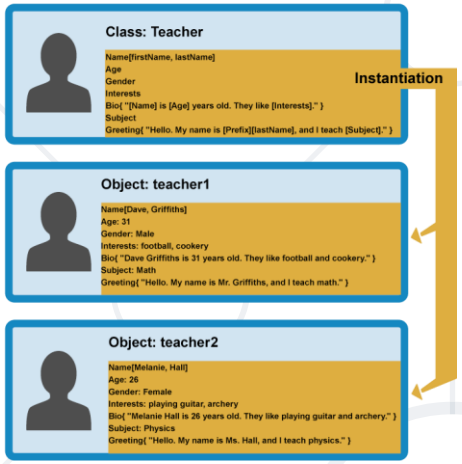
Objects and Classes

Using Objects and Classes
Defining Simple Classes



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

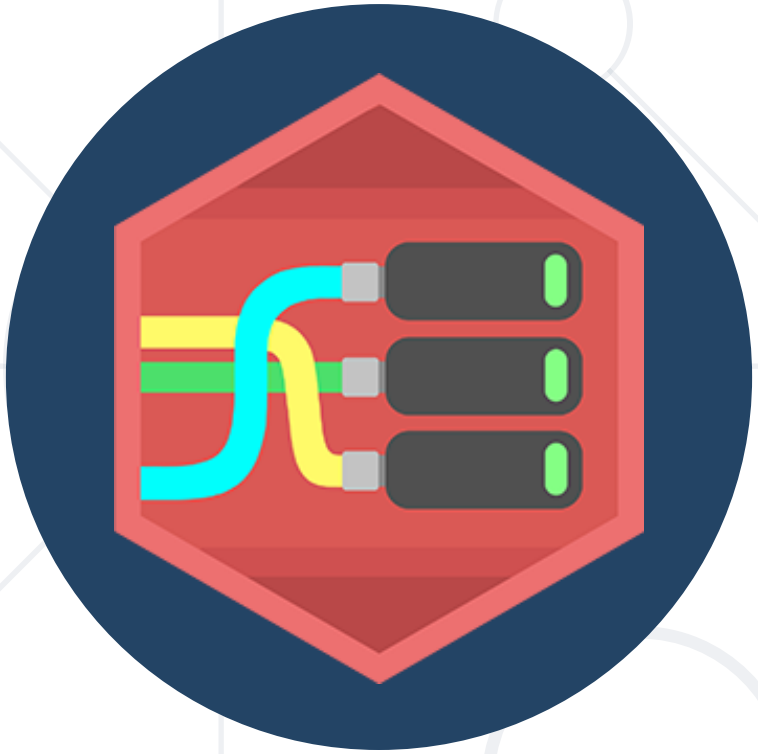
Table of Contents

Have a Question?

Software University

sli.do

#fund-js

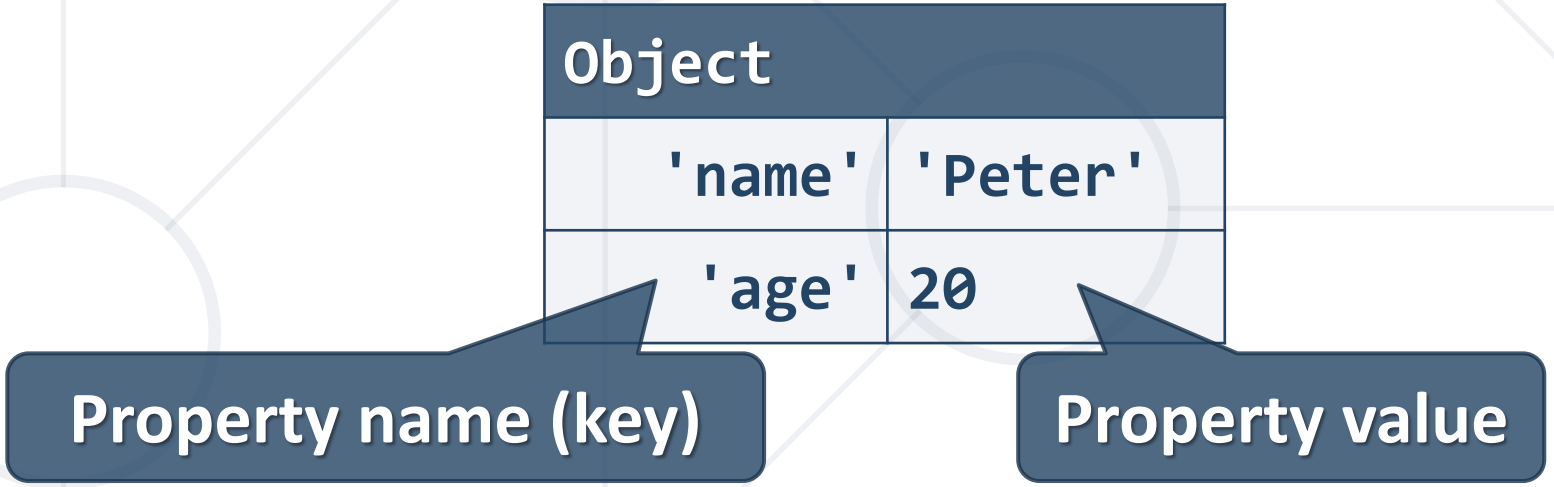


Objects

Definition, Properties and Methods

What Are Objects ?

- **Structure** of related data or functionality
- Contains **values** accessed by **string keys**
  - Data values are called **properties**
  - Function values are called **methods**



- You can **add** and **remove** properties **during runtime**

Object Definition

- We can create an object with an **object literal**

```
let person = { name:'Peter', age:20, height:183 };
```

- We can define an **empty object** and **add properties** later

```
let person = {};
person.name ='Peter';
person.age = 20;
person.hairColor = 'black';
```

```
person['lastName'] = 'Parker';
```

Access and set properties using **string indexation**

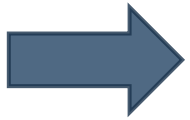
Problem: Person Info

- Create an **object** that has a first name, last name, and age
- **Return** the object at the end of your function

```
"Peter",
"Pan",
 20
```

```
firstName: Peter
lastName: Pan
age: 20
```

```
"Jack",
"Sparrow",
"unknown"
```



```
firstName: Jack
lastName: Sparrow
age: unknown
```

Solution: Person Info

- Create an object
- Set the properties `firstName`, `lastName`, and `age`
- Return the created object using the `return` keyword

```
function personInfo(firstName, lastName, age) {
  let person = {};
  person.firstName = firstName;
  // TODO: Add other properties

  return person;
}
```

Methods of Objects

- Functions within a JavaScript object are called **methods**
- We can **define** methods using several syntaxes:

```
let person = {
  sayHello: function() {
     console.log('Hi, guys');
  }
}
```

```
let person = {
  sayHello() {
     console.log('Hi, guys');
  }
}
```

- We can **add** a method to an already defined object

```
let person = { name:'Peter', age: 20 };
person.sayHello = () => console.log('Hi, guys');
```

Built-in Method Library

- Get array of all property **names** (keys)

```
Object.keys(cat); // ['name', 'age']
```

cat	
'name'	'Tom'
'age'	5

- Get array with of all property **values**

```
Object.values(cat); // ['Tom', 5]
```

- Get and array of all properties as **key-value tuples**

```
Object.entries(cat); // [['name', 'Tom'], ['age', 5]]
```

Iterate Through Keys

- Use `for-of` loop to iterate over the object properties by key:

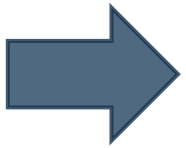
```
let obj = { name:'Peter', age:'18', grade:'5.50' };
for (let key of Object.keys(obj)) {
    console.log(`${key}: ${obj[key]}`);
}
```

Returns the value of the property

Problem: City

- Receive an object, which holds **name**, **area**, **population**, **country**, and **postcode**
- Loop through all the keys and print them with their values

```
Sofia
492
1238438
Bulgaria
1000
```

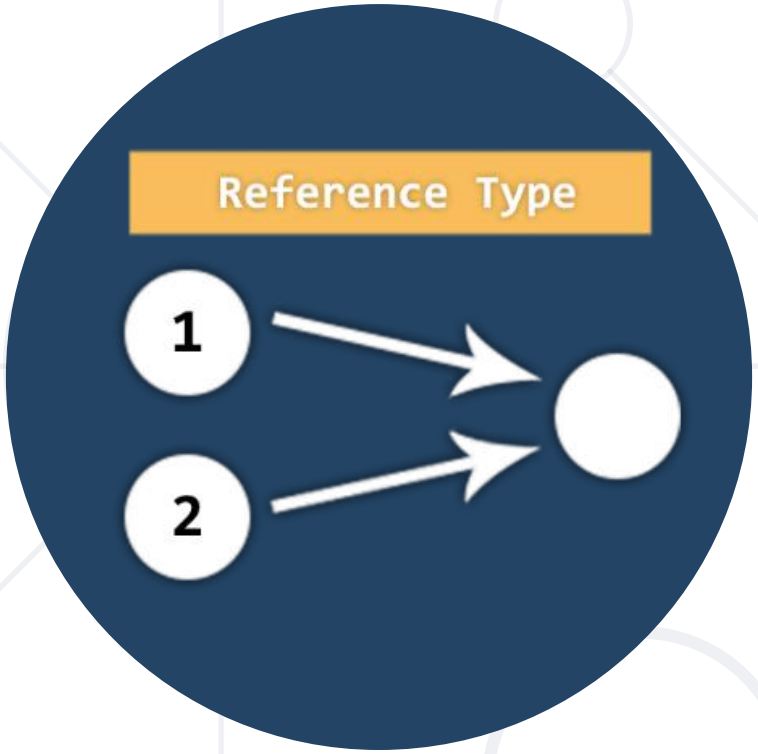


```
name -> Sofia
area -> 492
population -> 1238438
country -> Bulgaria
postCode -> 1000
```

Solution: City

- Get the object **entries**
- Loop through the object **entries** using **for-of** loop
- Print the object **keys** and **values**

```
function cityInfo(city) {
  let entries = Object.entries(city);
  for (let [ key, value ] of entries) {
    console.log(`${key} -> ${value}`);
  }
}
```



Value vs. Reference Types

Memory Stack and Heap

Reference vs. Value Types

- JavaScript has 7 data types that are copied by **value**:
  - **Boolean**, **String**, **Number**, **null**, **undefined**, **Symbol**, **BigInt**
  - These are **primitive types**
- JavaScript has 3 data types that are copied by having their **reference** copied:
  - **Array**, **Objects**, and **Functions**
  - These are all technically Objects, so we'll refer to them collectively as Objects

Example: Reference vs. Value Types

Software University

pass by reference

cup =

fillCup( )

pass by value

cup =

fillCup( )

Value Types

- If a primitive type is assigned to a variable, we can think of that variable as **containing** the primitive value

```
let a = 10;                    let c = a;
let b = 'abc';                 let d = b;
```

- They are **copied by value**

```
console.log(a, b, c, d);
// a = 10 b = 'abc' c = 10 d = 'abc'
```

Reference Types

- Variables that are assigned a non-primitive value are given a **reference** to that value

```
let arr = [];
let arrCopy = arr;
```

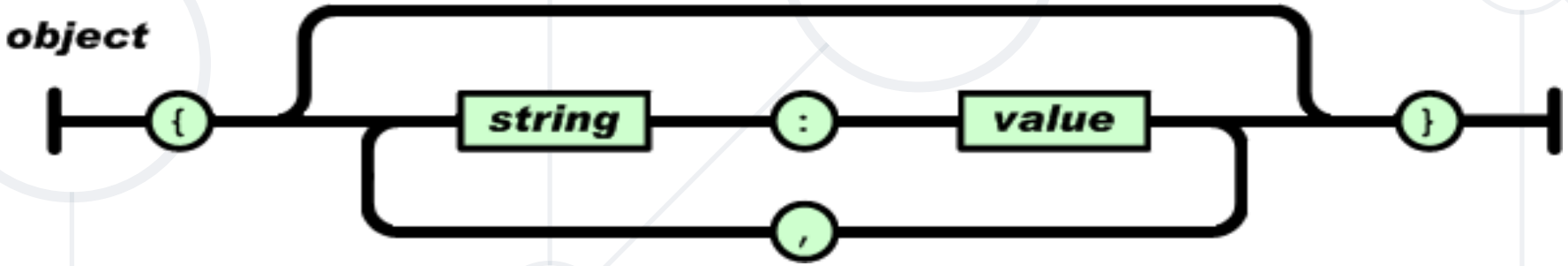
  - That reference **points to a location** in memory
  - **Variables** don't contain the value but **lead to the location**

JSON

JavaScript Object Notation

What is JSON

- **JSON** stands for **J**ava**S**cript **O**bject **N**otation
- **Open-standard** file format that uses text to transmit data objects
- JSON is **language independent**
- JSON is "**self-describing**" and easy to understand



JSON Usage

- Exchange data between **browser** and **server**
- JSON is a **lightweight** format compared to XML
- JavaScript has built-in functions to **parse JSON** so it's easy to use
- JSON uses **human-readable** text to transmit data

JSON Example

Brackets define a JSON

Keys are in double quotes

Keys and values separated by :

```
{
   "name": "Ivan",
   "age": 25,
   "grades": {
      "Math": [2.50, 3.50],
      "Chemistry": [4.50]
   }
}
```

It is possible to have nested objects

In JSON we can have arrays

JSON Methods

- We can convert object into **JSON** string using **JSON.stringify(object)** method

```
let text = JSON.stringify(obj);
```

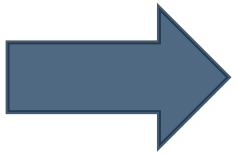
- We can convert JSON string into object using **JSON.parse(text)** method

```
let obj = JSON.parse(text);
```

Problem: Convert to Object

- Write a function, that receives a string in **JSON** format and converts it to object
- Print the entries of the object

```
'{
"name": "George",
"age": 40,
"town": "Sofia“
}'
```



```
name: George
age: 40
town: Sofia
```

Tips: Convert to Object

- Use `JSON.parse()` method to parse JSON string to an object
- Use `Object.entries()` method to get object's properties: names and values
- Loop through the entries and print them

```
function objConverter(json) {
    // TODO: Use the tips to write the function
}
```

Solution: Convert to Object

```
function objConverter(json) {
    let person= JSON.parse(json);

    let entries = Object.entries(person);

    for (let [key, value] of entries) {
        console.log(`${key}: ${value}`);
    }
}
```

Problem: Convert to JSON

- Write a function that receives a first name, last name, hair color and sets them to an object.
- Convert the object to **JSON string** and print it.

```
'George',
'Jones',
'Brown'
```

```
{"name": "George", "lastName":
"Jones", "hairColor": "Brown"}
```

Tips: Convert to JSON

- Create an object with the given input
- Use `JSON.stringify()` method to parse object to JSON string
- Keep in mind that the property name in the JSON string will be **exactly the same** as the property name in the object

```
function solve(name, lastName, hairColor){
    // TODO: Use the tips and write the code
}
```

Solution: Convert to JSON

```
function convertJSON(name, lastName, hairColor) {
    let person = {
        name,
        lastName,
        hairColor
     };
    console.log(JSON.stringify(person));
}
```

Classes

Object Models

What are Classes?

- **Templates** for creating objects
- Defines **structure** and **behavior**
- An object created by the class pattern is called an an **instance** of that class
- A class has a **constructor** – method called automatically to create an object
  - It **prepares** the new object for use
  - Can **receive** parameters and **assign** them to properties

Class Declaration

Use the `class` keyword followed by a name

```
class Student {
  constructor(name) {
    this.name = name;
  }
}
```

The `constructor` is a special method for creating and initializing an object

Class Example

- Creating a class:

this keyword is used to set a property of the object to a given value

```
class Student {
  constructor(name, grade) {
    this.name = name;
    this.grade = grade;
  }
}
```

- Creating an **instance** of the class:

```
let student = new Student('Peter', 5.50);
```

Functions in a Class

- Classes can also have functions as property, called **methods**:

```
class Dog {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(`${this.name} says Woof!`);
  }
}

let dog = new Dog('Sparky');
dog.speak(); // Sparky says Woof!
```

this in the object refers to itself

We access the method as a regular property

Problem: Cat

- Write a function that receives **array of strings** in the following format:
  `'{cat name} {age}'`
- Create a class `Cat` that receives the `name` and the `age` parsed from the input
- It should also have a method named `meow()` that will print
  `"{cat name}, age {age} says Meow"` on the console
- For each of the strings provided you must create a cat object

```
['Mellow 2','Tom 5']
```

```
Mellow, age 2 says Meow
Tom, age 5 says Meow
```

Tips: Cat

- Create a class
- Set properties name and age
- Set property '`meow`' to be a method that prints the result
- **Parse** the input data
- Create all objects using the class `constructor` and the parsed input data and store them in an array
- Loop through the array using **for…of** loop and invoke `.meow()` method

Solution: Cat

```
function catCreator(arr) {
    // TODO: Create the Cat class
     let cats = [];
     for (let i = 0; i < arr.length; i++) {
        let catData = arr[i].split(' ');
         cats.push(new Cat(catData[0], catData[1]));
     }
     // TODO: Iterate through cats[] and invoke .meow()
    using for…of loop
}
```

Live Exercises

Summary

- Objects hold **key-value pairs**
  - Access value by indexing with key
  - **Methods** are functions
- **References** point to data in memory
- **Parse** and **stringify** objects in **JSON**
- **Classes** are templates for objects

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
  - Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg