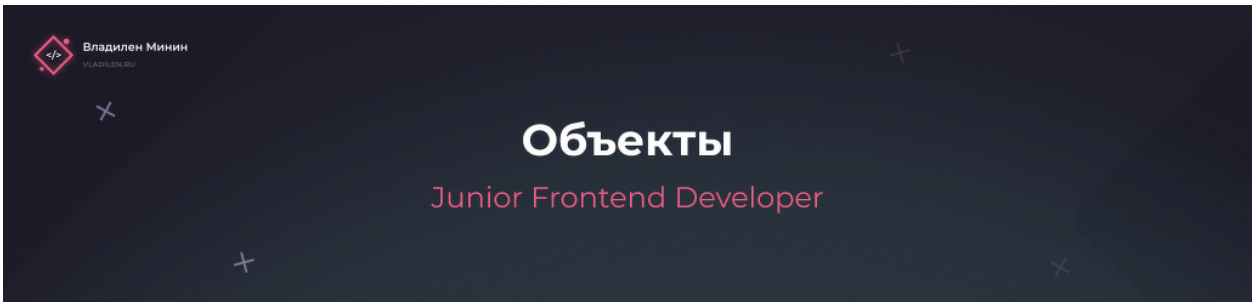

Цель урока:

Вы научитесь использовать объекты и выполнять базовые действия над ними: добавление, изменение и удаление свойств, перебор объектов, объединение нескольких объектов в один.

Содержание урока:

1. Создание объектов. Получение значений объектов по ключу
2. Удаление, добавление и изменение свойств объектов
3. Объект - ссылочный тип данных
4. Перебор объектов. Создание массивов из объектов (keys, values, entries)
5. Работа с ключами объекта
6. Объединение нескольких объектов в один
7. Опциональная цепочка

Дополнительные материалы:

1. Как использовать Symbol в качестве ключей объекта? https://learn.javascript.ru/symbol
2. Как получить массив всех symbol-ов в объекте? https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/...
3. Методы объектов: https://developer.mozilla.org/...
4. Опциональная цепочка “?.”: https://learn.javascript.ru/optional-chaining

---

Задание #1

Вам дан массив пользователей users. У каждого из них есть свойство status, которое может равняться или “online”, или “offline”.

```
const users = [
    {
        username: 'David',
        status: 'online',
        lastActivity: 10
    },
    {
        username: 'Lucy',
        status: 'offline',
        lastActivity: 22
    },
```

```
    {
        username: 'Bob',
        status: 'online',
        lastActivity: 104
    }
]
```

Вам необходимо создать новый массив onlineUsers, который будет содержать объекты только тех пользователей, у которых status равен “online”.

После выведите через alert сообщение “Сейчас в онлайн следующие пользователи: usersOnlineNames”, где usersOnlineNames - строка, в которой имена пользователей отображаются через запятую.

Для кода выше результат должен быть следующим: “Сейчас в онлайн следующие пользователи: David, Bob”.

Задание #2

Представьте, что вы разрабатываете программу выдачи талончиков для местной больницы. До этого талончики выписывались вручную и вам необходимо оптимизировать данную задачу.

Вам требуется создать функцию giveTalonsInOrder, которая сортирует очередь из пациентов. Она принимает в себя 2 параметра:

1. patients - массив объектов. Каждый объект хранит информацию об имени пациента и его уникальном номере - id.
2. orders - массив уникальных номеров id, который указывает порядок, в котором должны стоять в очереди пациенты.

Функция должна возвращать новый массив, в котором объекты из массива patients будут отсортированы по id из массива orders.

Посмотрите на возможный результат функции giveTalonsInOrder:

```
const ordersArr = [4, 2, 1, 3];
const people = [
    { id: 1, name: "Максим" },
    { id: 2, name: "Николай" },
    { id: 3, name: "Ангелина" },
    { id: 4, name: "Виталий" },
];

const result = giveTalonsInOrder(people, ordersArr);
console.log('result', result);
/* Возвращает массив
[
    { id: 4, name: 'Виталий' },
    { id: 2, name: 'Николай' },
    { id: 1, name: 'Максим' },
```

```
   { id: 3, name: 'Ангелина' }
]
*/
```

Задание #3

Вам необходимо создать функцию, которая будет работать с объектами. Назовите ее handleObject.

handleObject принимает в себя 3 параметра:

1. obj. Объект, с которым будет работать функция
2. key. ключ объекта
3. action. Действие, которое мы будем совершать над объектом

Параметр action может быть 4-мя значениями:

1. ‘get’. Если action равен ‘get’, то функция handleObject должна вернуть значение ключа key в объекте obj.
2. ‘add’. Если action равен ‘add’, то функция handleObject должна добавить новый ключ key в объект object и присвоить значение пустой строки “”. Также из функции необходимо возвратить обновленный объект obj.
3. ‘delete’. Если action равен ‘delete’, то функция handleObject должна удалить свойство key из объекта obj и возвратить обновленный объект.

Если action равен любому другому значению, то функция **handleObject** должна возвратить объект obj.

Протестируйте функцию на этом коде:

```
const student = {
   name: 'Maxim',
   programmingLanguage: 'JavaScript',
}

const result = handleObjects(student, 'programmingLanguage',
'delete');
console.log('result', result);
```

Задание #4

Вам необходимо создать функцию **giveJobToStudent**, которая будет добавлять новое свойство в объект и выводить информацию в модальном окне. Она принимает в себя 2 параметра:

1. student - объект, содержащий информацию о студенте
2. jobName - название новой работы студента

Функция **giveJobToStudent** выводит в модальное окно сообщение “Поздравляем! У студента **fullName** появилась новая работа! Теперь он **jobName**”, где **fullName** - это имя студента, а **jobName** - название новой работы студента.

Функция **giveJobToStudent** должна возвращать новый объект студента, в котором будут все ключи из объекта **student** и также появится новый ключ **job** со значением параметра **jobName**.

Для теста функции **giveJobToStudent** используйте следующий код:

```
const student = {
  fullName: 'Максим',
  experienceInMonths: 12,
  stack: ['HTML', 'CSS', 'JavaScript', 'React'],
}

const updatedStudent = giveJobToStudent(student, 'веб-разработчик');
/*
updatedStudent = {
    fullName: 'Максим',
    experienceInMonths: 12,
    stack: ['HTML', 'CSS', 'JavaScript', 'React'],
    job: 'веб-разработчик',
}
*/
```

Задание #5 (*дополнительное*)

Представьте, что вы разрабатываете интернет магазин по доставке еды. Вам поставили задачу подсчета итоговой суммы всех товаров в корзине.

Вам необходимо создать функцию **getTotalPriceOfShoppingBag**, которая будет принимать в себя 1 параметр **shoppingBag**. **shoppingBag** - это массив продуктов в корзине, состоящий из объектов, в каждом из которых хранится информация о названии продукта (product) и о количестве продукта в корзине (quantity).

Также у интернет-магазина есть глобальный объект, в котором хранится вся нужная информация о каждой единице продукта:

```
const groceries = {
   "Orange Juice": {
       price : 1.5,
       discount: 10,
   },
   "Chocolate": {
       price : 2,
       discount : 0,
   },
   // more items...
}
```

Функция **getTotalPriceOfShoppingBag** должна возвращать общую стоимость всех товаров в корзине с учетом скидок и с учетом указанных клиентом количеством продуктов. Итоговое значение должно быть округлено до сотых. Это можно сделать с помощью **toFixed** (https://developer.mozilla.org/...)

Посмотрите на возможно возвращаемое значение функции **getTotalPriceOfShoppingBag**:

```
const shoppingBag = [
   { product: 'Chocolate', quantity: 3 },
   { product: 'Orange Juice', quantity: 23 },
]
const totalPrice = getTotalPriceOfShoppingBag(shoppingBag);
console.log('totalPrice', totalPrice); // Возвращает 37.05
```

Задание #6 (*дополнительное*)

К вам пришёл заказчик, который является владельцем одной из игровых веб-платформ. Он хочет, чтобы вы разработали для его сайта новую игру, которая покорит сердца многих пользователей.

В игре есть 2 игрока: герой и враг. Они будут драться друг с другом. У каждого игрока есть шкала здоровья, которая изначально равна 100. При каждом ударе у противоположного игрока отнимается по 10 единиц здоровья. Побеждает тот, у кого здоровье осталось больше 0.

