Шпаргалка по объектно-ориентированному и функциональному программированию для JavaScript разработчика

ООП - парадигма программирования, в которой мы оперируем объектами, как основной действующей единицей (сущностью), которой есть свойства (- некоторое описание объекта) и методы (- действия, которые над этой сущностью совершаются). Например, у дверцы шкафа есть свойство "быть деревянной" и метод "открываться/закрываться".

В противовес ООП есть ФП, где мы оперируем как раз функциями или процедурами (не возвращающими ничего функциями), как основной единицей, вокруг которой мы строим модель нашего кода. Функция в таком случае уже будет не методом объекта, а процессом, у которого есть входные данные и результат. Например, в процесс закрытия шкафа поступает дверца и шкаф, а результатом выполнения этого процесса будет закрытый или открытый шкаф.

Эти два подхода не противоречат, а дополняют друг друга.

Наследование - возможность дочерних объектов принимать свойства и методы родительских объектов.
В JS: цепочка прототипов, extends у классов.

Полиморфизм - предоставление одного интерфейса для разных типов данных.
Пример: числа независимо от типа Number или BigInt ведут себя полиморфно - с ними можно проводить весь спектр мат.операций; шаблонное поле вывода, в которое мы можем подставлять любые значения или переменные.

Инкапсуляция - сокрытие данных реализации - упаковка данных и функций в один компонент (класс, модуль) и последующий контроль доступа к нему по принципу "чёрного ящика".
Пример: приватные поля # в классах, приватные переменные через IIFE, блочная область видимости.

Инкапсуляция vs Абстракция

Инкапсуляция = Сокрытие реализации и данных (защита внутреннего состояния).

Абстракция = Сокрытие сложности (проектирование интерфейса).

Абстракция - скрытие сложной реализации и предоставление простого интерфейса.
Пример: REST API - мы имеем дело с эндпоинтами, а на чём сам сервис и как написан нас мало интересует; классы в JS - это функция-конструктор реализующая абстракцию поверх прототипной модели.

Чистая функция - функция, возвращающая прогнозируемый результат (без побочных эффектов)

Иммутабельность - данные не изменяются, а создаются ссылки на новые.
Например, примитивы иммутабельны, а объекты мутабельны в JS.

High order function - функция, принимающая другую функцию (callback) в качестве аргумента

Каррирование (carrying) - разложение функции на несколько

Противоположные по смыслу концепции

Композиция функций - объединение нескольких функций в одну

Functors (функторы) - контейнеры, хранящие значения функций.
Например, массив функтор функции map

Классово-объектная модель ООП - это когда класс - это шаблон/чертёж, а объект - это экземпляр класса.
Пример: C#, Java, TS до компиляции.

Прототипная модель - это когда в код выстраивается в древовидную структуру от глобального объекта по цепочке прототипов к текущему объекту. Соответственно, по цепочке прототипов мы можем обратиться от дочерней сущности к родительской.
Пример: JavaScript

Компонентная модель - это когда в качестве шаблона (класса), свойств (атрибутов) и действий (методов) выступают не части кода, а отдельные архитектурные компоненты программы.
Например, в FSD в entities мы описываем бизнес-сущности, в features те действия, которые мы можем совершать, а в widgets собираем уже самостоятельный блок, который обладает своими свойствами и методами.