|  |  |
| --- | --- |
| **Объект** – это один из типов данных в JavaScript, который предназначен для хранения коллекции различных значений и более сложных сущностей.  Значением объекта может быть любой тип данных и даже функция  **Свойства (Свойства состоят из ключа и значения):**  **Обращение к свойствам** выполняется через точку или квадратные скобки:  const firstName = person.firstName;  **Значения, связанные с соответствующими ключами** можно не только получить, но и присвоить им новые значения:  person.firstName = 'Иван';  **А также добавить новые** свойства объекту:  person.middleName = 'Сергеевич';  **Удаление свойств из объекта** осуществляется с помощью оператора delete:  delete person.middleName;  **Проверить наличия ключа** в объекте можно посредством оператора in:  Console.log( 'firstName' in person) // true  **Если ключ содержится в переменной, то получить значение этого свойства можно только через квадратные скобки:**  const key = 'author of post';  const value = someObj[key];  **Перебор объектов:**  **1. Цикл for...in**  let cap = {  color: "Red",  volume: 300,  hasImage: true,};  for (let name in cap) {  alert(name); //color, volume, hasImage  alert(cap[name]); //Red. 300, true  **2. for…of - Object.keys, values, entries – перебор элементов (преобразование объекта в массив)**  const car = {  brand: 'Ford',  color: 'blue'}  const keys = Object.keys(car); // ['brand', 'color']  const values = Object.values(car); // ['Ford', 'blue']  const = Object.entries(car); // [['brand', 'Ford'], ['color', 'blue']]  **Метод Object.values()** в JavaScript — позволяет получить все значения, заключенные в объекте.  Object.values(название объекта);  **Метод Object.entries() —**возвращает многомерный массив, содержащий как ключи, так и значения объекта. Object.entries(название объекта)  **Метод Object.keys()** возвращает массив, содержащий все ключи внутри объекта и передаваемый в качестве аргумента. Object.keys(название объекта)  **Object.freeze()** позволяет заморозить объект, передаваемый ему в качестве аргумента. После этого нельзя будет обновить объект или добавить ему новые свойства. Object.freeze(название объекта)  **Метод Object.seal()** предотвращает добавление новых свойств к объекту, но позволяет обновлять и изменять свойства внутри объекта. Object.seal(название объекта)  **Метод Object.is()—** определяет, одинаковы ли два значения. Он возвращает true, если оба значения равны. В противном случае возвращается false.  Object.is(obj1, obj2)  **Object.create()—** позволяет создать новый объект на основе прототипа уже существующего объекта. | **Сравнение объектов выполняется по ссылкам:**  console.log( objA === objB )  **Упорядочение свойств объекта**  Свойства объекта упорядочены следующим образом: сначала идут целочисленные типы в порядке возрастания, остальные – в порядке создания.  let a = { "b":2, 3:"4", 1:"b"};  for(let c in a) {  alert(a[c]); // b, 4, 2  **Копирование объектов**  Переменная, содержащая объект на самом деле содержит не сам объект, а только ссылку на него. При копировании объектов в отличие от значений примитивных типов происходит передача ссылки.  // присвоим переменной student1 объект, а точнее ссылку на него  const student1 = {name: 'Carl'};  // присвоим объект, содержащийся в student1 переменной student2  **const student2 = student1;**  **метод Object.assign()** – создает не ссылку, а новый объект с такими же свойствами:  const student1 = Object.assign({}, student);  {} – создает новый объект;  Student – копируем объект  **Для копирования нескольких объектов**  const target = { a: 1 }; const source1 = { b: 2 }; const source2 = { c: 3 };  Object.assign(target, source1, source2);  Объект, в который нужно скопировать указывается в качестве первого аргумента, а те из которых – после него:  **Используя для дублирования цикл for…in** мы напрямую переписываем все свойства из объекта в объект.  let skills = {  "1": "HTML",  "2": "CSS",  "3": "JavaScript",};  let mySkills = {};  for (let key in skills) {  mySkills[key] = skills[key];}  **Спред-синтаксис (spread)** ... позволяет передавать итерируемые коллекции (например, массивы или строки) как список аргументов функции или добавлять содержащиеся в них элементы в новый массив.  **spread** применятся и для объектов, чтобы копировать пары ключ-значение из одного объекта в другой.  Есть три контекста, в которых он применяется:  **При вызове функции:**  function multiplyThreeNumbers(a, b, c) {  return a \* b \* c}  const nums = [1, 2, 3]  console.log(multiplyThreeNumbers(...nums))// 6  **При создании массивов с помощью литерала []**  const movies = ['Rocky', 'Terminator 2', 'The Matrix']  const series = ['Prison Break', 'Rick&Morty', 'Lost']  const watched = [...movies, ...series]  Данным способом можно создать копию массива или объеденить в массива в 1.  **При создании объекта с помощью литерала {}**  const person = { name: 'Иван', lastName: 'Объектов' }  const userData = { username: 'killer3000', ...person }  Если свойства в новом и старом объекте совпадают, то будет использоваться значение свойства, которое шло последним.  Если поставить спред в начало, то будет использоваться новое имя |