Сейчас вам необходимо создать функцию **startGame**, которая будет принимать в себя **2** параметра:

1. heroPlayer. Объект игрока, который содержит свойства name - имя героя; health - шкала здоровья, которая изначально равна 100; heatEnemy - функция, которая принимает в себя объект enemy и отнимает у объекта enemy 10 единиц здоровья (ключ health)
2. enemyPlayer. Объект врага, который содержит свойства name - имя героя; health - шкала здоровья, которая изначально равна 100; heatHero - функция, которая принимает в себя объект hero и отнимает у объекта hero 10 единиц здоровья (ключ health)

Внутри функции startGame вам необходимо случайным образом генерировать число от 0 до 1. Если выпадает 0, то нужно вызвать метод heatEnemy у объекта **heroPlayer**, если 1 - то heatHero у **enemyPlayer**

Для того, чтобы функция генерировала несколько раз случайные значения и игроки дрались, пока у кого-то не закончатся жизни, рекомендуется использовать цикл while, который будет выполнятся, пока у любого игрока свойство health не станет меньше либо равно нулю.

После выполнения цикла необходимо определить, какой игрок выиграл, и вывести сообщение через,где name и health - значения свойств победившего игрока.

```
alert(`${name} победил! У него осталось ${health} здоровья`)
```

Для генерации случайных значений используйте следующую функцию.

```
function getRandomNumberInRange(min, max) {
   return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Для проверки работоспособности функции **startGame** используйте данный код. В нем нет методов **heatEnemy** и **heatHero**, вам необходимо самим их реализовать.

```
const hero = {
   name: 'Batman',
   health: 100,
}
const enemy = {
   name: 'Joker',
   health: 100,
}
startGame(hero, enemy);
```

Задание #7 (*дополнительное*)

Представьте, что в полицейском участке проводится расследование. Было совершено несколько преступлений. У полицейских есть другие более приоритетные задания, поэтому они попросили вас написать программу, которая будет вычислять преступника по уже известным данным.

Ваша задача состоит в том, чтобы создать функцию **getKiller. getKiller** принимает в себя 2 параметра:

1. suspectInfo. Это объект, в котором ключи - это подозреваемые в преступлении люди, а значения - массивы людей, которых видели подозреваемые в день убийства
2. deadPeople. Это массив с именами людей, которых убил преступник

Преступником является тот, кто видел всех убитых людей в день убийства. Функция **getKiller** должна возвращать имя преступника.

Примеры результатов функции **getKiller:**

```
getKiller(
   {
      'James': ['Jacob', 'Bill', 'Lucas'],
      'Johnny': ['David', 'Kyle', 'Lucas'],
      'Peter': ['Lucy', 'Kyle'],
   },
   ['Lucas', 'Bill']
); // Убийца James
getKiller(
   {
      'Brad': [],
      'Megan': ['Ben', 'Kevin'],
```

```
        'Finn': [],
    },
    ['Ben']
); // Убийца Megan
```

Задание #8 (*дополнительное*)

Вы когда-нибудь играли в лотерею? Лотерея - это игра, в которой случайным образом определяют победителя и дают ему выигрыш. Сейчас ваша задача будет разработать логику для такой игры.

Вам необходимо создать функцию **getWinner**, которая принимает в себя 2 параметра:

1. applicants. Объект, в котором ключи - это номерки людей, по которым будет производится случайный отбор, а значения - это объекты кандидатов на выигрыш в лотерее
2. winnerObject. Это объект, в котором хранится всего 1 ключ prize, хранящий значения размера выигрыша в лотерее

Вам необходимо случайным образом выбрать победный номерок (случайный ключ в объекте applicants) и вернуть из функции **getWinner** объект, в котором будут хранится свойства из **winnerObject** и объект победителя.

Для получения случайного значения в диапазоне используйте следующую функцию:

```
function getRandomNumberInRange(min, max) {
    return Math.floor(Math.random() * (max - min)) + min;
}
```

Пример возвращаемого результата функции **getWinner:**

```
const todaysWinner = {
    prize: '10 000$',
}
const winnerApplicants = {
    '001': {
        name: 'Максим',
        age: 25,
    },
    '201': {
        name: 'Светлана',
        age: 20,
    },
    '304': {
        name: 'Екатерина',
        age: 35,
    },
}
const resultWinner = getWinner(winnerApplicants, todaysWinner);
console.log('resultWinner', resultWinner);
// { prize: '10 000$', name: 'Максим', age: 25 }
```