child class constructor is correct? The constructor accepts 4 parameters: fullName, position, experience, salary.

```
class Employee {

    constructor(fullName, position) {
        this.fullName = fullName;
        this.position = position;
    }
    getPosition() {
        return this.position;
    }
}

class IT_specialist extends Employee {

    // constructor implementation
}

const employee = new IT_specialist("Peter Peterson", "developer", 12, 2222);
```

Select one:

a.
```
constructor( fullName, position, experience, salary ) {
    this.experience = experience;
    this.salary = salary;
    super(fullName, position);
}
```

b.
```
constructor( fullName, position, experience, salary ) {
    super(fullName, position);
    this.experience = experience;
    this.salary = salary;
}
```
✔

c.
```
constructor( fullName, position, experience, salary ) {
    this.fullName = fullName;
    this.position = position;
    this.experience = experience;
    this.salary = salary;
}
```

d.
```
constructor( fullName, position, experience, salary ) {
    super(fullName, position, experience, salary);
    this.experience = experience;
    this.salary = salary;
}
```

e. none of the listed

QUESTION 15

Correct

Mark 1.00 out of 1.00

Indicate which of the following is not natively supported in JavaScript?

Select one or more:

- a. modularity
- b. overloading ✔
- c. polymorphism
- d. overriding
- e. generics ✔
- f. interfaces ✔

◄ Practical tasks. OOD / OOP

Jump to...

Useful links ►