**Предикат** отвечает на утвердительный вопрос «да» или «нет», возвращая значение типа bool.

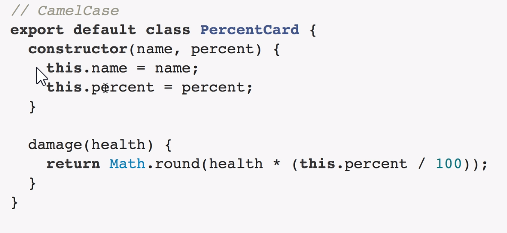
**Абстракция** является основой ООП и позволяет работать с объектами не вдаваясь в особенности их реализации

**Markdown** — язык разметки текстов. Такие тексты легко писать и читать. Их можно без труда сконвертировать в HTML. Большинство программистов предпочитают Markdown для написания документации, описаний своих проектов, написания блогов и так далее.

**Императивный подход** – возвращается измененная структура данных

**Функциональный подход** – возвращается новая структура, являющаяся преобразованием старой

***Класс***



**Полиморфизм –** способность функции обрабатывать данные разных типов

Функции высшего порядка

map

filter

reduce

**Массив**

arr.push(new element) – добавление эл

* добавление элементов в начало или конец массива, удаление элементов из начала или конца массива (shift, unshift, pop, push). Эта группа методов позволяет работать с массивом как с [очередью](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) (метод доступа FIFO) или [стеком](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) (метод доступа LIFO). Давайте подробнее рассмотрим это на примере:

*// инициализируем демонстрационный массив*

**const** planets = ['Mercury', 'Venus', 'Earth', 'Mars'];

*// Очередь является упорядоченным набором данных, организованным по принципу FIFO ('first in - first out'), т.е. добавление элементов всегда происходит в конец очереди, а извлечение (удаление) элементов - из её начала:*

*// push добавляет один или несколько элементов в конец массива*

*// и возвращает длину изменённого (мутированного) массива*

*// (если точнее, то метод возвращает обновлённое свойство length массива,*

*// являющееся значением самого последнего индекса, увеличенным на единицу)*

planets.push('Jupiter'); *// возвращаемое значение: 5*

planets.push('Saturn', 'Uranus'); *// возвращаемое значение: 7*

console.log(planets); *// [ 'Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus' ]*

*// shift удаляет первый элемент массива и возвращает его значение*

planets.shift(); *// удалили элемент из начала массива, возвращаемое значение: 'Mercury'*

planets.shift(); *// ещё раз удалили элемент из начала массива, возвращаемое значение: 'Venus'*

console.log(planets); *// [ 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus' ]*

*// Стек является упорядоченным набором данных, организованным по принципу LIFO ('last in - first out'), т.е. добавление и удаление элементов всегда происходит из конца такой коллекции:*

*// добавим элемент в конец массива уже известным нам способом*

planets.push('Neptune'); *// возвращаемое значение: 6*

console.log(planets); *// [ 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune' ]*

*// pop удаляет последний элемент массива и возвращает его значение*

planets.pop(); *// возвращаемое значение: 'Neptune'*

planets.pop(); *// возвращаемое значение: 'Uranus'*

console.log(planets); *// [ 'Earth', 'Mars', 'Jupiter', 'Saturn' ]*

* рассмотренная в курсах "золотая троица" функций высшего порядка (map, filter, reduce, к которой ещё можно добавить reduceRight). После того, как мы на практике Хекслет реализовали их вручную самостоятельно, никаких затруднений в пользовании встроенными в язык методами (их аналогами "из коробки") быть не должно.

