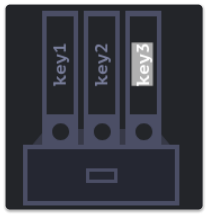
10-dars. JS objektlar (Objects), JavaScriptning ichki ob'yektlari

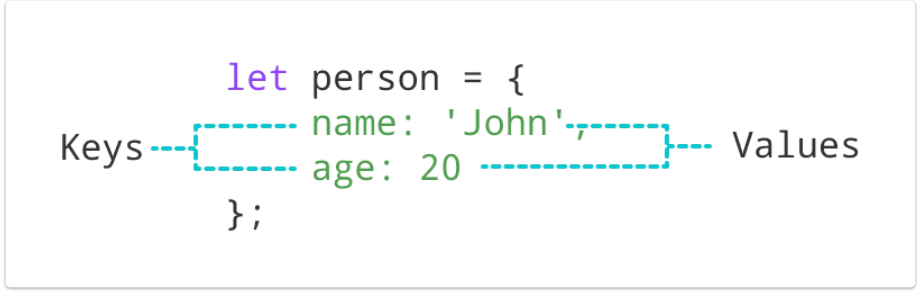
10-dars. JS objektlar (Objects), JavaScriptning ichki ob'yektlari

> *Objects are used to store keyed collections of various data and more complex entities*

> *An object can be created with figure brackets `{…}` with an optional list of properties. A property is a "key: value" pair, where `key` is a string (also called a "property name"), and `value` can be anything.*




```
let user = new Object(); // "object constructor" syntax
let user = {};  // "object literal" syntax
```



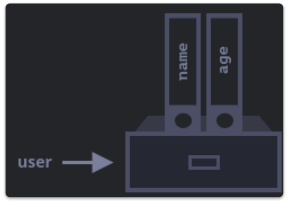

```
let user = {     // an object
  name: "John",  // by key "name" store value "John"
```

```
  age: 30         // by key "age" store value 30
};
```

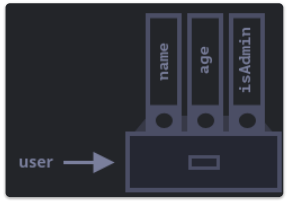
> *A property has a key (also known as "name" or "identifier") before the colon `":"` and a value to the right of it.*

In the `user` object, there are two properties:

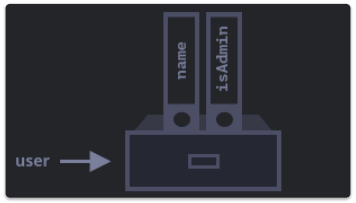
1. The first property has the name `"name"` and the value `"John"`.
2. The second one has the name `"age"` and the value `30`.




```
user.isAdmin = true;
```

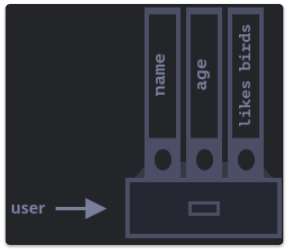



```
delete user.age;
```



We can also use multiword property names, but then they must be quoted:

```
let user = {
  name: "John",
  age: 30,
  "likes birds": true  // multiword property name must be quoted
};
```



Square brackets

```
// this would give a syntax error
user.likes birds = true
```

```
let user = {};

// set
user["likes birds"] = true;

// get
console.log(user["likes birds"]); // true

// delete
delete user["likes birds"];
```

```
let key = "likes birds";

// same as user["likes birds"] = true;
user[key] = true;
```

```
let user = {
  name: "John",
  age: 30
};

let key = "name";
console.log( user.key ) // undefined
```

Property value shorthand

```
function makeUser(name, age) {
  return {
    name: name,
    age: age,
    // ...other properties
  };
}

let user = makeUser("John", 30);
console.log(user.name); // John
```

```
function makeUser(name, age) {
  return {
    name, // same as name: name
    age,  // same as age: age
    // ...
  };
}
```

```
let user = {
  name,  // same as name:name
```

```
  age: 30
};
```

Property names limitations

```
// these properties are all right
let obj = {
  for: 1,
  let: 2,
  return: 3
};

console.log( obj.for + obj.let + obj.return );  // 6
```

Property existence test, “in” operator

```
"key" in object
```

```
let user = { name: "John", age: 30 };

console.log( "age" in user ); // true, user.age exists
console.log( "blabla" in user ); // false, user.blabla doesn't exist
```

```
let user = { age: 30 };

let key = "age";
console.log( key in user ); // true, property "age" exists
```

```
let obj = {
  test: undefined
};
```

```
console.log( obj.test ); // it's undefined, so - no such property?

console.log( "test" in obj ); // true, the property does exist!
```

The "for..in" loop

```
for (key in object) {
  // executes the body for each key among object properties
}
```

```
let user = {
  name: "John",
  age: 30,
  isAdmin: true
};

for (let key in user) {
  // keys
  console.log( key );  // name, age, isAdmin
  // values for the keys
  console.log( user[key] ); // John, 30, true
}
```

Ordered like an object

```
let codes = {
  "49": "Germany",
  "41": "Switzerland",
  "44": "Great Britain",
  // ..,
  "1": "USA"
};
```

```
for (let code in codes) {
  console.log(code); // 1, 41, 44, 49
}
```

```
let codes = {
  "+49": "Germany",
  "+41": "Switzerland",
  "+44": "Great Britain",
  // ..,
  "+1": "USA"
};

for (let code in codes) {
  console.log( +code ); // 49, 41, 44, 1
}
```

```
let user = {
  name: "John",
  surname: "Smith"
};
user.age = 25; // add one more

// non-integer properties are listed in the creation order
for (let prop in user) {
  console.log( prop ); // name, surname, age
}
```

```
let schedule = {};

console.log( isEmpty(schedule) ); // true

schedule["8:30"] = "get up";
```

```
console.log( isEmpty(schedule) ); // false
```

JavaScript Object Methods

```
const person = {
    name: 'Sam',
    age: 30,
    // using function as a value
    greet: function() { console.log('hello') }
}

person.greet(); // hello
```

Object.keys, values, entries

For plain objects, the following methods are available:

- Object.keys(obj) - returns an array of keys.
- Object.values(obj) - returns an array of values.
- Object.entries(obj) - returns an array of `[key, value]` pairs.

```
const object1 = {

a: 'somestring',

b: 42,

}

// console.log(Object.entries(object1))
```

```
const { a: key, b: value } = object1

// console.log(a, b)

// const [key, value] = Object.entries(object1)



console.log(key, value)



// for (const [key, value] of Object.entries(object1)) {

// console.log(`${key}: ${value}`)

// }



// Expected output:

// "a: somestring"

// "b: 42"
```