Как пройти собеседование по js?
И зачем?

Юра Федоренко

Кто здесь?

А какие варианты?

А какие варианты?

- Тестовое задание
- Рекомендация
- Собеседование

Кого ищем?

Что ищем?

Корпоративная культура

Корпоративная культура

Корпоративная культура

Никто не обязан вас учить

Что делать на собеседовании?

Что делать на собеседовании?

- Не врать

Что делать на собеседовании?

- Не врать
- Не воровать

Что делать на собеседовании?

- Не врать
- Не воровать
- Не быть мудаком

Что делать тем, кто собеседует?

Что делать тем, кто собеседует?

- Не врать
- Не воровать
- Не быть мудаком

Быть нормальным

Задачи

Типы данных

```
var arr = [1,2];
var brr = arr;
brr = [42, 43];
arr[0]; ???
```

```
var arr = [1,2];
var brr = arr;
brr[0] = 42;
arr; ???
```

Что вернет функция?

```
(function(){
    return [2,2,2,2].map(parseInt);
})()
```

Что вернет метод?

```
[2,2,2,2].map(parseInt);
```

Конечно же...

```
[2,2,2,2].map(parseInt);
```

Конечно же...

```
[2,2,2,2].map(parseInt);
```

```
[2,NaN,NaN,2]
```

O_o ??!!1

```
[2,2,2,2].map(parseInt);
```

```
[2,NaN,NaN,2]
```

А почему так?

```
[2,2,2,2].map(() => {
    parseInt()
});
```

А почему так?

```
[2,2,2,2].map((item, index) => {
    parseInt(item, index);
});
```

this

call, apply это понятно

A bind сможешь?

```
function fn(a, b) {
  console.log(a, b, this)
}

var magicFn = bind(fn, {});

magicFn(2, 3);
```

Нужно думать как bind!

```
function bind(cb, context) {


}
```

```
function bind(cb, context) {
  return function() {

  }
}
```

```
function bind(cb, context) {
  return function() {
    cb.apply(context)
  }
}
```

```
function bind(cb, context) {
  return function() {
    cb.apply(context, arguments)
  }
}
```

```
function bind(cb, context) {
  return function() {
    return cb.apply(context, arguments)
  }
}
```

```
function fn(a, b) {
  console.log(a, b, this)
}

var magicFn = bind(fn, {});

magicFn(2, 3);
```

Классика

```
for (var i = 0; i < 10; i++) {
  setTimeout(function(){
    console.log(i);
  }, i*1000);
}
```

Легкое решение

```
for (let i = 0; i < 10; i++) {
  setTimeout(function(){
    console.log(i);
  }, i*1000);
}
```

Просто решение

```
for (var i = 0; i < 10; i++) {
  setTimeout((function(i) {
    return function (){
      console.log(i);
    }
  })(i), i*1000);
}
```

Еще одно, просто решение

```
for (var i = 0; i < 10; i++) {
  (function(i) {
    setTimeout(function (){
      console.log(i);
    }, i*1000);
  })(i)
}
```

Умное решение

```
for (var i = 0; i < 10; i++) {
  setTimeout((function (i){
    console.log(i);
  }).bind(null, i), i*1000);
}
```

А как же new?

Что произойдет?

```
function() {
  this.name = 'yura';
}
```

У меня для вас три истории...

Метод

```
var o = {
   fn: function() {
      this.name = 'yura';
   }
}


o.fn();
```

Просто функция

```
function fn() {
  this.name = 'yura';
}

fn();
```

Конструктор

```
function fn() {
  this.name = 'yura';
}

new fn();
```

Конструктор

```
function fn() {
  this.name = 'yura';
}

new fn;
```

У меня для вас три истории...

Что происходит когда вызываем с new

- Создается новый объект, он становится this’ом
- Неявно возвращается
- Устанавливается прототип

Что происходит когда вызываем с new

- Создается новый объект, он становится this’ом
- Неявно возвращается
- Устанавливается прототип O_o

__proto__ *vs* prototype

Не одно и тоже!

Прототипное наследование

Конструктор

```
function fn() {
  this.name = 'yura';
}

new fn;
```

```
var obj = {
    a: 5,
    b: {
        c: 10
    }
};

obj.__proto__ = {
    a: 10,
    b: {
        c: 20
    }
};
```

```
var obj = {
    a: 5,
    b: {
        c: 10
    }
};

obj.__proto__ = {
    a: 10,
    b: {
        c: 20
    }
};
```

```
delete obj.a;
console.log(obj.a);

delete obj.a;
console.log(obj.a);

delete obj.b;
console.log(obj.b.c);

delete obj.b.c;
console.log(obj.b.c);
```

```
var obj = {
    a: 5,
    b: {
        c: 10
    }
};

obj.__proto__ = {
    a: 10,
    b: {
        c: 20
    }
};
```

```
delete obj.b;
console.log(obj.b.c);

var b = obj.b;

delete b.c;
console.log(obj.b.c);
```

t.me/callforward

?

t.me/djamah