

Гридин Александр

ПРЕЗЕНТАЦИЯ МИНИ-ПРОЕКТОВ В БЛОКЕ JS

Коротко о проектах

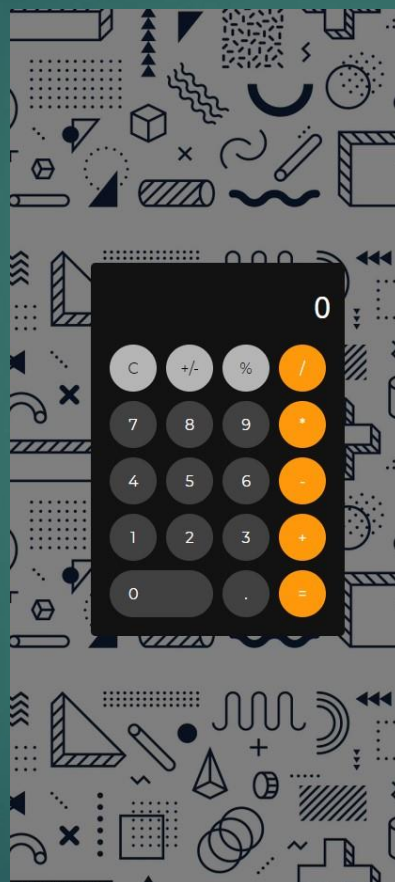
Генератор случайных чисел

Введите количество цифр в Вашем числе

Введите минимальное число

Введите максимальное число

Сгенерировать число



50/50

Что помогло людям писать до изобретения шариковых ручек?

A. пух

B. мех

C. перья

D. чешуя

15	1000000 ₴
14	500000 ₴
13	250000 ₴
12	125000 ₴
11	64000 ₴
10	32000 ₴
9	16000 ₴
8	8000 ₴
7	4000 ₴
6	2000 ₴
5	1000 ₴
4	500 ₴
3	300 ₴
2	200 ₴
1	100 ₴

Александр

Выигрыш:

0₴

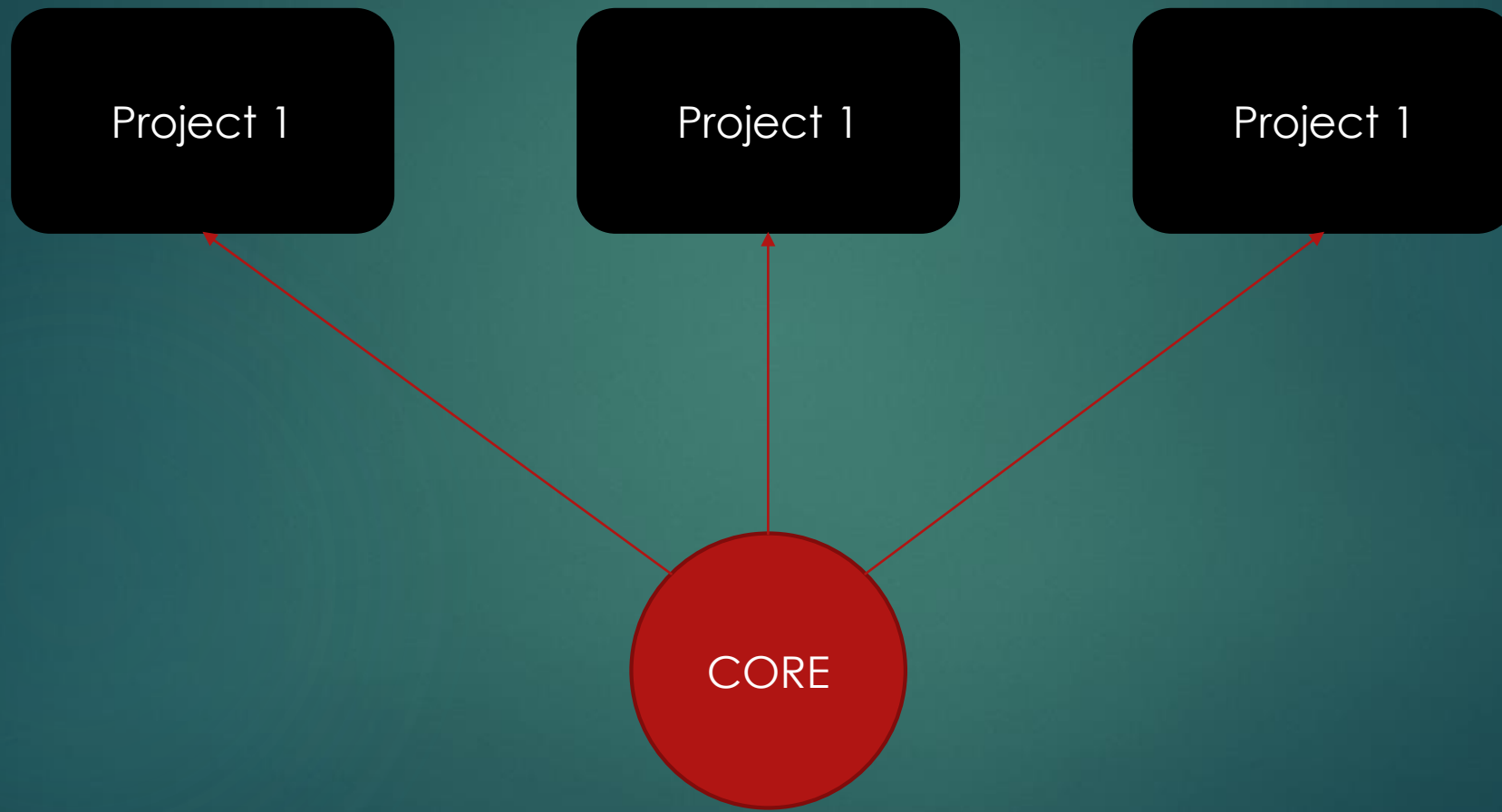
Забрать деньги.


