Front-end

JavaScript

JS. Rest, spread

- rest
- spread
- деструктуризация

JS. Rest, spread

Синтаксис

...rest

...spread

Они используют одинаковый синтаксис, но решают разные задачи.

JS. Rest, spread

Использование

Spread используется для разделения коллекций на отдельные элементы, а rest, наоборот, для соединения отдельных значений в массив

JS. Rest, spread

Spread (оператор расширения)

Spread оператор – это три точки перед именем переменной (или константы), внутри определения объекта. Он раскладывает соответствующий объект внутри нового объекта. С его помощью можно получить только копию, он не может изменять существующие объекты.

Позволяет разложить массив на отдельные значения.

Spread-оператор предоставляет самый простой способ скопировать элементы одного массива в другой

JS. Rest, spread

Spread (оператор расширения)

Клонирование свойств массивов

```
let arr = ['will', 'love'];

let data = ['You', ...arr, 'spread', 'operator'];

console.log(data); // ['You', 'will', 'love', 'spread', 'operator']
```

Подобным образом можно скопировать и весь массив целиком

```
let arr = [1, 2, 3, 4, 5];

let data = [...arr];

console.log(data); // [1, 2, 3, 4, 5]
```

Важно понимать, что при подобном использовании оператора ... происходит именно копирование всех свойств, а не ссылки на массив.

JS. Rest, spread

Spread (оператор расширения)

Лучший способ конкатенации массивов

Для конкатенации массива часто используется concat:

```
let arr1 = [0, 1, 2];

let arr2 = [3, 4, 5];

arr1 = arr1.concat(arr2);
```

С использованием spread syntax:

```
let arr1 = [0, 1, 2];

let arr2 = [3, 4, 5];

arr1 = [...arr1, ...arr2];
```

JS. Rest, spread

Rest (остаточный параметр)

rest означает, что нужно взять все элементы, которые остались от деструктуризации и поместить их в массив с именем rest . Этому массиву можно дать любое имя. Rest срабатывает в самом конце, когда все остальные данные уже разложены по своим константам (или переменным). Именно поэтому он называется rest (оставшиеся).

Возвращает массив

JS. Rest, spread

Rest (остаточный параметр)

Оператор ... интерпретируется по-разному, в зависимости от контекста применения. Spread используется для разделения коллекций на отдельные элементы, а rest, наоборот, для соединения отдельных значений в массив.

```
let log = function(a, b, ...rest) {

  console.log(a, b, rest);

};

log('Basic', 'rest', 'operator', 'usage'); // Basic rest ['operator', usage]
```

Используя параметр ...rest в функции log вы говорите интерпретатору: “собери все оставшиеся элементы в массив с именем rest”.

```
let log = function(...args) {

  conole.log(args);

};

log(1, 2, 3, 4, 5); // [1, 2, 3, 4, 5]
```

Таким образом, больше нет необходимости переводить псевдо-массив arguments функций в настоящий массив – оператор rest сделает это сам

JS. Rest, spread

Rest (остаточный параметр или остаток)

rest используется только в конце, а spread может использоваться в любом месте

JS. Rest, spread

Деструктуризация

Что такое деструктуризация массива?

Деструктуризация массива (англ. array destructuring) — это особый синтаксис, позволяющий извлекать значения из массива и записывать их в новые переменные с минимумом кода.

Например, без использования синтаксиса деструктуризации массива значение элемента массива копируется в новую переменную следующим образом:

```
const profile = ["John", "Doe", "code.com"];


const firstName = profile[0];

const lastName = profile[1];

const website = profile[2];
```

JS. Rest, spread

Деструктуризация

Деструктуризация просто подразумевает разбивку сложной структуры на простые части. В JavaScript, таковая сложная структура обычно является объектом или массивом. Используя синтаксис деструктуризации, вы можете выделить маленькие фрагменты из массивов или объектов. Такой синтаксис может быть использован для объявления переменных или их назначения.

В деструктуризации массивов, вы используете литерал массива с левой стороны выражения назначения. Каждое имя переменной в литерале массива соответствует элементу с таким же индексом в деструктуризированном массиве. Вот быстрый пример.

```
const rgb = [255, 200, 0];

// Деструктуризация массива

const [red, green, blue] = rgb;
```

JS. Rest, spread

Деструктуризация. Значения по умолчанию

```
const rgb = [200];

// Назначаем дефолтные значения 255 для red и blue

const [red = 255, green, blue = 255] = rgb;
```

JS. Rest, spread

Деструктуризация

Обратите внимание, что в приведенном выше фрагменте много повторяющегося кода, что не удовлетворяет принципу DRY.

А теперь посмотрим, как деструктуризация массива позволит нам сделать код более аккуратным и лаконичным.

```
const profile = ["John", "Doe", "code.com"];

const [firstName, lastName, website] = profile;
```

JS. Rest, spread

Деструктуризация

Деструктуризация массива напрямую

JavaScript позволяет деструктурировать массив напрямую следующим образом:

```
const [firstName, lastName, website] = [
  "John",
"Doe",
"code.com"];
```

JS. Rest, spread

Деструктуризация

А что если понадобится присвоить "Oluwatobi" переменной firstName, а остальные элементы массива — другой переменной? Как это сделать? Давайте разберемся и с этим вопросом.

Использование деструктуризации массива для присваивания переменной остальной части литерала массива

JavaScript позволяет использовать оператор остатка внутри деструктурирующего массива для присваивания переменной значений остальных элементов обычного массива.

Рассмотрим пример:

```
const [firstName, ...otherInfo] = ["John", "Doe", "code.com"];
```

JS. Rest, spread

Деструктуризация

Обмен переменными

Одно очень хорошее примечание в деструктуризации массивов заключается в обмене локальными переменными. Представьте, что вы делаете приложение манипуляций с фото и хотите менять height и width размеры для них, при ориентировке фото измененного в landscape и portrait. Старый способ сделать это будет выглядеть примерно так

JS. Rest, spread

Деструктуризация

```
let width = 300;
let height = 400;
const landscape = true;
console.log(`${width} x ${height}`); // 300 x 400
if (landscape) {
        let initialWidth = width; // доп. переменная, чтобы скопировать изначальную ширину
        width = height; // меняем значения переменных
        height = initialWidth; // высота теперь равна скопированному значению ширины
console.log(`${width} x ${height}`); // 400 x 300
}
```

JS. Rest, spread

Деструктуризация

Код выше решает задачу, хотя мы и использовали дополнительные переменные, чтобы скопировать значение одной из смененных переменных. С помощью деструктуризации, мы можем проделать такой обмен простым назначением. Следующий код покажет, как с помощью деструктуризации можно это провернуть:

```
let width = 300;

let height = 400;

const landscape = true;

console.log(`${width} x ${height}`); // 300 x 400

if (landscape) {

        [width, height] = [height, width]; // меняем значения переменных

        console.log(`${width} x ${height}`); // 400 x 300

}
```