INDEX

```
import { INPUT } from './Input/input.js'

    INPUT();
```

Отсюда запускается функция, выполняющая блок “INPUT”

STATE

Object state

```
let figure = {
let coordField = {
let demoCoordField = {
let state = {
```

Здесь не обошлось без добавления других объектов

Единый объект, состояние приложения в любой момент времени

Setters

```
function setGameStatus(status) {
function refreshState() {
function setFigureType() {
function setScore(score) {
function setLevel() {
function setSpeed() {
```

ОПЕРАЦИИ С ДЕМО-ФИГУРОЙ

```
function addDemo(type) {
function drawDemo(type) {
function defineDemoFigure(type) {
```

ОПЕРАЦИИ С ОСАДКОМ

```
function killLiveFigure() {
function changeTagId() {
function addDeadField() {
function writeLays() {
function checkLays() {
function deleteLay() {
function dropUpperLays() {
function rewriteDeadFigure() {
```

Взаимодействуют со State, иногда запускают Output

ОПЕРАЦИИ С АКТИВНОЙ ФИГУРОЙ

```
function addLive(type) {
function Live (name, numberOfPixels, x, y) {
function drawLive(type) {
function defineFigure(type) {
function moveLiveFigure(dx, dy) {
function changeOrientation(type) {
function setNewPosition(checkResult) {
function clearLiveFigure() {
```

3. Сеттеры в состояние
Функции, обновляющие значения в объекте состояния. Например, могут быть такие функции:

- setGameStatus(status)
- setFallenCubes(matrix)
- removeFullRows()
- resetGame()
- setDoubleSpeed()
- и т.д

Getters

```
function checkFieldBorder(dx, dy) {
function checkRotation() {
function isLoser() {
function isAnyLayIsFull() {
```

Производят вычисления на основе данных из State, возвращают результат в Handlers

2. Геттеры из состояния
Функции, возвращающие другие важные данные,
вычисляя их из объекта состояния.
Например, могут быть такие функции.

getFullRows() — функция, которая проверяет,
есть ли полные строки в state.fallenCubes и возвращающая
массив индексов таких строк.
Поможет понять, какие строки нужно удалить.

isGameOver() — функция возвращает true|false,
проверяющая, не достал ли осадок до верха экрана

Геттеров, будет наверное немного.
Там где не надо вычислять, читай из состояния напрямую.
Но то, что можно вычислить, выноси в геттер.

Handlers

```
function startGame() {
function continueGame() {
```

Создание фигур

```
function handleAddFirstLive() {
function handleDemoNext() {
function addDead() {
function addNextLive() {
```

Движение фигуры

```
function handleFallFigure() {
function actionLiveFigure(key) {
function move(dx, dy) {
function rotation(type) {
```

Операции со слоями

```
function laysOperation() {
```

4. Обработчики событий.

Пусть в этих функциях будет логика обработки событий:

`handleRotate()`, `handleMove('left' | 'right)`, `handleResetGame()`, и т.д.

Дергают сеттеры, геттеры и иногда Output

INPUT

```
start.onclick
startGame();

continueBtn.onclick
continueGame()

actionLiveFigure(event.key)
```

Обработчики

II. Подписки на хоткеи и нажатия кнопок

Тут будут сгруппированы все (или большинство) addEventListener и onClick.

Используй тут функции-обработчики событий, которые ты импортируешь из модуля состояния:

```jsx
import { handleRotate, handleMove, handleSpeedUp } from './state'

// И потом:
case " ":
   handleRotate();
   break;
case "ArrowLeft":
   handleMove('left');
   break;
case "ArrowRight":
   handleMove('right');
   break;
case "ArrowDown":
   handleSpeedUp();
   break;
```

Представляй это, как “input” в рантайме твоего приложения.

OUTPUT

```
function showScore() {
function updateScore() {
function addTagLive() {
function addTagDemo() {
function gameOver() {
function vanishPix(elems) {
```

III. Функции, меняющие что-то в UI.
Представляй этот слой как “output” твоего приложения.

В нём будут находиться функции, которые добавляют, удаляют и редактируют объекты в DOM. Постарайся больше ничего не делать.

Например:

- updateFallingFigureCoordinates(newCoordinates)
- removeFullRows()
- showGameOverPopup()

Можешь дёргать эти функции из функций-обработчиков из модуля состояния.

Вообще, функции-обработчики, получатся жирными и содержать бОльшую часть бизнес логики, просто пока прими это. Потом можно порефакторить, и может быть применить новые паттерны.

В функциях для UI избегай расчётов со state,
а в геттерах не меняй напрямую UI.
Делай так, чтобы у каждой функции была своя зона ответственности.