

20214987
anh.pn214987@sis.hust.edu.vn

1

2

3

4

5

6

7

8

9

10

11

12

13

Câu hỏi

Question #fbc7ed

1 point possible (ungraded, results hidden)
Các phát biểu nào đúng về khái niệm *prototype* trong JavaScript?

- ☐ `prototype` là một thuộc tính của `Object`
- ☒ mỗi JavaScript object đều có một *prototype* cài đặt các phương thức của mình
- ☐ các JavaScript object sử dụng thuộc tính `prototype` để tham chiếu (reference) đến class của mình
- ☐ thuộc tính `__proto__` của mỗi object tham chiếu đến native code của object

Submit

Answer submitted.

Question #54ae16

1 point possible (ungraded, results hidden)
Cho biết các class và object được khai báo như sau

```
class A { }  
class B { }  
a = new A()  
b = new B()
```

Phương pháp nào để biến `class B` trở thành kế thừa `class A` ?

- ☒ `B.prototype.__proto__ = A.prototype`
- ☐ `b.prototype = a`
- ☐ `b.__proto__ = a`
- ☒ `b.__proto__.__proto__ = a.__proto__`

Submit

Answer submitted.


Question #417636

1 point possible (ungraded, results hidden)

Đối với mọi hàm `f`, lời gọi `f.call(f)` và `f.apply(f)` đều thu được kết quả như nhau. Nhận định này đúng hay sai?

- ☒ Đúng
- ☐ Sai

Submit

 Answer submitted.

Question #ee188c

1 point possible (ungraded, results hidden)


Cho đoạn code như bên dưới

```
function f() { console.log(this.x); }  
var x = 10;  
f.x = 10;
```

Lời gọi hàm `f.call(null)` và `f.call(f)` có những điểm giống nhau và khác nhau nào sau đây?

- ☐ Luôn có kết quả như nhau
- ☒ Sử dụng biến số `x` thuộc các scope khác nhau
- ☒ Đều thực thi phương thức `call()` của object `f`
- ☐ Tất cả đều sai

Submit

 Answer submitted.


Question #600392

1 point possible (ungraded, results hidden)

Trong JavaScript, callback là gì?

- ☐ Một loại biến được truyền vào hàm
- ☐ Một hàm được trả về từ một hàm khác
- ☒ Một hàm được truyền làm tham số vào một hàm khác để cùng được thực thi trong khi thực thi hàm khác đó
- ☐ Một kiểu dữ liệu để lưu trữ hàm

Submit

 Answer submitted.

Question #0beaf3

1 point possible (ungraded, results hidden)

Cần xây dựng hàm *asynchronous* `asyncFun()` thực hiện giả lập các hoạt động asynchronous và trả về thông báo "Done" khi hoàn thành. Những phương pháp khai báo nào sau đây là phù hợp với yêu cầu trên?



```
function asyncFunc() {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Done"), 1000);
  });
}
```



```
async function asyncFunc() {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Done"), 1000);
  });
}
```



```
async function asyncFunc() {
  return ("Done");
}
```




```
function asyncFunc() {
  return ("Done");
}
```



Tất cả đều sai

Submit

 Answer submitted.

Question #cda29c

1 point possible (ungraded, results hidden)

Cho một khai báo hàm như bên dưới, trong đó `xxx` là một keyword JavaScript đang thiếu:

```
function* generateNumbers() {
  xxx 1;
  xxx 2;
  xxx 3;
}

var gen = generateNumbers();
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
```


Cho biết kết quả trên console của đoạn mã bên trên là:

```
1
2
3
undefined
```

Hãy cho biết `xxx` là gì?

yield

Submit

 Answer submitted.


Question #25afaa

1 point possible (ungraded, results hidden)

Những *native function* nào sau đây của JavaScript cho phép thực thi mã lệnh kiểu không đồng bộ

- ☐ `timeout`
- ☒ `setTimeout`
- ☐ `startInterval`
- ☐ `delay`
- ☒ `setInterval`
- ☐ `interal`

Submit

 Answer submitted.


Question #48cb49

1 point possible (ungraded, results hidden)

Trong JavaScript, làm thế nào để triển khai một function với số lượng tham số không xác định?

- ☐ Sử dụng từ khóa `varargs`
- ☒ Sử dụng toán tử `(...)`
- ☒ Sử dụng object `arguments` trong nội dung function
- ☐ Không có cách nào để làm điều này

Submit

 Answer submitted.

Question #80f488

1 point possible (ungraded, results hidden)

Cho các hàm async JavaScript như bên dưới.

```
function asyncFunc() {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Done"), 1000);
  });
}

async function test() {
  const result = xxx asyncFunc();
  console.log(result);
}

test();
```

Kết quả thực thi đoạn code bên trên là "Done". Hãy cho biết từ khóa đang thiếu `xxx` trong nội dung hàm `test()` là gì?

await

Submit

 Answer submitted.

Question #9843bd

1 point possible (ungraded, results hidden)

Giả sử `a` và `b` là 2 object JavaScript:

```
let a = { }
let b = { }
```

Phương án nào để biến `object b` trở thành kế thừa từ `object a`?

- ☐ `b.prototype = a`
- ☒ `b.__proto__ = a`
- ☐ cả 2 phương án trên
- ☐ không phương án nào đúng

Submit

Answer submitted.

Question #88a327

1 point possible (ungraded, results hidden)

Sử dụng built-in object `arguments` của function để khai báo hàm `sum()` tính tổng các số đưa vào theo các tham số khi gọi hàm:

```
function sum() {
  let n=0;
  for (i in arguments) n+=i;
  return n;
}
```

Khai báo bên trên đang cho kết quả không đúng mong muốn:

```
sum(1,2,3) // trả về kết quả 0012
sum(1,2,3,4) // trả về kết quả 00123
sum(1,2,3,4,5) // trả về kết quả 001234
```

Các phương pháp sau đây, cái nào có thể sửa lỗi chương trình?

- ☐ Sử dụng phương thức `arguments.reduce()` để tính tổng
- ☒ Thay đổi cú pháp `for..` để lấy tổng theo value thay vì key
- ☐ Không thể sửa vì khai báo function như trên không xác định được các tham số khi gọi hàm
- ☒ Đổi từ khóa `in` thành `of`

Submit

Answer submitted.

Question #56fd64

1 point possible (ungraded, results hidden)

Cho một khai báo hàm như bên dưới, trong đó `xxx` là một đoạn code JavaScript đang thiếu trong phần khai báo các tham số của hàm:

```
function sumOf(xxx) {
  let total = 0;
  for (a of arg) {
    if (p=='odd' && a%2==1) total += a;
    if (p=='even' && a%2==0) total += a;
  }
  return total;
}
```

Biết rằng các lời gọi hàm sau đây có các kết quả tương ứng là:

```
sumOf('odd') // = 0
sumOf('even') // = 0
sumOf('odd',1,2,3,4,5) // = 9
sumOf('even',1,2,3,4,5,6,7,8) // = 20
```

Hãy cho biết xxx là gì (*chú ý không nhập vào các ký tự trắng*)?

p,...arg