ООП и ФП в Javascript

WHO IS THE BEST OF THE BEST???

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы программирования

- Процедурная
- Объекто-ориентированная
- Функциональная

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы программирования

- Процедурная
- Объекто-ориентированная
- Функциональная

Характерные языки

- BASIC
- C
- Fortran
- Pascal

ООП и ФП в Javascript

Парадигмы программирования

- Процедурная
- Объекто-ориентированная
- Функциональная

Характерные языки

- C++
- Java
- Ruby

ООП и ФП в Javascript

Парадигмы программирования

- Процедурная
- Объекто-ориентированная
- Функциональная

Характерные языки

- LISP
- F#
- Scala
- Haskell
- ML

ООП и ФП в Javascript

Императивный подход (как)

Процедурное программирование

Объекто-ориентированное программирование

Декларативный подход (что)

Функциональное программирование

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Императивный подход (как)

Процедурное программирование

Объекто-ориентированное программирование

JS

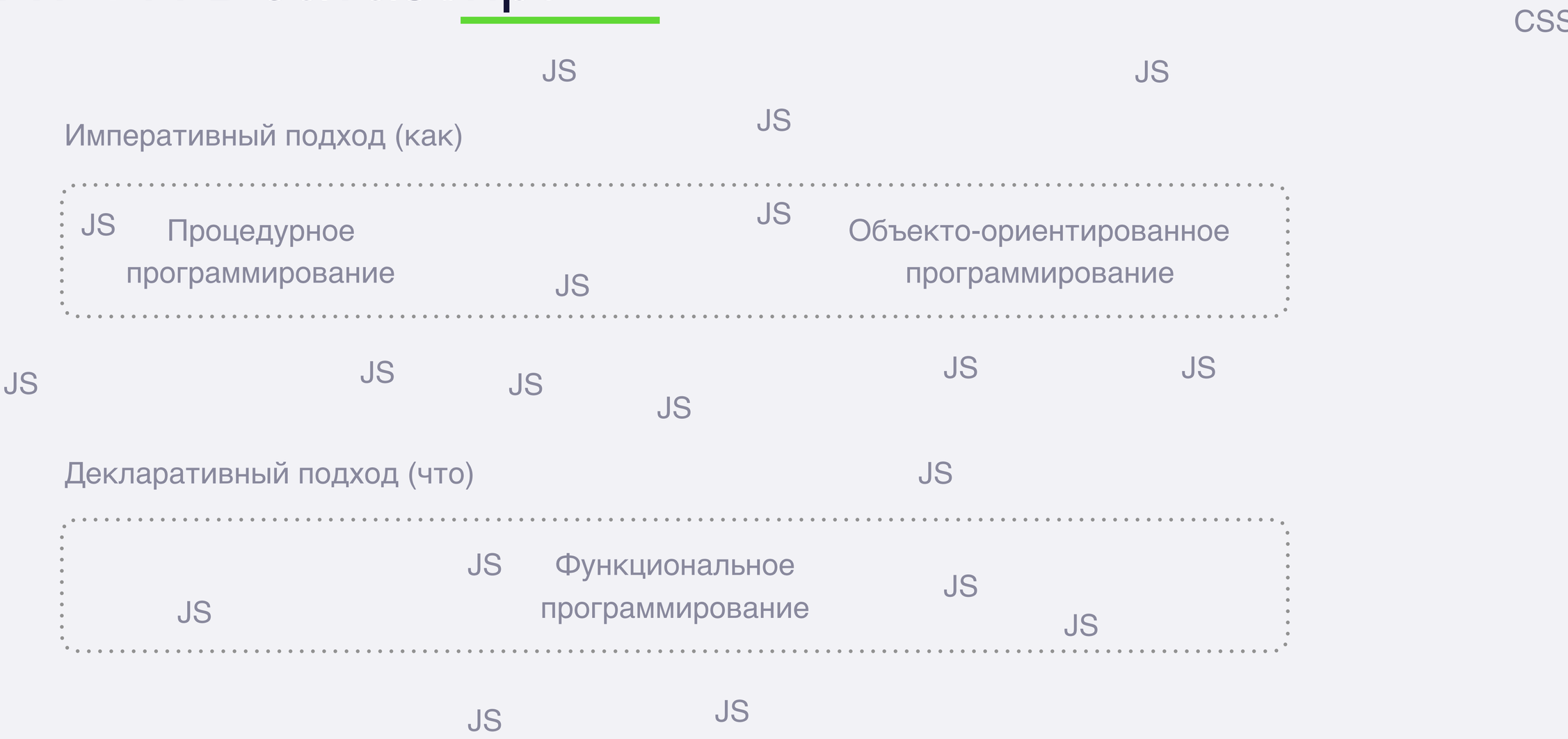
Декларативный подход (что)

Функциональное программирование

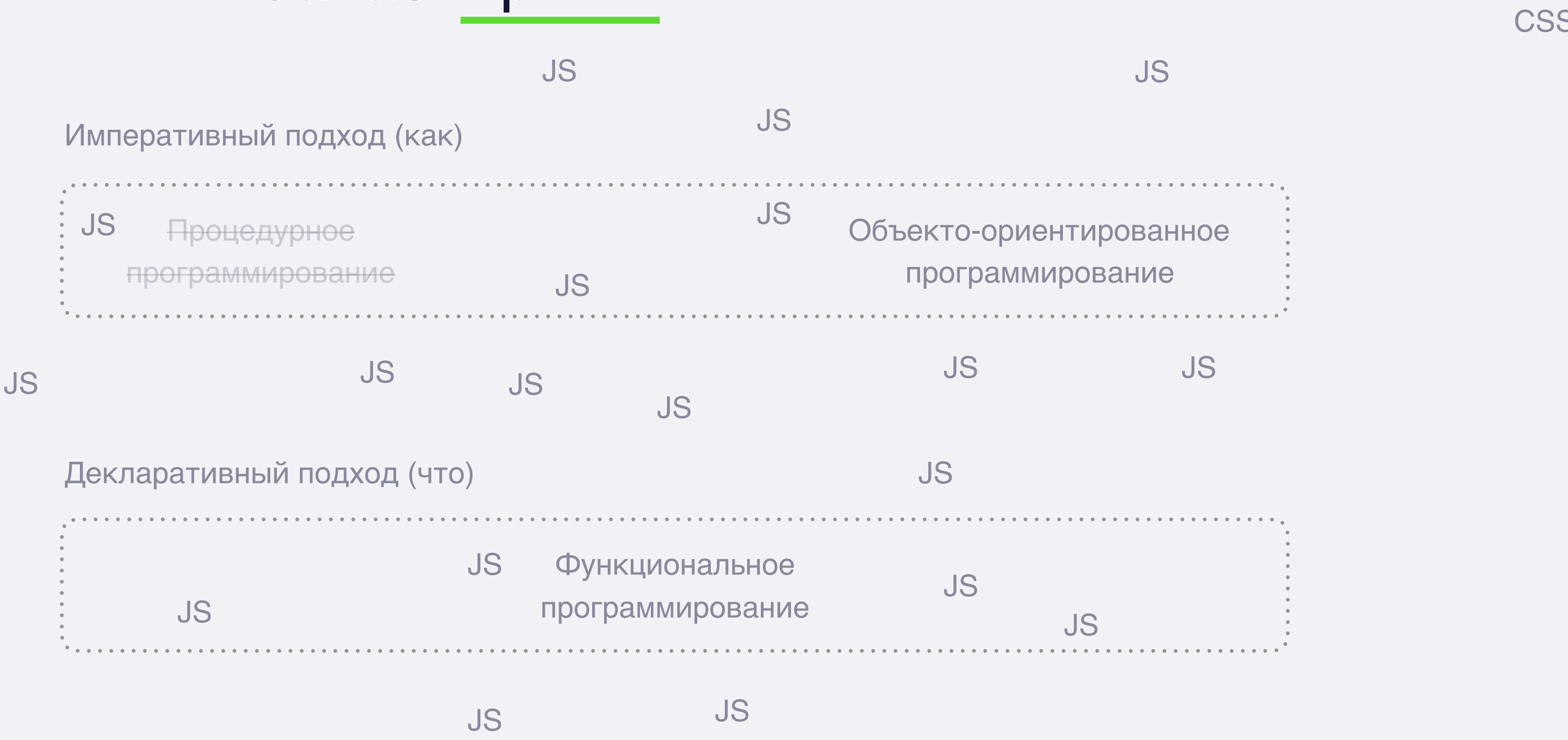
Сбербанк Онлайн

Devteam

ООП и ФП в Javascript



ООП и ФП в Javascript



Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы ООП

- Инкапсуляция
- Наследование
- Полиморфизм

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы ООП

- Инкапсуляция - данные и методы сокрыты в классе и его экземпляре
- Наследование
- Полиморфизм

Спойлер: onClick = (event) => { this.setState({ value: event.target.value }) }

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы ООП

- Инкапсуляция
- Наследование - новый класс на основе уже существующего
- Полиморфизм

Спойлер: class MyPerfectTodoList extends React.Component

ООП и ФП в Javascript

Парадигмы ООП

- Инкапсуляция
- Наследование
- Полиморфизм - способность функции или системы оставаться идентичной при изменяемых условиях эксплуатации

Спойлер: [].toString(), ({}).toString(), (2).toString(), false.toString()

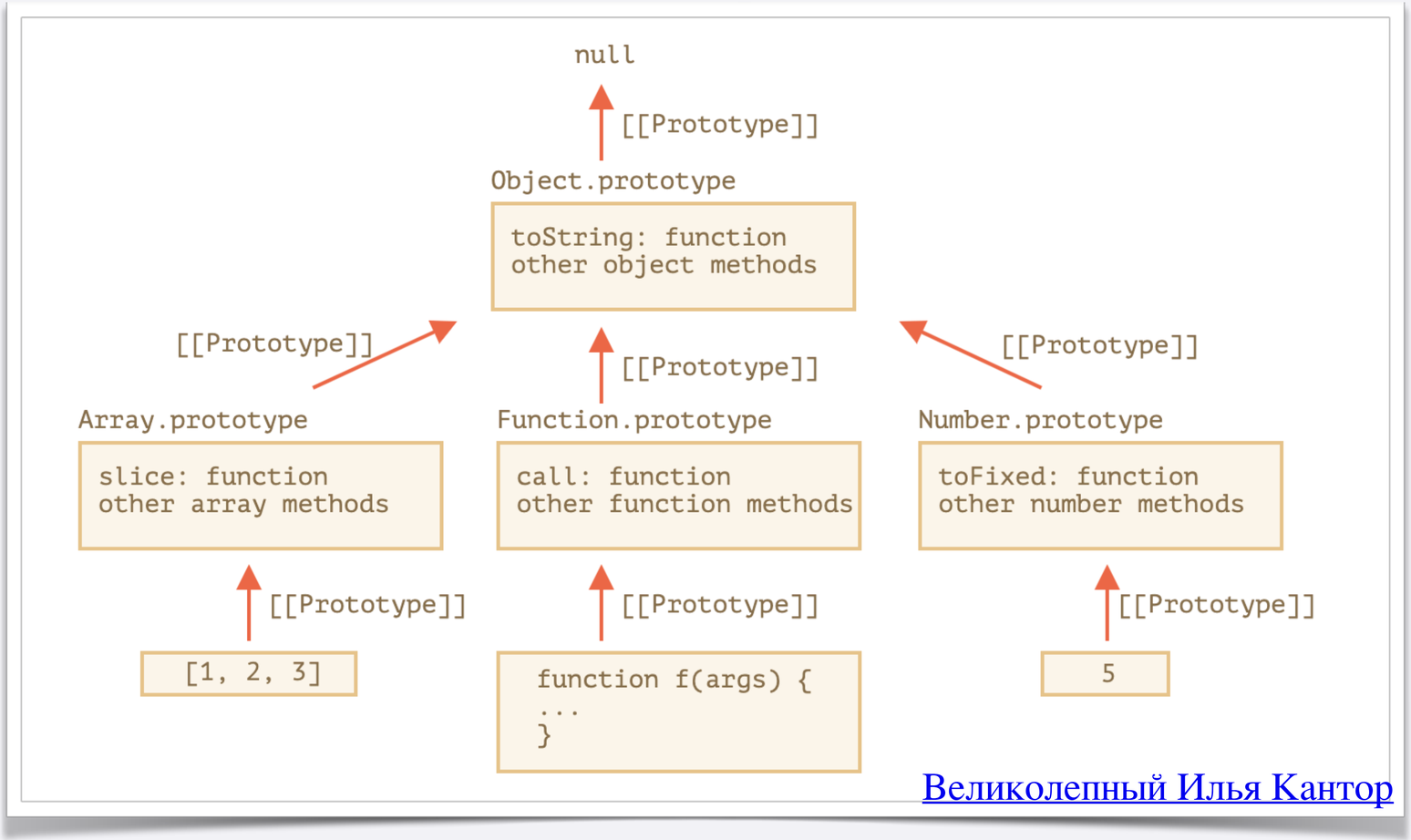
ООП и ФП в Javascript

ООП в ключевых словах JS

- prototype
- __proto__
- class
- extends
- super
- instanceof
- constructor
- new
- this
- static
- ~~private, public, protected, @overrides, implements, final~~

ООП и ФП в Javascript

Прототипы



Великолепный Илья Кантор

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Прототипы - важное

- Object.getPrototypeOf(f) === f.__proto__
- f.__proto__ === F.prototype
- this всегда объект перед точкой
- F.prototype.constructor === F
- {} ~ new Object()
- Если у меня нет своего свойства, я беру его в родителе

ООП и ФП в Javascript

Прототипы - важное

- Object.getPrototypeOf(f) === f.__proto__
- f.__proto__ === F.prototype
- this всегда объект перед точкой
- F.prototype.constructor === F
- {} ~ new Object()
- Если у меня нет своего свойства, я беру его в родителе

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Контекст

```
const obj = {
  showMe () {
    console.log(this)
  }
}

obj.showMe()
```

```
function Color (red, green, blue, opacity = 1) {
  this.r = red
  this.g = green
  this.b = blue
  this.opacity = opacity
}

new Color(0, 0, 0) === { r: 0, g: 0, b: 0, opacity: 1 }
Color(0, 0, 0) === undefined
```

ООП и ФП в Javascript

Прототипы - практика

- Создайте объект counter со свойством count и методом add, увеличивающим this.count на единицу
- Создайте объект clock с методом ticktock, вызывающим this.add с интервалом в 1000мс
- Свяжите clock и counter так, чтобы clock.ticktock() вел отсчет в clock
- Добейтесь того, чтобы clock и counter начинали отсчет со значения 3600
- Добейтесь того, чтобы clock менял значение и в counter

ООП и ФП в Javascript

Еще немного про функциональное наследование

```
function Somebody (surname) {
  this.surname = surname
}

function Person (name, surname) {
  Somebody.call(this, surname)
  this.name = name
}

const smbd = new Somebody('Bond')
const prsn = new Person('James', 'Bond')
```

```
function Somebody (surname) {
  this.surname = surname
}

function Person (name, surname) {
  Object.assign(this, new Somebody(surname))
  this.name = name
}

const smbd = new Somebody('Bond')
const prsn = new Person('James', 'Bond')
```

ООП и ФП в Javascript

От боли ссылок к адекватному ~~сахарку~~ синтаксису

```
function Color (red, green, blue, opacity = 1) {
  this.r = red
  this.g = green
  this.b = blue
  this.opacity = opacity
}
```

```
class Color {
  constructor (red, green, blue, opacity = 1) {
    this.r = red
    this.g = green
    this.b = blue
    this.opacity = opacity
  }
}
```

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

От боли ссылок к адекватному ~~сахарку~~ синтаксису

```
function Color (red, green, blue, opacity = 1) {
  this.r = red
  this.g = green
  this.b = blue
  this.opacity = opacity
}
```

```
class Color {
  constructor (red, green, blue, opacity = 1) {
    this.r = red
    this.g = green
    this.b = blue
    this.opacity = opacity
  }

  reverse () {
    this.r = 255 - this.r
    this.g = 255 - this.g
    this.b = 255 - this.b
  }
}
```

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Наследование классов

```
class SoStrangeColor extends Color {
  constructor (…args) {
    super(…args)
  }

  reverse () {
    super.reverse()
    this.opacity = 1 - this.opacity
  }
}
```

ООП и ФП в Javascript

Наследование классов

```
class SoStrangeColor extends Color {
  constructor (...args) {
    super(...args)
  }

  reverse () {
    super.reverse()
    this.opacity = 1 - this.opacity
  }
}
```

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Статические

```
class A {
  static count = 0
  constructor () {
    A.count += 1
  }
}

new A()
new A()
new A()
console.log(A.count) // 3
```

и приватные свойства

```
class A {
  #foo
  constructor (foo, bar) {
    this.#foo = foo
    this.bar = bar
  }
}

const a = new A(1, 2)
console.log(a.#foo) // throw error
console.log(a.bar) // 2
console.log(Object.keys(a)) // [‘bar’]
```

ООП и ФП в Javascript

Классы - практика

- Создайте объект counter со свойством count и методом add, увеличивающим this.count на единицу
- Создайте объект clock со методом ticktock, запускающим часы так, чтобы увеличивался this.count на единицу с интервалом в 1000мс
- Свяжите clock и counter так, чтобы clock.ticktock() вел отсчет в clock
- Добейтесь того, чтобы clock и counter начинали отчет со значения 3600
- Добейтесь того, чтобы clock менял значение и в counter

Реализуйте то же самое с классами и их экземплярами

ООП и ФП в Javascript

Зачем мне ООП в Javascript?

- Все данные разбиты на сущности
- Хранение логики и состояния приложения в структурированном виде
- Возможна композиция из различных классов (привет, React)
- Безопасная работа с типами данных
- Все данные в JS - объекты, связанные прототипами, понимание их работы спасает от велосипедов

ООП и ФП в Javascript

Домашнее задание - ООП

- Реализация хранилища состояния веб-сайта с использованием наследования для хранения различных схожих типов данных

ООП и ФП в Javascript

Парадигмы ФП

- Функции высшего порядка
- Чистые функции
- Рекурсия

ООП и ФП в Javascript

Парадигмы ФП

- Функции высшего порядка - принимает и возвращает функцию
- Чистые функции
- Рекурсия

Спойлер: const memoFn = _.memoize(fn); memoFn(1); memoFn(1)

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы ФП

- Функции высшего порядка
- Чистые функции - не содержит side-effects
- Рекурсия

Спойлер: ~~function (a) { window.a = a };~~ function (a, updateProp) { return { …a, prop: updateProp } }

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Парадигмы ФП

- Функции высшего порядка
- Чистые функции
- Рекурсия - вместо циклов и присваиваний постоянное итерирование по функции

Спойлер: function factorial (i) { if (i === 1) return 1; return i * factorial(i - 1) }

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Каррирование

```
function pow (a, b) {
  return a * b
}
```

...

```
pow(2, 3) === curriedPow(2)(3)
```

ООП и ФП в Javascript

Каррирование

```
function pow (a, b) {
  return a * b
}

function curriedPow (a) {
  return function (b) {
    return a * b
  }
}

pow(2, 3) === curriedPow(2)(3)
```

ООП и ФП в Javascript

Каррирование

```
const twice = curriedPow(2)

twice(3) // 6
twice(4) // 8
twice(0) // 2
```

ООП и ФП в Javascript

Каррирование

Каррирование можно использовать, чтобы собрать необходимую функцию и использовать ее вместо данных - составлять программу из инструментов

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Каррирование - практика

- Напишите функцию-декоратор, которая автоматически каррировала бы любую функцию.
  Например: tripleSum принимает всегда 3 аргумента, тогда результат const acTripleSum = autoC(tripleSum) позволит получать одинаковый результат при вызовах:
  acTripleSum(1, 2, 3) === acTripleSum(1)(2)(3) === acTripleSum(1, 2)(3) === acTripleSum(1)(2, 3)
- Найдите применение такой функции и ее результирующим функциям

Подсказка: используйте псевдомассив arguments или spread …args

ООП и ФП в Javascript

Композиция

sum(group(invert(myData)))

ООП и ФП в Javascript

Композиция

```
const prepare = compose(sum, group, invert)

prepare(myData)
```

ООП и ФП в Javascript

Композиция и каррирование

```
const prepare = compose(sum, groupBy(getName), invert)

prepare(myData)
```

ООП и ФП в Javascript

Библиотеки для работы в ФП стиле

- lodash (lodash/fp) - простой в изучении, позволяющий перейти от процедурной работы к функциональной
- ramda - эталон ФП в js, но еще не fantasy land
- ramda-fantasy - мы потеряли вас, если вы попали туда

ООП и ФП в Javascript

lodash

- _.curry
- _.memoize
- _.identity
- _.partial
- _.chain
- _.flow

ООП и ФП в Javascript

Композиция - практика

- Напишите функцию compose
- Разбейте преобразование массива данных на несколько простых действий с каррированием так, чтобы их было выгодно переиспользовать
- Используйте lodash

ООП и ФП в Javascript

Монады, моноиды, функторы

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Монады, моноиды, функторы

Пожалуйста, берегите коллег!

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Монады, моноиды, функторы

~~Пожалуйста, берегите коллег~~! Читайте и проникайтесь, здесь нужна философия.

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Зачем мне ФП в Javascript?

- Гибкий инструментарий по работы с данными
- Парсеры
- Геттеры/сеттеры
- Селекторы данных
- Решение задачи на высочайшем уровне абстрагирования от данных, это надо уметь!
- ~~Возможность выпендриться среди сообщества программистов~~

ООП и ФП в Javascript

Домашнее задание - ФП

- Реализовать парсер данных от backend
- Сконструировать селекторы данных из объекта с использованием мемоизации

ООП и ФП в Javascript

ООП vs ФП

- ООП друг типизированных языков
- Местами избыточный код, несмотря на переиспользование методов
- ООП довольно прост, но иногда непредсказуем

- У ФП высокий порог вхождения
- С ФП упрощается отладка кода, тестирование сводится к простому in-out
- ФП максимально абстрагируется от обычных операций кода: присваивания, циклов

Они решают разные задачи!

ООП и ФП в Javascript

Chaining

```
[{ id: '1' }, {}, { id: '2' }, { id: '3' }]
  .filter(({ id }) => id)
  .map(({ id }) => id)
  .reduce((memo, id) => memo + id, '') // 123

[{ id: '1' }, {}, { id: '2' }, { id: '3' }]
  .filter(({ id }) => id)
  .map(({ id }) => id)
  .join(',') // 1,2,3
```

ООП и ФП в Javascript

Chaining

```
const  chainObj = {
  value: [],
  add (value) {
    this.value.push(value)
    return this
  },
  getValue () {
    return this.value
  }
}

chainObj.add(1).add(2).add(3).getValue().add(4) // [1, 2, 3]
```

Сбербанк Онлайн

Devteam

ООП и ФП в Javascript

Полезные ссылки

- http://learn.javascript.ru/prototypes - прототипы
- http://learn.javascript.ru/classes - ES6 классы
- https://habr.com/ru/company/mailru/blog/327522/ - много букв по пути до Ramda
- https://github.com/fantasyland/fantasy-land - спецификации здорового ФП