Коротко о проектах

- ▶ **Генератор случайных чисел** – генерирует числа случайным образом
- ▶ **Калькулятор** – вполне предсказуемо считает
- ▶ **Кто хочет стать миллионером?** – получилось вполне играбельно

Ничего особенного снаружи....., но с изюминкой внутри ☺

В чем же изюминка?





Что же это за CORE
такой?...

А вот и я!



Примечательно, что общего с React и Vue здесь только логотипы и кусочки названий, а также, Желание разделить проекты на «компоненты»

Зачем? С какой целью?

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Кто хочет стать миллионером?</title>
8
9   <link rel="preconnect" href="https://fonts.gstatic.com">
10  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
11  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.3/css/all.css"
12    integrity="sha384-SZxX4whj79/gErwc0Yf+zWLeJdY/qpuqC4cAa9rOGUstPomtqpuNWT9wdPEn2fk" crossorigin="anonymous">
13  <link rel="stylesheet" href="./css/style.css">
14 </head>
15 <body>
16
17   <div id="revue-app"></div>
18
19   <script type="module" src="index.js"></script>
20 </body>
21 </html>
22
23
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Генератор случайных чисел</title>
8
9   <link rel="preconnect" href="https://fonts.gstatic.com">
10  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
11
12  <link rel="stylesheet" href="./css/style.css">
13 </head>
14 <body>
15
16   <div id="revue-app"></div>
17
18   <script type="module" src="index.js"></script>
19 </body>
20 </html>
21
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Калькулятор</title>
8
9   <link rel="preconnect" href="https://fonts.gstatic.com">
10  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
11
12  <link rel="stylesheet" href="./css/style.css">
13 </head>
14 <body>
15
16   <div id="revue-app"></div>
17
18   <script type="module" src="index.js"></script>
19 </body>
20 </html>
21
```

Файлы **index.html** каждого из проектов

Зачем? С какой целью?

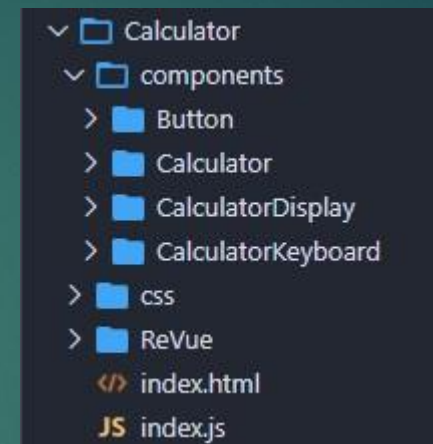
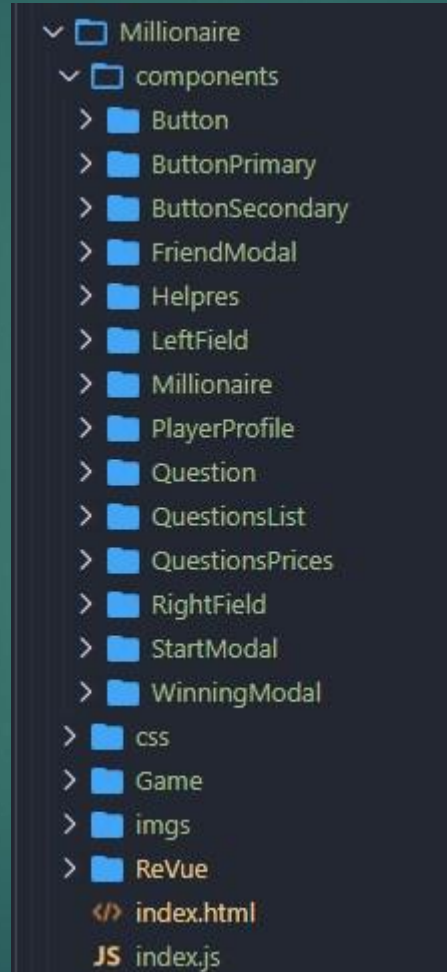
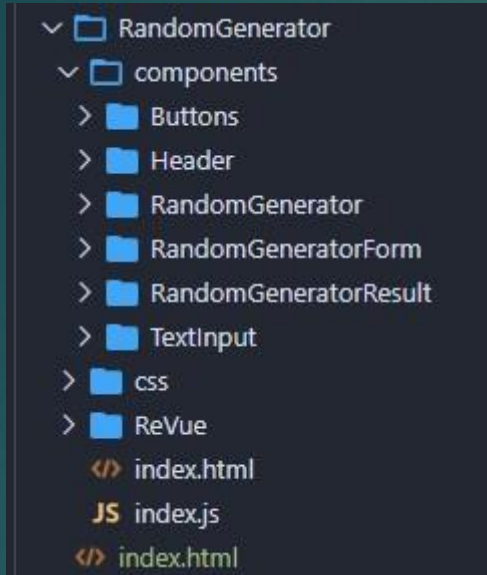
```
1 import ReVue from './ReVue/ReVue.js';
2 import Calculator from './components/Calculator/Calculator.js';
3
4 ReVue.sayHello();
5
6 let calculator = new Calculator();
7
8 ReVue.build([
9   calculator
10]);
11
12 console.log(ReVue.components);
13
```

```
1 import ReVue from './ReVue/ReVue.js';
2 import Millionaire from './components/Millionaire/Millionaire.js';
3
4 ReVue.sayHello();
5
6 let millionaire = new Millionaire();
7
8 ReVue.build([
9   millionaire
10]);
11
12 console.log(ReVue.components);
13
```

```
1 import ReVue from './ReVue/ReVue.js';
2 import RandomGenerator from './components/RandomGenerator/RandomGenerator.js';
3
4 ReVue.sayHello();
5
6 let randomGenerator = new RandomGenerator();
7
8 ReVue.build([
9   randomGenerator,
10]);
11
12 console.log(ReVue.components);
13
```

Файлы [index.js](#) каждого из проектов

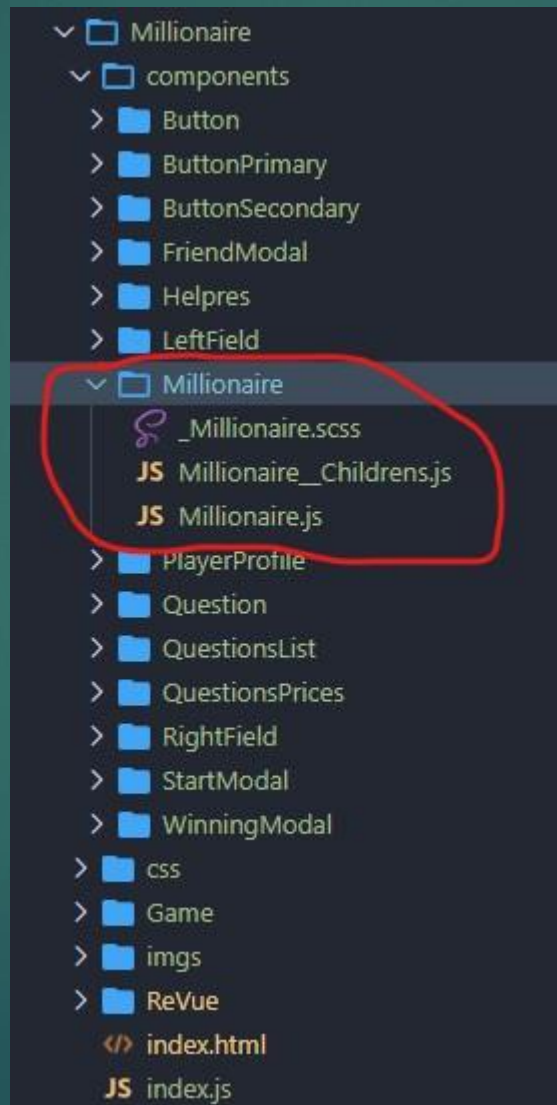
Зачем? С какой целью?



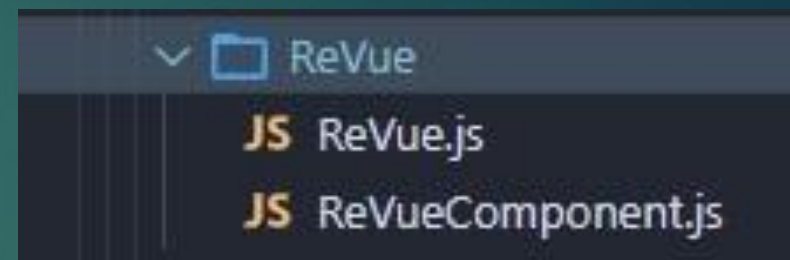
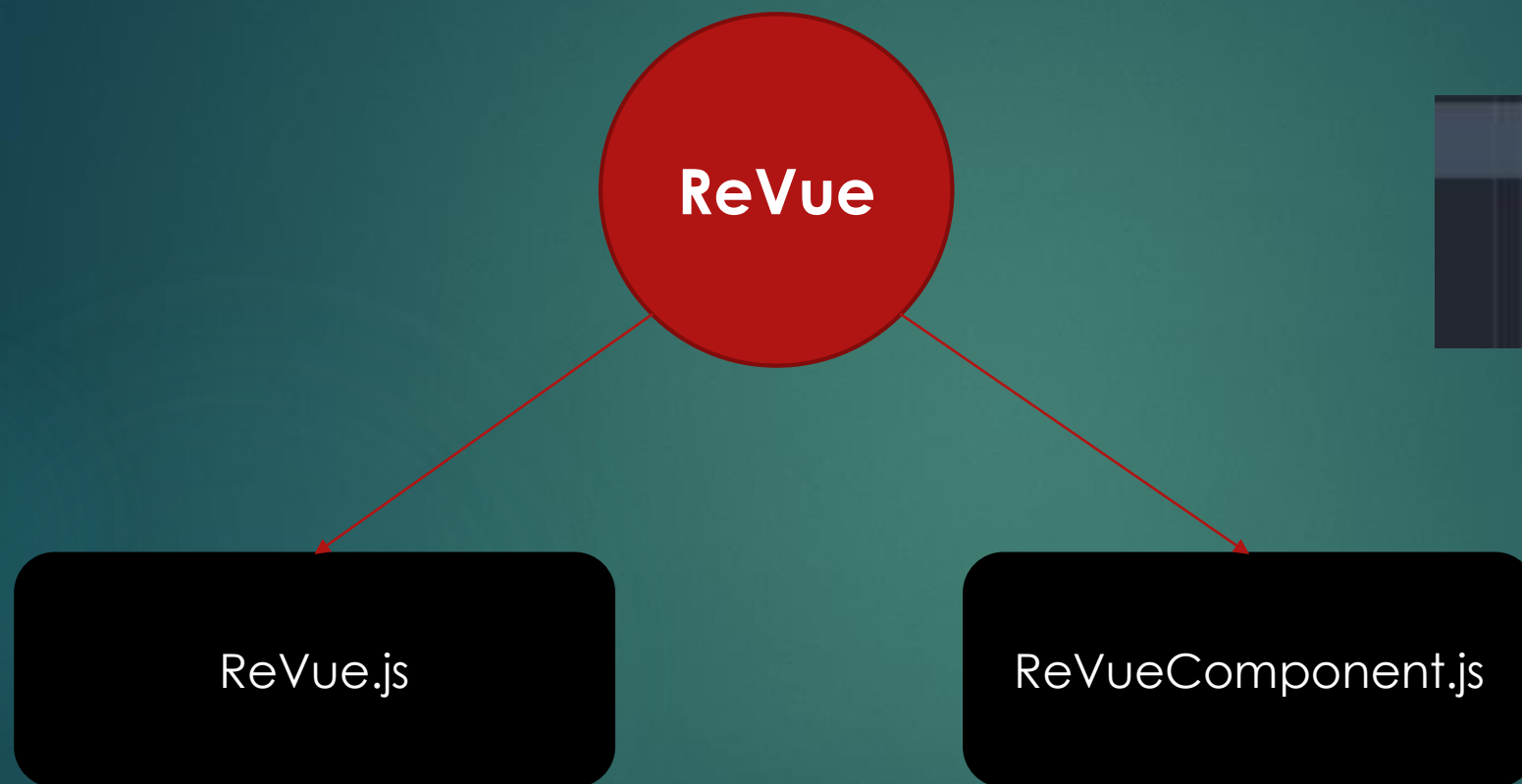
Файловая структура каждого из проектов

Зачем? С какой целью?

Ну и развернем папки
некоторых компонентов



Давайте познакомимся по ближе



ReVue.js

```
1  const ReVue = {  
2    components: [],  
3    elementsCounter: 0,  
4  
5    > sayHello(){ ...  
7    },  
8  
9    > getComponentByName(name){ ...  
15   },  
16  
17   > build(components){ ...  
29   }  
30 }  
31  
32 > function mergeComponents(parent){ ...  
42 }  
43  
44 export default ReVue;  
45
```

getComponentByName() – метод получения компонента по его имени из массива **components**

build() – метод, который «строит» итоговое приложение

mergeComponents() – вспомогательная функция, для объединения компонентов в один общий DOM элемент. Работает рекурсивно и вызывается из метода **build()**

ReVueComponent.js

```
1 import ReVue from './ReVue.js';
2
3 class ReVueComponent{
4   constructor(props){
5     this.name = null;
6     this.id = null;
7     this.tagName = null;
8     this.attributes = null;
9     this.props = props;
10    this.element = null;
11    this.childrens = [];
12    this.eventListeners = null;
13    this.parent = null;
14  }
15
16  > initElement(data){ ...
26  }
27
28  > setChildrens(){ ...
36  }
37
38  > setName(name){ ...
41  }
42
43  > setEventListeners(eventListeners){ ...
46  }
47
48  > setProperties(props){ ...
51  }
52
53  > buildElement(){ ...
69  }
70
71  > deleteComponent(){ ...
83  }
84
85  > addClassNames(str){ ...
88  }
89
90  > addAttributes(obj){ ...
96  }
97
98  > sayHi(){ ...
101  }
102  }
103
104  > function deleteChilds(parent){ ...
120  }
121
122  > function handleElementAttributes(element, attributes){ ...
146  }
147
148  > function handleElementEventListeners(element, eventListeners){ ...
154  }
155
156  export default ReVueComponent;
157
```

initElement() – метод для быстрой инициализации компонента, если он создается через `new ReVueComponent()`

setChildrens() – устанавливает взаимосвязь между `parent` и `child`

setName() – устанавливает имя компонента (имя необходимо для получения компонента с помощью `ReVue.getComponentByName()`)

setEventListeners() – устанавливает обработчики событий (лучше, конечно, было его назвать `addEventListeners()`)

setProperties() – устанавливает `props`

buildElement() – на основании свойств создает DOM элемент и записывает в свойство `element`

deleteComponent() – удаляет компонент и всех его `childrens` как из массива `ReVue.components`, так и из DOM (работает рекурсивно)

addClassNames() – добавляет CSS классы

addAttributes() – добавляет атрибуты

Пример создания компонента с помощью `new ReVueComponent()`

```
let container = new ReVueComponent().initElement({
  tagName: 'div',
  attributes: {
    classNames: 'millionaire__friend-modal-container'
  },
  childrens: [
    title,
    message,
    confirmButton,
  ]
});
```

Данный способ предназначен для создания компонента «на лету», без необходимости создания под него отдельного класса.

Удобно для элементов-оберток и просто малозначимых элементов

Пример создания компонента с помощью наследования класса **ReVueComponent**

```
1 import ReVueComponent from '../ReVue/ReVueComponent.js';
2 import {childrens} from './FriendModal__Childrens.js';
3
4 class FriendModal extends ReVueComponent{
5   constructor(){
6     super();
7
8     this.tagName = 'div';
9     this.name = 'FriendModal';
10    this.attributes = {
11      classNames: 'millionaire__friend-modal hidden'
12    };
13
14    this.childrens = childrens;
15    this.setChildrens().buildElement();
16  }
17 }
18
19 export default FriendModal;
```

Теоретически, все компоненты должны быть повторно используемыми, но это не точно....и связано это с принципом работы `ReVue.getComponentByName()`, но об этом позже...

В конструктор компонента также можно передавать props

```
1 import ReVueComponent from '../ReVue/ReVueComponent.js';
2
3 class Button extends ReVueComponent{
4   constructor(props){
5     super(props);
6
7     this.tagName = 'button';
8     this.attributes = {
9       classNames: 'button',
10      type: 'button',
11    }
12
13    if(props && props.text){
14      this.attributes.textContent = props.text;
15    }
16
17    if(props && props.value){
18      this.attributes.value = props.value;
19    }
20  }
21 }
22
23 export default Button;
```

Это очень удобно, когда нужно создать несколько одинаковых компонентов но с разным текстом, например, как кнопки ответов в Миллионере, ну и карточки с вопросами конечно же.

Создаем элементы списка с кнопками

```
props.answers.forEach((answer, i) => {  
  let answerButton = new ButtonPrimary({  
    text: answer,  
    value: i,  
  });  
  
  answerButton.value = i;  
  answerButton.buildElement();  
  
  let answerLi = new ReVueComponent().initElement({  
    tagName: 'li',  
    childrens: [answerButton]  
  });  
  
  answerButtons.push(answerLi);  
});
```

Функция генерации карточек с вопросами в Миллионере

```
function prepareChildrens(questionsData){  
  let childrens = [];  
  
  questionsData.forEach((dataItem) => {  
    let question = new Question({  
      questionText: dataItem.question,  
      answers: dataItem.answers,  
      correctAnswer: dataItem.correctAnswer,  
    }).buildElement();  
  
    childrens.push(question);  
  });  
  
  return childrens;  
}
```

Компоненту можно прописывать свои уникальные св-ва и методы

```
class Calculator extends ReVueComponent{
  constructor(){
    super();

    this.tagName = 'div';
    this.name = 'Calculator';
    this.attributes = {
      classNames: 'calculator'
    };

    this.childrens = prepareChildrens();
    this.setChildrens().buildElement();

    this.result = 0;
    this.operands = [];
    this.status = 'init';
    this.currentOperation = '';
  }

  handleOperation(a, b, operation){
    if(operation){
      switch (operation) {
        case '+':
          return a + b;
        case '-':
          return a - b;
        case '/':
          return a / b;
        case '*':
          return a * b;
      }
    }

    return a;
  }
}
```

Например у калькулятора есть доп. св-ва result, operands, status, currentOperation и метод handleOperation(), что, как по мне, довольно неплохо отражается на коде:

```
if(calculator.operands.length === 2 && button.value === '='){
  console.log(calculator.currentOperation)

  calculator.result = calculator.handleOperation( ... calculator.operands, calculator.currentOperation);

  if(!isFinite(calculator.result)){
    calculator.operands = [];
    calculator.status = 'error';
    displayInput.element.value = 'Ошибка';
    calculator.currentOperation = '';

    return;
  }

  calculator.operands = [];
  calculator.status = 'result';
  displayInput.element.value = calculator.result;
  calculator.currentOperation = '';

  return;
}
```

Это фрагмент кода, описывающий поведение кнопки «=» в калькуляторе

Так в чем подвох?

Это все хорошо работает при верстке проекта с помощью JS, но дискомфорт начинается при интеграции логики...

```
55 function handleAnswers(answers){
56   return function(e){
57     if(e.target.tagName === 'BUTTON'){
58       let answerButton = e.target;
59       let answerButtonComponent = ReVue.getComponentByName(answerButton.dataset.name);
60       let answersList = answerButtonComponent.parent.parent;
61       let question = answersList.parent;
62       let questionPrices = ReVue.getComponentByName('QuestionsPrices');
63       let playerProfileMoneys = ReVue.getComponentByName('playerEarnedMoneysValue');
64
65       answersList.childrens.forEach((li) => {
66         li.childrens[0].element.setAttribute('disabled', 'true');
67       });
68
69       if(!question.hasAnswer){
70         answerButton.classList.add('warning');
71         question.hasAnswer = true;
72
73         setTimeout(() => {
74           if(+answerButton.value === Game.questions[Game.currentQuestionIndex].correctAnswer){
75             answerButton.classList.add('success');
76             Game.currentQuestionIndex++;
77
78             setTimeout(() => {
79               questionPrices.childrens.forEach((li) => {
80                 if(li.element.classList.contains('current')){
81                   li.element.classList.remove('current');
82                 }
83               });
84
85               questionPrices.childrens[Game.currentQuestionIndex-1].element.classList.add('current');
86               Game.playerCurrentMoneys = questionPrices.childrens[Game.currentQuestionIndex-1].value;
87               playerProfileMoneys.element.textContent = `${Game.playerCurrentMoneys}${Game.currency}`;
88               if(questionPrices.childrens[Game.currentQuestionIndex-1].constant){
89                 Game.playerEarnedMoneys = Game.playerCurrentMoneys;
90               }
91               answerButtonComponent.parent.parent.parent.deleteComponent();
92
93               if(Game.currentQuestionIndex === Game.questions.length){
94                 let gameWrapper = ReVue.getComponentByName('MillionaireGameWrapper');
95                 gameWrapper.deleteComponent();
96
97                 let winningModal = ReVue.getComponentByName('WinningModal');
98                 winningModal.element.classList.remove('hidden');
99               }
100             }, 1500);
101           } else {
102             answerButton.classList.add('error');
103             answerButtonComponent.parent.parent.childrens[Game.questions[Game.currentQuestionIndex].correctAnswer].childrens[0].element.classList.add('success');
104
105             setTimeout(() => {
106               let winningModal = ReVue.getComponentByName('WinningModal');
107               winningModal.childrens[0].childrens[0].element.innerHTML = `<h2>Игра завершена!</h2><div class="text">Вы выиграли <br /><span class="sum">${Game.playerEarnedMoneys}${Game.currency}</span></div>`;
108               winningModal.element.classList.remove('hidden');
109             }, 2000);
110           }
111         }, 1500);
112       }
113     }
114   }
115 }
```

Реализация работы кнопок
ответов в Миллионере...

Несовершенство метода ReVue.getComponentByName()

```
9   getComponentByName(name){  
10     let findedComponent = this.components.find((component) => {  
11       return component.name === name;  
12     });  
13  
14     return findedComponent;  
15   },  
16
```

Тут Вы все и так видите...
Просто возвращается первый найденный
элемент массива.

Из-за этого получается, что на странице, к примеру, не может быть одновременно 2 калькулятора, т.к. логика, построенная через `ReVue.getComponentByName()`, просто начнет работать некорректно.

Теоретически это можно было бы решить вводом какого-то уникального контекста, но сроки поджимали, сами понимаете 😊

В мире нету ничего идеального 😊

Вообще, много чего тут можно было бы доработать и улучшить, но целью мини-проектов не было создание и презентация инструментария для их реализации, по этому везде пришлось идти на компромиссы и маленькие костылики 😊

Ну и еще раз внимательно посмотрите на логотип (а он достаточно честный):



Ну а теперь немного
мини-проектов