*// reduceRight - зеркальное отражение reduce, осуществляет свёртку, обрабатывая элементы справа налево*

**const** planets = ['Меркурий', 'Венера', 'Земля', 'Марс'];

**const** closer = planets.reduceRight((acc, planet) => `${acc} ${planet}`, 'всё ближе к Солнцу:'); *// 'всё ближе к Солнцу: Марс Земля Венера Меркурий'*

**const** further = planets.reduce((acc, planet) => `${acc} ${planet}`, 'всё дальше от Солнца:'); *// 'всё дальше от Солнца: Меркурий Венера Земля Марс'*

* за "превращение" строки в массив, а так же за обратное действие отвечают, соответственно, split и join

**const** planets = 'Mercury,Venus,Earth,Mars';

console.log(planets.split(',')); *// [ 'Mercury', 'Venus', 'Earth', 'Mars' ]*

console.log(planets.split(',').join('-')); *// 'Mercury-Venus-Earth-Mars'*

* поиск в массиве и проверка соблюдения условий элементами массива (find, findIndex, indexOf, lastIndexOf, some, every, includes...)

*// includes определяет наличие элемента в массиве, возвращая true/false*

**const** planets = ['Mercury', 'Venus', 'Earth', 'Mars'];

planets.includes('Earth'); *// true*

planets.includes('Earth', 3); *// ищет, начиная с 3-й индексной позиции, поэтому false*

planets.includes('Saturn'); *// false*

* сортировка массива (sort, reverse...) и др. более или менее распространённые операции с массивами.

Ассоциативные массивы – Map

*// Создание пустого объекта*

**const** map = **new** Map();

**const** map = **new** Map()

map.set('key', 'value');

map.get('key'); *// value*

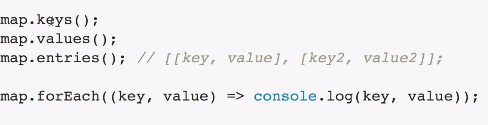
map.has('key'); *// true*

*// Если ключа нет, то вернется undefined*

map.get('unkown'); *// undefined*

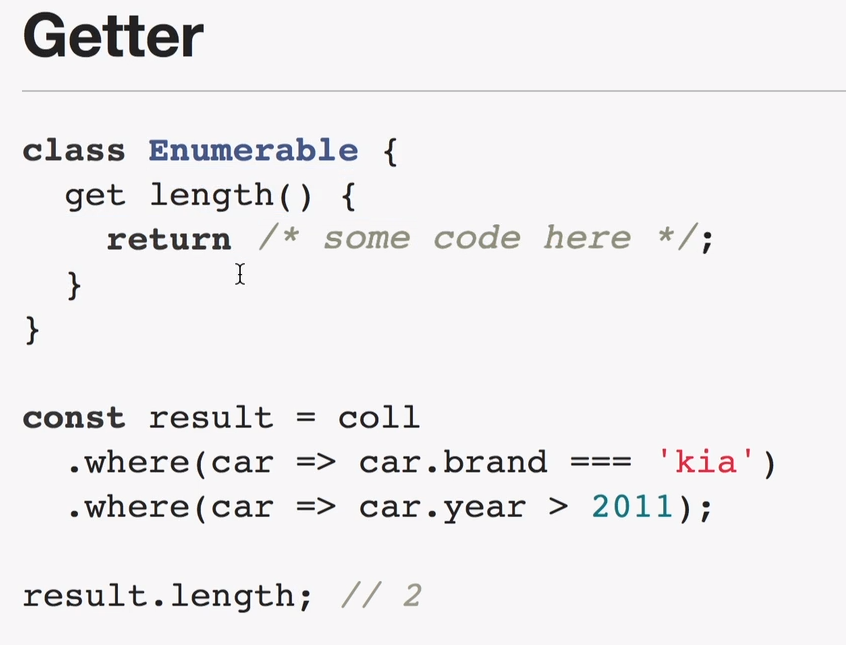
map.has('unkown'); *// false*

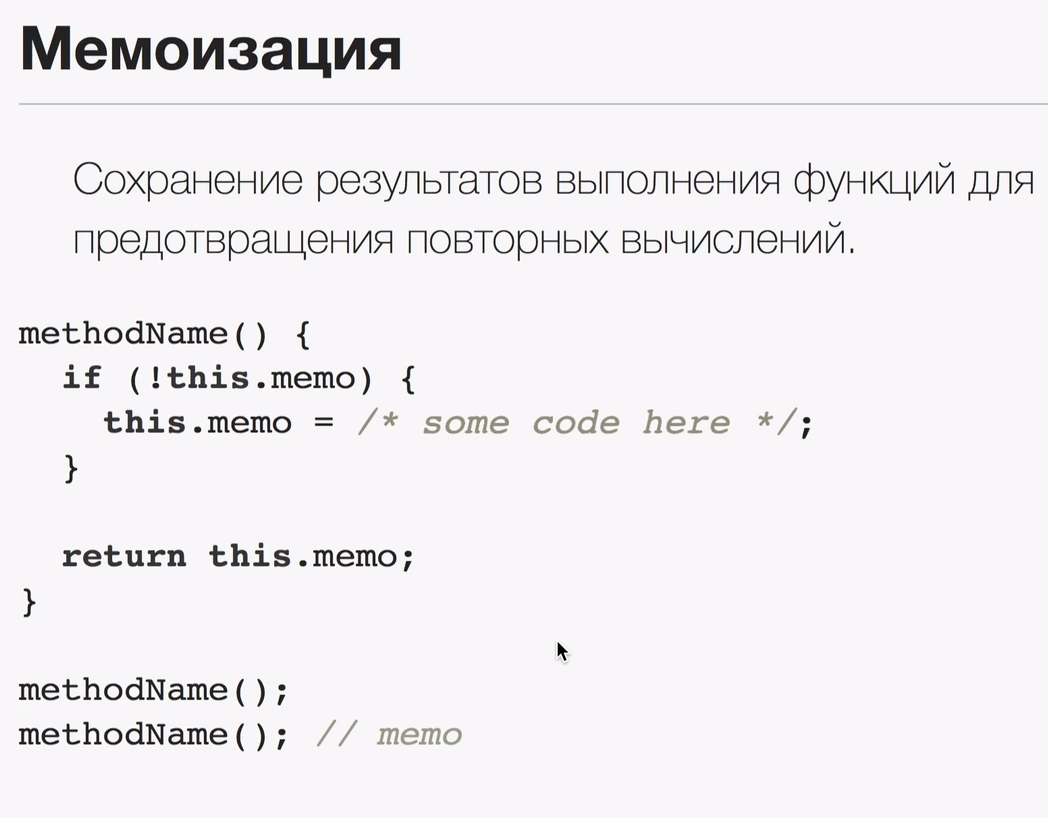
map.size; *// 2*

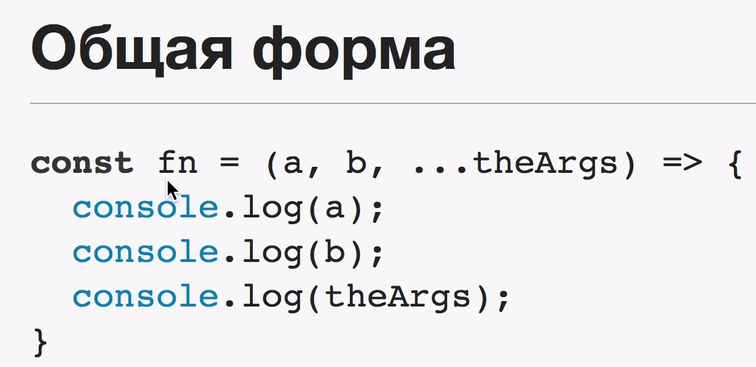


Неизменяемость 

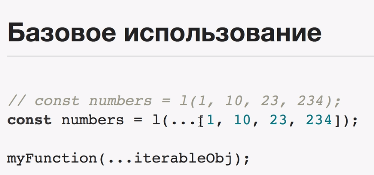






**Rest оператор**

**Spread оператор**



**Дестракчеринг**

<https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment>